

Sequence-Aware Factorization Machines for Temporal Predictive Analytics

Tong Chen[†] Hongzhi Yin^{†*} Quoc Viet Hung Nguyen[‡] Wen-Chih Peng[§] Xue Li[†] Xiaofang Zhou[†]

[†]*School of Information Technology and Electrical Engineering, The University of Queensland,*

{tong.chen,h.yin1}@uq.edu.au {xueli,zxf}@itee.uq.edu.au

[‡]*School of Information and Communication Technology, Griffith University, quocviethung.nguyen@griffith.edu.au*

[§]*Department of Computer Science, National Chiao Tung University, wcpeng@g2.nctu.edu.tw*

Abstract—In various web applications like targeted advertising and recommender systems, the available categorical features (e.g., product type) are often of great importance but sparse. As a widely adopted solution, models based on Factorization Machines (FMs) are capable of modelling high-order interactions among features for effective sparse predictive analytics. As the volume of web-scale data grows exponentially over time, sparse predictive analytics inevitably involves dynamic and sequential features. However, existing FM-based models assume no temporal orders in the data, and are unable to capture the sequential dependencies or patterns within the dynamic features, impeding the performance and adaptivity of these methods. **Hence, in this paper, we propose a novel Sequence-Aware Factorization Machine (SeqFM) for temporal predictive analytics, which models feature interactions by fully investigating the effect of sequential dependencies. As static features (e.g., user gender) and dynamic features (e.g., user interacted items) express different semantics, we innovatively devise a multi-view self-attention scheme that separately models the effect of static features, dynamic features and the mutual interactions between static and dynamic features in three different views.** In SeqFM, we further map the learned representations of feature interactions to the desired output with a shared residual network. To showcase the versatility and generalizability of SeqFM, we test SeqFM in three popular application scenarios for FM-based models, namely ranking, classification and regression tasks. Extensive experimental results on six large-scale datasets demonstrate the superior effectiveness and efficiency of SeqFM.

I. INTRODUCTION

As an important supervised learning scheme, predictive analytics play a pivotal role in various applications, ranging from recommender systems [1], [2] to financial analysis [3] and online advertising [4], [5]. In practice, the goal of predictive analytics is to learn a mapping function from the observed variables (i.e., features) to the desired output.

When dealing with categorical features in predictive analytics, a common approach is to convert such features into one-hot encodings [6]–[8] so that standard regressors like logistic regression [9] and support vector machines [10] can be directly applied. Due to the large number of possible category variables, the converted one-hot features are usually of high dimensionality but sparse [11], and simply using raw features rarely provides optimal results. On this occasion, the interactions among different features act as the winning formula

for a wide range of data mining tasks [7], [12], [13]. The interactions among multiple raw features are usually termed as **cross features** [7] (a.k.a. multi-way features and combinatorial features). For example, individual variables *occupation* = {*lecturer*, *engineer*} and *level* = {*junior*, *senior*} can offer richer contextual information for user profiling with cross features, such as (*junior*, *engineer*) and (*senior*, *lecturer*). To avoid the high cost of task-specific manual feature engineering, **factorization machines** (FMs) [14] are proposed to embed raw features into a latent space, and model the interactions among features via the inner product of their embedding vectors.

To better capture the effect of feature interactions, variants of the plain FM are proposed, like field-aware FM for online advertising [4] and CoFM [15] for user behavior modelling. However, these variants are still constrained by their limited linear expressiveness [11] when modelling the subtle and complex feature interactions. Recently, motivated by the capability of learning discriminative representations from raw inputs, deep neural networks (DNNs) [16] have been adopted to extend the plain FM. For instance, He *et al.* [11] bridges the cross feature scheme of FM with the non-linear form of DNN, and proposes a neural factorization machine (NFM). Instead of the straightforward inner product in FM, NFM takes the sum of all features' linear pairwise combinations into a feed-forward neural network, and generates a latent representation of high-order feature interactions. With the idea of learning high-order feature interactions with DNNs, various DNN-based FMs are devised for predictive analytics [7], [17]–[21].

In short, there are two major trends of improvements over the plain FM. One is to make the model “deep” with multi-layer network structures in order to exhaustively extract useful information from feature interactions, e.g., the residual network in DeepCross [7], the pairwise product layer in PNN [21], and the compressed interaction network in xDeepFM [12]. The other is to make the model “wide” by considering multiple feature interactions in varied domains (usually coupled with “deep” structures), e.g., separately modelling user logs and texts with CoFM [15], or fusing shallow low-order output with dense high-order output via Wide&Deep [18], DeepFM [20] and xDeepFM [19]. Note that in the remainder of this paper, to avoid ambiguity, we use the term **FM-based models to imply both the plain FM and all its variants.**

However, these popular FM-based models mostly perform

*Corresponding author; contributing equally with the first author.
2020 IEEE 36th International Conference on Data Engineering (ICDE)

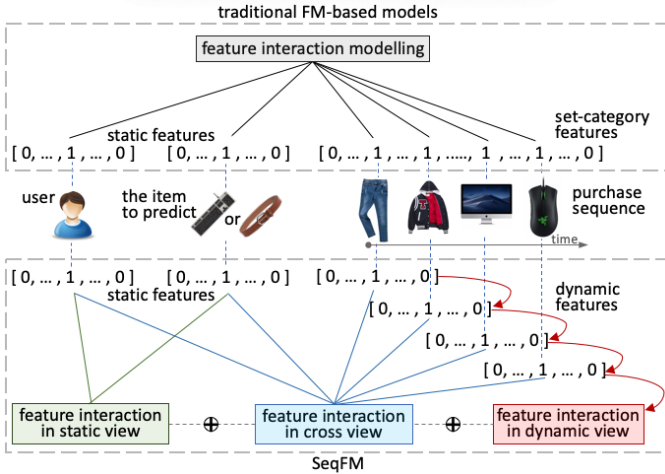


Fig. 1. The differences in feature interaction modelling between traditional FM-based models (upper part) and our proposed SeqFM (lower part). Note that the embedding process of sparse features is omitted to be succinct.

predictive analytics with the assumption that there is no temporal order in the data. As a result, regardless of the temporal information available in various prediction tasks, the data will be partitioned for training/evaluation randomly rather than chronologically, such as [8], [11], [17], [22]. Considering a real-world recommendation scenario, the time-dependent order of products purchased by each user should be considered, and the recommender system can only utilize users’ past purchase records to estimate their future preferences [23]. To this end, we focus on the problem of *temporal predictive analytics* which considers such temporal causality, and is more practical and realistic in various application scenarios.

Despite the efforts on enhancing the plain FM, all the aforementioned FM-based models still lack the consideration of the sequential dependencies within high-order feature interactions, which is proven to be critical for many temporal prediction tasks [3], [24]–[27]. With the rapidly increasing volume of web-scale data, temporal predictive analytics inevitably involves features that are dynamically changing over time, e.g., users’ shopping transactions on e-commerce platforms. We term such features the *dynamic features*. In contrast, we refer to features that stay fixed (e.g., user ID) as *static features*. Let us consider a generic item recommendation task, where the goal is to predict whether a user will buy a specific item or not, as shown in Figure 1. Apart from the one-hot encoding of both the user and candidate item, the common way for current FM-based models to account for this user’s shopping record is to derive set-category features [5], [8], [22] that mark all her/his previously bought items (see Figure 1). As is inferred from the user’s transaction (jeans → jacket → computer → mouse), the current intent of this user is to purchase accessories for her/his new computer like keyboards, rather than other clothes. However, since traditional FM-based models view all the purchased items from a constant point of time, all these dynamic features are evenly treated when modelling feature interactions. Consequently, traditional FM-based models can hardly distinguish the likelihood of purchasing a keyboard with purchasing a belt, because there are similar

items in the set-category features for both keyboards and belts, and the sequential characteristics of dynamic features cannot be properly captured. Though the recently proposed translation-based FM [28] performs recommendation by taking the sequential property of features into account, it models the influence of only the last item (i.e., the mouse), thus easily making the recommended keyboard a mismatch for the purchased computer. Moreover, for FM-based models, the deficiency of handling sequential dependencies will create a severe performance bottleneck when the diversity and amount of dynamic features grow over time.

In light of this, we aim to develop a general yet effective FM-based model to thoroughly mine the sequential information from the dynamic features for accurate temporal predictive analytics. Hence, in this paper, we propose a *Sequence-Aware Factorization Machine* (SeqFM), which is the first FM-based model to systematically combine sequential dependencies with feature interactions while inheriting the non-linear expressiveness from DNNs and retaining the compactness w.r.t. the plain FM. As demonstrated in Figure 1, SeqFM is built upon a multi-view learning scheme. Due to different semantic meanings carried by static and dynamic features, we model different types of feature interactions in three different contexts (i.e., views): static view for static features, dynamic view for dynamic features, and cross view for both. To bypass the high demand on space and time of sequential neural models using convolutional or recurrent computations, in each specific view, we leverage the self-attention mechanism [29], which is highly efficient and capable of uncovering sequential and semantic patterns between features. For the dynamic view and cross view, we further propose two masked self-attention units to respectively preserve the directional property of feature sequence and block irrelevant feature interactions. After encoding the high-order interactions between features via the multi-view self-attention, a shared residual feed-forward network is deployed to extract latent information from feature interactions.

Intuitively, compared with “deep” or “wide” FM variants, we aim to make our model “sequence-aware”, thus making full use of the contexts within dynamic features. As a flexible and versatile model, we introduce three application scenarios for SeqFM, namely ranking, classification, and regression, where corresponding experiments reveal significant improvements over existing FM-based models. Furthermore, the simple structure of SeqFM also ensures linear computational complexity and light-weight parameter size.

In summary, our work contributes in the following aspects:

- We point out that mining features’ sequential dependencies can greatly benefit the modelling of feature interactions in real-world FM-based models. We introduce, to the best of our knowledge, the first study to endow FM-based models with full sequence-awareness for temporal predictive analytics.
- We propose SeqFM, a novel sequence-aware factorization machine. SeqFM utilizes an innovative multi-view self-attention scheme to model the high-order feature interactions in a sequence-aware manner.

- We conduct extensive experiments on a wide range of benchmark datasets to showcase the superiority of SeqFM in different temporal predictive analytic tasks, validate the importance of sequence-awareness in SeqFM, and reveal promising practicality and scalability of SeqFM.

II. PRELIMINARIES

Notations. Throughout this paper, all vectors and matrices are respectively denoted by bold lower case and bold upper case letters, e.g., \mathbf{g} and \mathbf{G} . All vectors are **row vectors** unless specified, e.g., $\mathbf{x} \in \mathbb{R}^{1 \times n}$. To maintain simplicity, we use the superscripts \circ , \triangleright and \star to distinguish parameters in the static view, dynamic view and cross view, respectively.

Factorization Machines (FMs). FMs are originally proposed for collaborative recommendation. Specifically, for a given instance [*user ID*=2, *gender*=male, *cities visited*=Sydney&Shanghai], its input is a high-dimensional sparse feature $\mathbf{x} \in \{0, 1\}^{1 \times m}$ constructed by the concatenation of multiple one-hot encodings [7], [19], [21]:

$$\mathbf{x} = \underbrace{[0, 1, 0, \dots, 0]}_{\text{user ID}} \underbrace{[1, 0]}_{\text{gender}} \underbrace{[0, 1, 0, 1, 0, \dots, 0]}_{\text{cities visited}}, \quad (1)$$

where any real-valued feature (e.g., age) can also be directly included in \mathbf{x} [8], [11], but we will focus on the sparse categorical feature in our paper. Then, FMs are linear predictors that estimate the desired output by modelling all interactions between each pair of features within \mathbf{x} [14]:

$$\hat{y} = w_0 + \sum_{i=1}^m w_i x_i + \sum_{i=1}^m \sum_{j=i+1}^m \langle \mathbf{v}_i, \mathbf{v}_j \rangle \cdot x_i x_j, \quad (2)$$

where m is the total amount of features, w_0 is the global bias, w_i is the weight assigned to the i -th feature, and $\langle \cdot, \cdot \rangle$ denotes the dot product of two vectors. $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^{1 \times d}$ are corresponding embedding vectors for feature dimension i and j , while d is the embedding dimension. Thus, the first two terms in Eq.(2) can be viewed as a linear weighting scheme, while the third term models the effect of pairwise feature interactions [17].

III. SEQUENCE-AWARE FACTORIZATION MACHINES

In this section, we first overview our proposed Sequence-Aware Factorization Machines (SeqFM), and then detail each key component in the model.

Given a sparse feature vector $\mathbf{x} \in \{0, 1\}^{1 \times m}$, the output \hat{y} of SeqFM is computed via:

$$\hat{y} = w_0 + \sum_{i=1}^m w_i x_i + f(\mathbf{x}), \quad (3)$$

where the first two terms denote the linear components similar to the ones in Eq.(2), and the global bias and weights of different features are modelled respectively. $f(\mathbf{x})$ denotes our proposed factorization component. Based on the construction rule of \mathbf{x} , it can be viewed as the additive form of the one-hot encodings for all non-zero features. Thus, $\mathbf{x} = \sum_{i=1}^n \mathbf{g}_i$, where $\mathbf{g}_i = [0, \dots, 0, 1, 0, \dots, 0]$ is an m -dimensional one-hot vector corresponds to one individual non-zero feature, and n denotes the total number of non-zero features.

To conduct temporal predictive analytics with sequence-awareness, we split the original sparse feature vector \mathbf{x} into two views, namely the **static view** and **dynamic view**. In the running example of Eq.(1), user ID and gender are modelled in the static view while visited cities are modelled in the dynamic view. Then, we can obtain the static feature $\mathbf{x}^\circ \in \{0, 1\}^{1 \times m^\circ}$ and dynamic feature $\mathbf{x}^\triangleright \in \{0, 1\}^{1 \times m^\triangleright}$ where $m^\circ + m^\triangleright = m$. Correspondingly, the additive form of input features naturally splits into $\mathbf{x}^\circ = \sum_{i=1}^{n^\circ} \mathbf{g}_i^\circ$ and $\mathbf{x}^\triangleright = \sum_{i=1}^{n^\triangleright} \mathbf{g}_i^\triangleright$, where n° and n^\triangleright are the respective numbers of non-zero features in two views, and $n^\circ + n^\triangleright = n$. Here, we use feature matrices $\mathbf{G}^\circ \in \{0, 1\}^{n^\circ \times m^\circ}$ and $\mathbf{G}^\triangleright \in \{0, 1\}^{n^\triangleright \times m^\triangleright}$ to stack these sparse input vectors, of which each row is an individual one-hot vector.

It is worth mentioning that the dynamic feature matrix $\mathbf{G}^\triangleright$ is constructed in a chronological order. That is to say, $\mathbf{G}^\triangleright$ can be viewed as a sequence of dynamic features, so for row $i < j$, $\mathbf{g}_i^\triangleright \in \mathbf{G}^\triangleright$ is always observed earlier than $\mathbf{g}_j^\triangleright \in \mathbf{G}^\triangleright$. As dynamic features may update frequently over time, we pose a threshold on the maximum sequence length that our model handles. To make the notations clear, we keep using n^\triangleright to denote the maximum length for the dynamic feature sequence. If the dynamic feature sequence length is greater than the specified n^\triangleright , we consider the most recent n^\triangleright features. If the sequence length is less than n^\triangleright , we repeatedly add a padding vector $\{0\}^{1 \times m^\triangleright}$ to the top of $\mathbf{G}^\triangleright$ until the length is n^\triangleright .

So far, we can rewrite the SeqFM model in Eq.(3) as:

$$\hat{y} = w_0 + [(\mathbf{G}^\circ \mathbf{w}^\circ)^\top; (\mathbf{G}^\triangleright \mathbf{w}^\triangleright)^\top] \mathbf{1} + f(\mathbf{G}^\circ, \mathbf{G}^\triangleright), \quad (4)$$

where $\mathbf{w}^\circ \in \mathbb{R}^{m^\circ \times 1}$ and $\mathbf{w}^\triangleright \in \mathbb{R}^{m^\triangleright \times 1}$ are column vectors representing weights for all features, $[\cdot; \cdot]$ denotes the horizontal concatenation of two vectors, and $\mathbf{1}$ is a $(n^\circ + n^\triangleright) \times 1$ vector consisting of 1s. In Eq.(4), the first two terms serve the same purpose as those in Eq.(3), while $f(\mathbf{G}^\circ, \mathbf{G}^\triangleright)$ denotes the multi-view self-attentive factorization scheme. The work flow of SeqFM is demonstrated in Figure 2. In what follows, we will describe the design of $f(\mathbf{G}^\circ, \mathbf{G}^\triangleright)$ in detail.

A. Embedding Layer

As demonstrated in Figure 2, we first convert the sparse features \mathbf{G}° and $\mathbf{G}^\triangleright$ into dense representations with embedding. The embedding scheme is essentially a fully connected layer that projects each one-hot feature \mathbf{g} to a dense embedding vector as the following:

$$\mathbf{E}^\circ = \mathbf{G}^\circ \mathbf{M}^\circ, \mathbf{E}^\triangleright = \mathbf{G}^\triangleright \mathbf{M}^\triangleright, \quad (5)$$

where $\mathbf{M}^\circ \in \mathbb{R}^{m^\circ \times d}$ and $\mathbf{M}^\triangleright \in \mathbb{R}^{m^\triangleright \times d}$ are embedding matrices in the static and dynamic view, and d is the latent embedding dimension. As such, we can obtain two embedded feature matrices $\mathbf{E}^\circ \in \mathbb{R}^{n^\circ \times d}$ and $\mathbf{E}^\triangleright \in \mathbb{R}^{n^\triangleright \times d}$, where each row is a embedding vector for the original feature.

B. Static View with Self-Attention

From Eq.(2), it is clear that in the traditional FM, feature interactions are modelled in a vector-wise manner [19], where the dot product of two vectors is used. To better encode the subtle and fine-grained information, recent FM-based models

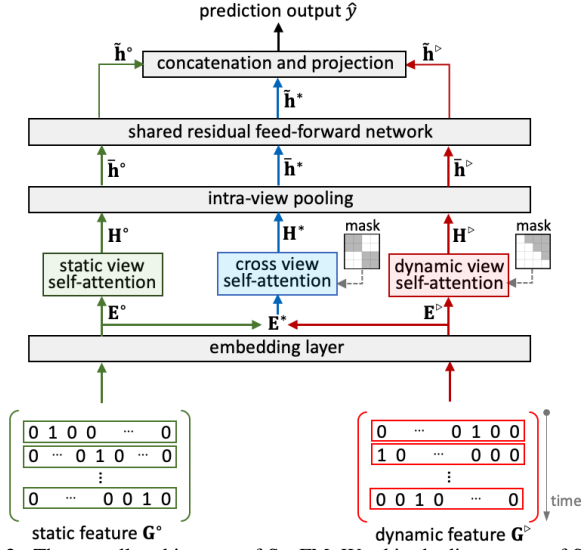


Fig. 2. The overall architecture of SeqFM. We skip the linear term of SeqFM for better readability.

[11], [17], [21] shift to bit-wise (a.k.a. element-wise) interactions of feature embeddings, such as element-wise product and weighted sum. **In order to comprehensively capture the complex interactions among features, we propose to jointly investigate vector-wise and bit-wise feature interactions with the self-attention [29], which is a linear module that can be efficiently computed.** We start with the self-attention module in the static view:

$$\mathbf{H}^o = \text{softmax}\left(\frac{\mathbf{Q}^o \mathbf{K}^{o\top}}{\sqrt{d}}\right) \mathbf{V}^o, \quad (6)$$

where $\mathbf{H}^o \in \mathbb{R}^{n^o \times d}$ is the latent interaction representation for all n^o static features, while \sqrt{d} is the scaling factor to smooth the row-wise *SoftMax* output and avoid extremely large values of the inner product, especially when the dimensionality is high. $\mathbf{Q}^o, \mathbf{K}^o, \mathbf{V}^o \in \mathbb{R}^{n^o \times d}$ respectively represent the queries, keys and values obtained using linear projection:

$$\mathbf{Q}^o = \mathbf{E}^o \mathbf{W}_Q^o, \mathbf{K}^o = \mathbf{E}^o \mathbf{W}_K^o, \mathbf{V}^o = \mathbf{E}^o \mathbf{W}_V^o, \quad (7)$$

and $\mathbf{W}_Q^o, \mathbf{W}_K^o, \mathbf{W}_V^o \in \mathbb{R}^{d \times d}$ are corresponding trainable projection weight matrices for queries, keys and values. To be concise, we reformulate the self-attention module in Eq.(6) and Eq.(7) as the following:

$$\mathbf{H}^o = \text{softmax}\left(\frac{\mathbf{E}^o \mathbf{W}_Q^o \cdot (\mathbf{E}^o \mathbf{W}_K^o)^\top}{\sqrt{d}}\right) \cdot \mathbf{E}^o \mathbf{W}_V^o, \quad (8)$$

and **each row $\mathbf{h}_i^o \in \mathbf{H}^o$ corresponds to the i -th feature.** Intuitively, we have $\mathbf{h}_i^o = w_{i1} \mathbf{v}_1^o + w_{i2} \mathbf{v}_2^o + \dots + w_{in^o} \mathbf{v}_{n^o}^o$ where $w_{i1}, w_{i2}, \dots, w_{in^o}$ are self-attentive weights assigned to projected features $\mathbf{v}_1^o, \mathbf{v}_2^o, \dots, \mathbf{v}_{n^o}^o \in \mathbf{V}^o$. In fact, because the vector-wise interactions are encoded via the self-attentive weights from the dot product scheme with *SoftMax* normalization, and the bit-wise interactions are encoded in an additive form of features, the self-attention is able to account for both bit-wise and vector-wise feature interactions between the i -th feature and all other features. Furthermore, being able to learn asymmetric interactions, the projection operation with three distinctive subspaces makes the model more flexible.

C. Dynamic View with Self-Attention

In the dynamic view, due to the nature of sequential dependencies among n^p dynamic features, the i -th dynamic feature ($i \leq n^p$) will only have the interactive influence from the previous features at j ($j \leq i$). For example, in the movie rating prediction task, we can only infer a user's rating to a new movie from her/his ratings to the movies this user has watched before. That is to say, the feature interactions in the dynamic view are **directional**. Thus, to incorporate the directional property into the self-attention mechanism, we propose the following masked self-attention to model the feature interactions in the dynamic view:

$$\mathbf{H}^p = \text{softmax}\left(\frac{\mathbf{E}^p \mathbf{W}_Q^p \cdot (\mathbf{E}^p \mathbf{W}_K^p)^\top}{\sqrt{d}} + \mathbf{M}^p\right) \cdot \mathbf{E}^p \mathbf{W}_V^p, \quad (9)$$

where $\mathbf{H}^p \in \mathbb{R}^{n^p \times d}$ carries the interaction contexts for all dynamic features, and $\mathbf{W}_Q^p, \mathbf{W}_K^p, \mathbf{W}_V^p \in \mathbb{R}^{d \times d}$. Compared with other sequential approaches like recurrent neural networks, self-attention enables vector-wise feature interactions and is more computationally efficient [25], [29]. Notably, $\mathbf{M}^p \in \{-\infty, 0\}^{n^p \times n^p}$ is a constant attention mask that allows each dynamic feature \mathbf{e}_i^p to interact with \mathbf{e}_j^p only if $j \leq i$. Specifically, in the mask \mathbf{M}^p , for its row and column indexes $i, j \leq n^p$, the value of each entry $m_{ij}^p \in \mathbf{M}^p$ is determined as:

$$m_{ij}^p = \begin{cases} 0, & \text{if } i \geq j \\ -\infty, & \text{otherwise} \end{cases}. \quad (10)$$

The Rationale of Attention Mask. We denote the matrix product of the query and key matrices in Eq.(9) as \mathbf{A} , i.e., $\mathbf{A} = \frac{\mathbf{E}^p \mathbf{W}_Q^p \cdot (\mathbf{E}^p \mathbf{W}_K^p)^\top}{\sqrt{d}} \in \mathbb{R}^{n^p \times n^p}$. Similar to [29], in \mathbf{A} , each row $a_{i1}, a_{i2}, \dots, a_{in^p}$ contains n^p interaction scores between the i -th dynamic feature and all n^p dynamic features. Then, for the i -th feature, ***SoftMax* is utilized to normalize these affinity scores to a probability distribution, i.e., $p_{i1}, p_{i2}, \dots, p_{in^p} = \text{softmax}(a_{i1}, a_{i2}, \dots, a_{in^p})$.** By adding the attention mask \mathbf{M}^p , for the i -th feature, the interaction scores from $i+1$ become $-\infty$, while the earlier ones in the sequence remain unchanged. Consequently, with the *SoftMax*, $p_{ij} \neq 0$ for $j \leq i$ while $p_{ij} \approx 0$ for $j > i$, ensuring the interaction strength on the i -th feature only associates with historical features where $j \leq i$.

D. Cross View with Self-Attention

Because static and dynamic features possess varied semantics, in the cross view, we deploy the third attention head to model how static features interact with dynamic features. Similarly, we define another masked self-attention unit below:

$$\mathbf{H}^* = \text{softmax}\left(\frac{\mathbf{E}^* \mathbf{W}_Q^* \cdot (\mathbf{E}^* \mathbf{W}_K^*)^\top}{\sqrt{d}} + \mathbf{M}^*\right) \cdot \mathbf{E}^* \mathbf{W}_V^*, \quad (11)$$

where $\mathbf{E}^* \in \mathbb{R}^{(n^o + n^p) \times d}$ represents the cross view feature matrix constructed by vertically concatenating feature matrices from both static and dynamic views along the first dimension:

$$\mathbf{E}^* = \begin{bmatrix} \mathbf{E}^o \\ \mathbf{E}^p \end{bmatrix}. \quad (12)$$

In Eq.(11), $\mathbf{H}^* \in \mathbb{R}^{(n^o + n^p) \times d}$ stacks the interaction contexts for all $n^o + n^p$ features, and there are corresponding query, key

and value projection matrices $\mathbf{W}_Q^*, \mathbf{W}_K^*, \mathbf{W}_V^* \in \mathbb{R}^{d \times d}$. $\mathbf{M}^* \in \{-\infty, 0\}^{(n^\circ + n^\triangleright) \times (n^\circ + n^\triangleright)}$ is the attention mask devised for the cross view. Each entry $m_{ij}^* \in \mathbf{M}^*$ is formulated via:

$$m_{ij}^* = \begin{cases} 0, & \text{if } i \leq n^\circ < j \text{ or } j \leq n^\circ < i \\ -\infty, & \text{otherwise} \end{cases}. \quad (13)$$

Following the explanation of the attention mask in Section III-C, our cross view attention mask blocks possible feature interactions within the same category, and only allows cross-category feature interactions (i.e., interactions between static features and dynamic features). Intuitively, with this dedicated view, the model further extracts information from the mutual interactions between static properties and dynamic properties of features in a fine-grained manner.

E. Intra-View Pooling Operation

After calculating the representations for feature interactions in all three views, we feed these latent feature matrices into our proposed intra-view pooling layer, which **compresses all latent vectors from each feature matrix into a unified vector representation**. Specifically, for $\mathbf{h}_i^\circ \in \mathbf{H}^\circ$, $\mathbf{h}_i^\triangleright \in \mathbf{H}^\triangleright$ and $\mathbf{h}_i^* \in \mathbf{H}^*$, the intra-view pooling operation is defined as:

$$\bar{\mathbf{h}}^{view} = \frac{1}{n^{view}} \sum_{i=1}^{n^{view}} \mathbf{h}_i^{view}, \quad (14)$$

where $(\bar{\mathbf{h}}^{view}, \mathbf{h}_i^{view}, n^{view}) \in \{(\bar{\mathbf{h}}^\circ, \mathbf{h}_i^\circ, n^\circ), (\bar{\mathbf{h}}^\triangleright, \mathbf{h}_i^\triangleright, n^\triangleright), (\bar{\mathbf{h}}^*, \mathbf{h}_i^*, n^* + n^\triangleright)\}$, and we use $\bar{\mathbf{h}}$, $\bar{\mathbf{h}}^\triangleright$ and $\bar{\mathbf{h}}^*$ to denote the final vector representations after the pooling operation for static view, dynamic view and cross view, respectively. Compared with the standard self-attention encoder structure in [29], the intra-view pooling operation does not introduce additional model parameters. Moreover, the intra-view pooling operation compactly encodes the information of pairwise feature interactions in the static, dynamic and cross views.

F. Shared Residual Feed-Forward Network

With the multi-view self-attention and the intra-view pooling, all feature interactions are aggregated with adaptive weights. However, **it is still a linear computation process. To further model the complex, non-linear interactions between different latent dimensions, we stack a shared l -layer residual feed-forward network upon the intra-view pooling layer:**

$$\begin{aligned} \tilde{\mathbf{h}}_{(1)}^{view} &= \bar{\mathbf{h}}^{view} + \text{ReLU}(\text{LN}(\bar{\mathbf{h}}^{view})\mathbf{W}_1 + \mathbf{b}_1), \\ \tilde{\mathbf{h}}_{(2)}^{view} &= \tilde{\mathbf{h}}_{(1)}^{view} + \text{ReLU}(\text{LN}(\tilde{\mathbf{h}}_{(1)}^{view})\mathbf{W}_2 + \mathbf{b}_2), \\ &\dots \\ \tilde{\mathbf{h}}_{(l)}^{view} &= \tilde{\mathbf{h}}_{(l-1)}^{view} + \text{ReLU}(\text{LN}(\tilde{\mathbf{h}}_{(l-1)}^{view})\mathbf{W}_l + \mathbf{b}_l), \end{aligned} \quad (15)$$

where $(\bar{\mathbf{h}}^{view}, \mathbf{h}_i^{view}) \in \{(\bar{\mathbf{h}}^\circ, \mathbf{h}_i^\circ), (\bar{\mathbf{h}}^\triangleright, \mathbf{h}_i^\triangleright), (\bar{\mathbf{h}}^*, \mathbf{h}_i^*)\}$, ReLU is the rectified linear unit for non-linear activation, while $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^{1 \times d}$ are weight and bias in each layer. Note that though the network parameters are different from layer to layer, the three views share the same feed-forward network, as shown in Figure 2. In the following, we introduce the three key components in the shared residual feed-forward network.

Residual Connections. The core idea behind residual networks is to propagate low-layer features to higher layers by residual connection [30]. By combining low-layer interaction features with the high-layer representations computed by the feed-forward network, the residual connections essentially allow the model to easily propagate low-layer features to the final layer, which can help the model enhance its expressive capability using different information learned hierarchically. **Intuitively, in our shared residual feed-forward network, to generate a comprehensive representation for feature interactions in each view, the l -th layer iteratively fine-tunes the representation learned by the $(l-1)$ -th layer (i.e., $\tilde{\mathbf{h}}_{(l-1)}^{view}$) by adding a learned residual, which corresponds to the second term in Eq.(15).**

Layer Normalization. In Eq.(15), $\text{LN}(\cdot)$ denotes the layer normalization function [31], **which is beneficial for stabilizing and accelerating neural network training process by normalizing the layer inputs across features.** Unlike batch normalization [32], in layer normalization, each sample from a batch uses independent statistics [25], and the computation at training and test times follows the same process. Specifically, for an arbitrary layer input $\tilde{\mathbf{h}}_{(l')}^{view}$, $\text{LN}(\tilde{\mathbf{h}}_{(l')}^{view})$ is calculated as:

$$\text{LN}(\tilde{\mathbf{h}}_{(l')}^{view}) = \mathbf{s} \odot \frac{\tilde{\mathbf{h}}_{(l')}^{view} - \mu}{\epsilon} + \mathbf{b}, \quad (16)$$

where $l' \leq l$ and $\tilde{\mathbf{h}}_{(l')}^{view} \in \{\tilde{\mathbf{h}}_{(l')}^\circ, \tilde{\mathbf{h}}_{(l')}^\triangleright, \tilde{\mathbf{h}}_{(l')}^*\}$. Also, \odot is the element-wise product, μ and ϵ are respectively the mean and variance of all elements in $\tilde{\mathbf{h}}_{(l')}^{view}$. Note that a small bias term will be added to ϵ in case $\epsilon=0$. The scaling weight $\mathbf{s} \in \mathbb{R}^{1 \times d}$ and the bias term $\mathbf{b} \in \mathbb{R}^{1 \times d}$ are parameters to be learned which help restore the representation power of the network.

Layer Dropout. To prevent SeqFM from overfitting the training data, we adopt dropout [33] on all the layers of our shared residual feed-forward network as a regularization strategy. In short, we randomly drop the neurons with the ratio of $\rho \in (0, 1)$ during training. Hence, dropout can be viewed as a form of ensemble learning which includes numerous models that share parameters [34]. It is worth mentioning that all the neurons are used when testing, which can be seen as a model averaging operation [33] in ensemble learning.

G. View-Wise Aggregation

With the $\tilde{\mathbf{h}}_{(l)}^\circ$, $\tilde{\mathbf{h}}_{(l)}^\triangleright$ and $\tilde{\mathbf{h}}_{(l)}^*$ calculated by the l -layer shared residual feed-forward network, we perform view-wise aggregation to combine all the information from different types of feature interactions. The final representation is generated by horizontally concatenating the latent representations from three views:

$$\mathbf{h}^{agg} = [\tilde{\mathbf{h}}_{(l)}^\circ; \tilde{\mathbf{h}}_{(l)}^\triangleright; \tilde{\mathbf{h}}_{(l)}^*], \quad (17)$$

where $\mathbf{h}^{agg} \in \mathbb{R}^{1 \times 3d}$ denotes the aggregated representation of non-linear, high-order feature interactions within SeqFM. Since the representations learned by the shared residual feed-forward network are sufficiently expressive with an appropriate network depth l , we do not apply extra learnable weights to the view-wise aggregation scheme.

H. Output Layer

After the aggregation of the latent representations from the static, dynamic and cross views, the final vector representation \mathbf{h}^{agg} is utilized to compute the scalar output for the multi-view self-attentive factorization component via vector dot product:

$$f(\mathbf{G}^\circ, \mathbf{G}^\triangleright) = \langle \mathbf{p}, \mathbf{h}^{agg} \rangle, \quad (18)$$

where $\mathbf{p} \in \mathbb{R}^{1 \times 3d}$ is the projection weight vector. At last, we summarize the entire prediction result of SeqFM as:

$$\hat{y} = w_0 + [(\mathbf{G}^\circ \mathbf{w}^\circ)^\top; (\mathbf{G}^\triangleright \mathbf{w}^\triangleright)^\top] \mathbf{1} + \langle \mathbf{p}, \mathbf{h}^{agg} \rangle. \quad (19)$$

As the scopes of both the input and output are not restricted, SeqFM is a flexible and versatile model which can be adopted for different tasks. In Section IV, we will introduce how SeqFM is applied to ranking, classification, and regression tasks as well as the optimization strategy of SeqFM.

I. Time Complexity Analysis

Excluding the embedding operation that is standard in all FM-based models, the computational cost of our model is mainly exerted by the self-attention units and the feed-forward network. As the three self-attention units are deployed in parallel, we only consider the cross view attention head that takes the most time to compute. Hence, for each training sample, the overall time complexity of these two components is $O((n^\circ + n^\triangleright)^2 d) + O(ld^2) = O((n^\circ + n^\triangleright)^2 d + ld^2)$. Because l is typically small, the dominating part is $O((n^\circ + n^\triangleright)^2 d)$. As n° is constant in the static view and n^\triangleright is fixed with a threshold, SeqFM has linear time complexity w.r.t. the scale of the data.

IV. APPLICATIONS AND OPTIMIZATION OF SEQFM

We hereby apply SeqFM to three different temporal predictive analytic settings, involving ranking, classification, and regression tasks. We also describe our optimization strategy.

A. SeqFM for Ranking

We deploy SeqFM for next-POI (point-of-interest) recommendation, which is commonly formulated as a ranking task [2], [35], [36]. For each user, next-POI recommendation aims to predict a personalized ranking on a set of POIs and return the top- K POIs according to the predicted ranking. This is accomplished by estimating a ranking score for each given user-POI pair ($user, POI$). For this ranking task, the input of SeqFM is formulated as follows:

$$\mathbf{G}^\circ = \left\{ \begin{array}{l} \mathbf{g}_1^\circ \rightarrow \text{user one-hot} \\ \mathbf{g}_2^\circ \rightarrow \text{candidate POI one-hot} \\ \vdots \\ \mathbf{g}_{n^\circ}^\circ \end{array} \right\} \text{ other static features}, \quad \mathbf{G}^\triangleright = \left\{ \begin{array}{l} \mathbf{g}_1^\triangleright \\ \mathbf{g}_2^\triangleright \\ \vdots \\ \mathbf{g}_{n^\triangleright}^\triangleright \end{array} \right\} \text{ one-hot sequence of visited POIs}. \quad (20)$$

Note that other static features include the user/POI's side information (e.g., occupation, gender, etc.) and are optional subject to availability. We denote the ($user, POI$) pair as (u, v) to be concise. For each user u , we denote an observed

user-POI interaction as a positive pair (u, v^+). Correspondingly, a corrupted user-POI pair (u, v^-) can be constructed, where v^- is a POI that user u has never visited. Thus, a training sample is defined as a triple $(u_i, v_j^+, v_k^-) \in \mathcal{S}$, and \mathcal{S} denotes the set of all training samples. Following [37], we leverage the Bayesian Personalized Ranking (BPR) loss to optimize SeqFM for the ranking task:

$$\begin{aligned} \mathcal{L} &= -\log \prod_{(u_i, v_j^+, v_k^-) \in \mathcal{S}} \sigma(\hat{y}_{ij} - \hat{y}_{ik}) \\ &= -\sum_{(u_i, v_j^+, v_k^-) \in \mathcal{S}} \log \left(\sigma(\hat{y}_{ij} - \hat{y}_{ik}) \right), \end{aligned} \quad (21)$$

where $\sigma(\cdot)$ is the *Sigmoid* function. We omit the regularization term for model parameters as the layer dropout scheme is already capable of preventing our model from overfitting. For each user u_i , \hat{y}_{ij} and \hat{y}_{ik} respectively denote the ranking score for item v_j^+ and item v_k^- . The rationale of the BPR loss is that, the ranking score for a POI visited by the user should always be higher than the ranking score for an unvisited one.

B. SeqFM for Classification

For classification task, we conduct click-through rate (CTR) prediction, which is also one of the most popular applications for FM-based models [4], [7], [18]–[20]. Given an arbitrary user and her/his previously visited links (e.g., web pages or advertisements), the target of CTR prediction is to predict whether this user will click through a given link or not. We formulate the input of SeqFM for this classification task as:

$$\mathbf{G}^\circ = \left\{ \begin{array}{l} \mathbf{g}_1^\circ \rightarrow \text{user one-hot} \\ \mathbf{g}_2^\circ \rightarrow \text{candidate link one-hot} \\ \vdots \\ \mathbf{g}_{n^\circ}^\circ \end{array} \right\} \text{ other static features}, \quad \mathbf{G}^\triangleright = \left\{ \begin{array}{l} \mathbf{g}_1^\triangleright \\ \mathbf{g}_2^\triangleright \\ \vdots \\ \mathbf{g}_{n^\triangleright}^\triangleright \end{array} \right\} \text{ one-hot sequence of clicked links}. \quad (22)$$

To enable the capability of classification, a *Sigmoid* operation is added to the output layer. To keep the notations clear, we re-formulate the \hat{y} in Eq.(19) as:

$$\hat{y} = \sigma(w_0 + [(\mathbf{G}^\circ \mathbf{w}^\circ)^\top; (\mathbf{G}^\triangleright \mathbf{w}^\triangleright)^\top] \mathbf{1} + \langle \mathbf{p}, \mathbf{h}^{agg} \rangle), \quad (23)$$

where $\sigma(\cdot)$ denotes the *Sigmoid* function. Here, the $\hat{y} \in (0, 1)$ can be viewed as the possibility of observing a ($user, link$) instance. By replacing ($user, link$) with the notion (u, v), we quantify the prediction error with log loss, which is a special case of the cross-entropy:

$$\begin{aligned} \mathcal{L} &= -\sum_{(u_i, v_j^+) \in \mathcal{S}^+} \log \hat{y}_{ij} - \sum_{(u_i, v_j^-) \in \mathcal{S}^-} \log(1 - \hat{y}_{ij}) \\ &= -\sum_{(u_i, v_j) \in \mathcal{S}} \left(y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij}) \right), \end{aligned} \quad (24)$$

where $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$ is the set of labeled (u, v) pairs. Since we only have positive labels of observed interactions denoted by $(u, v^+) \in \mathcal{S}^+$, we uniformly sample negative labels $(u, v^-) \in \mathcal{S}^-$ from the unobserved interactions during training and control the number of negative samples w.r.t. the size of the positive ones.

C. SeqFM for Regression

Finally, we apply SeqFM to a regression task, namely rating prediction which is useful for mining users' preferences and personalities [1], [8]. We use the same problem setting as [8], [14], that is, given a user and her/his rated items, we estimate this user's rating to a new target item. SeqFM takes the following as its input:

$$\mathbf{G}^\circ = \begin{bmatrix} \mathbf{g}_1^\circ \\ \mathbf{g}_2^\circ \\ \vdots \\ \mathbf{g}_{n^\circ}^\circ \end{bmatrix} \begin{cases} \rightarrow \text{user one-hot} \\ \rightarrow \text{target item one-hot} \\ \text{other static} \\ \text{features} \end{cases}, \mathbf{G}^\triangleright = \begin{bmatrix} \mathbf{g}_1^\triangleright \\ \mathbf{g}_2^\triangleright \\ \vdots \\ \mathbf{g}_{n^\triangleright}^\triangleright \end{bmatrix} \begin{cases} \text{one-hot sequence} \\ \text{of rated items} \end{cases} \quad (25)$$

We denote each $(user, item)$ pair as (u, v) . For each (u_i, v_j) , the emitted output \hat{y}_{ij} is a continuous variable that tries to match up with the ground truth rating y_{ij} . Thus, we can directly apply the squared error loss below:

$$\mathcal{L} = \sum_{(u_i, v_j) \in S} (\hat{y}_{ij} - y_{ij})^2, \quad (26)$$

where S denotes the training set. Note that sampling negative training cases is unnecessary in the conventional rating prediction task.

D. Optimization Strategy

As SeqFM is built upon the deep neural network structure, we can efficiently apply Stochastic Gradient Decent (SGD) algorithms to learn the model parameters by minimizing each task-specific loss \mathcal{L} . Hence, we leverage a mini-batch SGD-based algorithm, namely Adam [38] optimizer. For different tasks, we tune the hyperparameters using grid search. Specifically, the latent dimension (i.e., factorization factor) d is searched in $\{8, 16, 32, 64, 128\}$; the depth of the shared residual feed-forward network l is searched in $\{1, 2, 3, 4, 5\}$; the maximum sequence length n^\triangleright is searched in $\{10, 20, 30, 40, 50\}$; and the dropout ratio ρ is searched in $\{0.5, 0.6, 0.7, 0.8, 0.9\}$. We will further discuss the impact of these key hyperparameters to the prediction performance of SeqFM in Section VI. For ranking and classification tasks, we draw 5 negative samples for each positive label during training. In addition, we set the batch size to 512 according to device capacity and the learning rate to 1×10^{-4} . We iterate the whole training process until \mathcal{L} converges.

V. EXPERIMENTAL SETTINGS

In this section, we outline the evaluation protocols for our proposed SeqFM¹.

A. Datasets

To validate the performance of SeqFM in terms of ranking, classification, and regression, for each task we consider two real-world datasets, whose properties are introduced below.

¹Public access to codes:
http://bit.ly/SeqFM or http://bit.ly/bitbucket-SeqFM

TABLE I
STATISTICS OF DATASETS IN USE.

Task	Dataset	#Instance	#User	#Object	#Feature (Sparse)
Ranking	Gowalla	1,865,119	34,796	57,445	149,686
	Foursquare	1,196,248	24,941	28,593	82,127
Classification	Trivago	2,810,584	12,790	45,195	103,180
	Taobao	1,970,133	37,398	65,474	168,346
Regression	Beauty	198,503	22,363	12,101	46,565
	Toys	167,597	19,412	11,924	50,748

- **Gowalla (Ranking):** This is a global POI check-in dataset² collected from February 2009 to October 2010. Each user's visited POIs are recorded with a timestamp.
- **Foursquare (Ranking):** This POI check-in dataset³ is generated world-wide from April 2012 to September 2013, containing users' visited POIs at different times.
- **Trivago (Classification):** This dataset is obtained from the ACM RecSys Challenge⁴ in 2019. It is a web search dataset consisting of users' visiting (e.g., clicking) logs on different webpage links.
- **Taobao (Classification):** It is a subset of user shopping log data released by Alibaba⁵. We extract and sort users' clicking behavior on product links chronologically.
- **Beauty (Regression):** A series of users' product ratings⁶ are crawled from Amazon from May 1996 to July 2014, and different product categories are treated as separate datasets. Beauty is one of the largest categories.
- **Toys (Regression):** This is another Amazon user rating dataset on toys and games.

All datasets used in our experiment are in large scale and publicly available. The primary statistics are shown in Table I, where we use the word "object" to denote the POI, link, and item in different applications. Following [2], [37], [39], [40], we filter out inactive users with less than 10 interacted objects and unpopular objects visited by less than 10 users. Note that for Beauty and Toys, we directly use the provided versions without further preprocessing.

B. Baseline Methods

We briefly introduce the baseline methods for comparison below. First of all, we choose the latest and popular FM-based models as the common baselines for all ranking, classification, and regression tasks. Then, for each task, we further select two state-of-the-art methods originally proposed for the specific task scenario as an additional competitor.

- **FM:** This is the original Factorization Machine [14] with proven effectiveness in many prediction tasks [8], [22].
- **Wide&Deep:** The Wide&Deep [18] model uses a DNN to learn latent representations of concatenated features.
- **DeepCross:** It stacks multiple residual network blocks upon the concatenation layer for feature embeddings in order to learn deep cross features [7].

²<https://snap.stanford.edu/data/loc-gowalla.html>

³<https://sites.google.com/site/yangdingqi/home/foursquare-dataset>

⁴<http://www.recsyschallenge.com/2019/>

⁵<https://tianchi.aliyun.com/>

⁶<http://snap.stanford.edu/data/amazon/productGraph/>

- **NFM**: The Neural Factorization Machine [11] encodes all feature interactions via multi-layer DNNs coupled with a bit-wise bi-interaction pooling layer.
- **AFM**: The Attentional Factorization Machine [17] introduces an attention network to distinguish the importance of different pairwise feature interactions.
- **SASRec (Ranking)**: This is the Self-Attention-based Sequential Recommendation Model [25] with long-term and short-term context modelling.
- **TFM (Ranking)**: The Translation-based Factorization Machine [28] learns an embedding and translation space for each feature dimension, and adopts Euclidean distance to quantify the strength of pairwise feature interactions.
- **DIN (Classification)**: The Deep Interest Network [5] can represent users' diverse interests with an attentive activation mechanism for CTR prediction.
- **xDeepFM (Classification)**: It stands for the Extreme Deep Factorization Machine [19], which has a compressed interaction network to model vector-wise feature interactions to perform CTR prediction.
- **RRN (Regression)**: The Recurrent Recommender Network [1] is a deep autoregressive model for temporal rating prediction.
- **HOFM (Regression)**: This is the Higher-Order Factorization Machine described in [41]. HOFM improves [14] with space-saving and time-efficient kernels to allow shared parameters for prediction tasks.

C. Evaluation Metrics

To fit the scenario of temporal predictive analytics, we adopt the *leave-one-out* evaluation protocol which is widely used in the literature [5], [6], [28], [37]. Specifically, within each user's transaction, we hold out her/his last record as the ground truth for test and the second last record for validation. All the rest records are used to train the models. Set-category features are used as input for all FM-based baseline models.

Evaluating Ranking Performance. To evaluate the ranking performance, we adopt the well-established Hits Ratio at Rank K ($\text{HR}@K$) and Normalized Discounted Cumulative Gain at Rank K ($\text{NDCG}@K$) which are commonly used in information retrieval and recommender systems [2], [42], [43]. Specifically, for each positive test instance $(user, POI) \in S^{test}$, we mix the POI with J random POIs that are never visited by the user. Afterwards, we rank all these $J+1$ POIs for the user. Then, we use $\text{HR}@K$ to measure the ratio that the ground truth item has a hit (i.e., is present) on the top- K list, and use $\text{NDCG}@K$ to further evaluate whether if the model can rank the ground truth as highly as possible:

$$\text{HR}@K = \frac{\#hit@K}{|S^{test}|}, \text{NDCG}@K = \frac{\sum_{s \in S^{test}} \sum_{r=1}^K \frac{rel_{s,r}}{\log_2(r+1)}}{|S^{test}|}, \quad (27)$$

where $\#hit@K$ is the number of hits in the test set. For each test case $s \in S^{test}$, $rel_{s,r} = 1$ if the item ranked at r is the ground truth, otherwise $rel_{s,r} = 0$. We set $J=1,000$ to balance the running time and task difficulty. For K , we adopt the popular setting of 5, 10, 20 for presentation.

Evaluating Classification Performance. We adopt two evaluation metrics for the classification task, namely Area under the ROC Curve (AUC) [7], [12] and Root Mean Squared Error (RMSE) [11], [17]. For each positive test instance $(user, link) \in S^{test}$, we draw a random negative link that the user has never clicked, and predict the interaction possibility for both links.

Evaluating Regression Performance. We evaluate the regression performance with Mean Absolute Error (MAE) and Root Relative Squared Error (RRSE), which are popular among relevant research communities [3], [24], [44]. Mathematically, they are defined as follows:

$$\text{MAE} = \frac{\sum_{y \in S^{test}} |\hat{y} - y|}{|S^{test}|}, \text{RRSE} = \frac{\sqrt{\frac{\sum_{y \in S^{test}} (\hat{y} - y)^2}{|S^{test}|}}}{\sqrt{\text{VAR}_{S^{test}}}}, \quad (28)$$

where \hat{y} and y denote the predicted and real value respectively, and $\text{VAR}_{S^{test}}$ is the variance of all ground truth values.

D. Parameter Settings

To be consistent, we report the overall performance of SeqFM on all tasks with a unified parameter set $\{d = 64, l = 1, n^b = 20, \rho = 0.6\}$. Detailed discussions on the effects of different parameter settings will be shown in Section VI-B. For all baseline methods, since all tasks are conducted on standard and generic datasets with common evaluation metrics, we adopt the optimal parameters in their original works.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

Following the settings in Section V, we conduct experiments to showcase the advantage of SeqFM in terms of both effectiveness and efficiency. In particular, we aim to answer the following research questions (RQs) via experiments:

- RQ1:** How effectively SeqFM can perform temporal predictive analytics compared with state-of-the-art FM-based models.
- RQ2:** How the hyperparameters affect the performance of SeqFM in different prediction tasks.
- RQ3:** How SeqFM benefits from each component of the proposed model structure.
- RQ4:** How is the training efficiency and scalability of SeqFM when handling large-scale data.

A. Prediction Performance (RQ1)

We summarize the performance of all models in terms of ranking, classification, and regression with Table II, III, and IV respectively. We discuss our findings as follows.

Ranking Performance. The results of the ranking task (next-POI recommendation) are reported in Table II. Note that higher $\text{HR}@K$ and $\text{NDCG}@K$ values imply better prediction performance. Obviously, on both Gowalla and Foursquare, SeqFM significantly and consistently outperforms all existing FM-based models with $K \in \{5, 10, 20\}$. In particular, the advantages of SeqFM in terms of $\text{HR}@5$ and $\text{NDCG}@5$ imply that our model can accurately rank the ground truth POI in the top-5 positions, which can better suit each user's intent and boost the recommendation success rate. Though SASRec

TABLE II
RANKING TASK (NEXT-POI RECOMMENDATION) RESULTS. NUMBERS IN BOLD FACE ARE THE BEST RESULTS FOR CORRESPONDING METRICS.

Method	Gowalla						Foursquare					
	HR@K			NDCG@K			HR@K			NDCG@K		
	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
FM [14]	0.232	0.318	0.419	0.158	0.187	0.211	0.241	0.303	0.433	0.169	0.201	0.217
Wide&Deep [18]	0.288	0.401	0.532	0.199	0.238	0.267	0.233	0.317	0.422	0.165	0.192	0.218
DeepCross [7]	0.273	0.379	0.505	0.182	0.204	0.241	0.282	0.355	0.492	0.198	0.210	0.229
NFM [11]	0.286	0.395	0.525	0.199	0.236	0.264	0.239	0.325	0.435	0.170	0.198	0.225
AFM [17]	0.295	0.407	0.534	0.204	0.242	0.270	0.279	0.379	0.504	0.199	0.212	0.233
SASRec [25]	0.310	0.424	0.559	0.209	0.253	0.285	0.266	0.350	0.467	0.175	0.204	0.216
TFM [28]	0.307	0.430	0.556	0.216	0.256	0.283	0.283	0.390	0.512	0.203	0.223	0.248
SeqFM	0.345	0.467	0.603	0.243	0.283	0.316	0.324	0.431	0.554	0.227	0.262	0.293

TABLE III
CLASSIFICATION TASK (CTR PREDICTION) RESULTS. NUMBERS IN BOLD FACE ARE THE BEST RESULTS FOR CORRESPONDING METRICS.

Method	Trivago		Taobao	
	AUC	RMSE	AUC	RMSE
FM [14]	0.729	0.564	0.602	0.597
Wide&Deep [18]	0.782	0.529	0.629	0.590
DeepCross [7]	0.845	0.433	0.735	0.391
NFM [11]	0.767	0.537	0.616	0.583
AFM [17]	0.811	0.465	0.656	0.544
DIN [5]	0.923	0.338	0.781	0.375
xDeepFM [19]	0.913	0.350	0.804	0.363
SeqFM	0.957	0.319	0.826	0.335

shows promising effectiveness on Gowalla, it underperforms when facing higher data sparsity on Foursquare. Another observation is that all FM-based models with deep neural networks (i.e., Wide&Deep, DeepCross, NFM and AFM) outperform the plain FM. As a model specifically designed for sequential recommendation, TFM naturally performs better than the common baselines on both POI check-in datasets. However, SeqFM still achieves higher ranking effectiveness. This is because TFM is designed to only consider the most recently visited object (POI) in the dynamic feature sequence, while SeqFM utilizes the self-attention mechanism to extract richer information from the entire sequence.

Classification Performance. We list all the results of the classification task (CTR prediction) in Table III. A better result corresponds to a higher AUC score and a lower RMSE value. At the first glance, it is clear that our SeqFM achieves the highest classification accuracy on both Trivago and Taobao. Similar to the observations from the ranking task, existing variants of the plain FM show the benefit of adopting deep neural networks. As for the task-specific models for CTR prediction, the attentive activation unit in DIN can selectively determine the weights of different features based on a given link, while xDeepFM is able to thoroughly model the high-order interactions among different features with its dedicated interaction network. However, there is a noticeable performance gap between both additional baselines and our proposed SeqFM. This proves the insight of our work, which points out that instead of simply treating all dynamic features as flat set-category features in existing FM-based models, the sequence-aware interaction scheme for dynamic features in SeqFM is more helpful for temporal predictive analytics.

Regression Performance. Table IV reveals all models' performance achieved in the regression task (rating prediction) on Beauty and Toys. For both MAE and RRSE metrics, the lower

TABLE IV
REGRESSION TASK (RATING PREDICTION) RESULTS. NUMBERS IN BOLD FACE ARE THE BEST RESULTS FOR CORRESPONDING METRICS.

Method	Beauty		Toys	
	MAE	RRSE	MAE	RRSE
FM [14]	1.067	1.125	0.778	1.023
Wide&Deep [18]	0.965	1.090	0.753	0.989
DeepCross [7]	0.949	1.003	0.761	1.010
NFM [11]	0.931	0.986	0.735	0.981
AFM [17]	0.945	0.994	0.741	0.997
RRN [1]	0.943	0.989	0.739	0.983
HOFM [41]	0.952	1.054	0.748	1.001
SeqFM	0.890	0.975	0.704	0.956

the better. As demonstrated by the results, despite the intense competition in the regression task, SeqFM yields significant improvements on the regression accuracy over all the baselines. Furthermore, though showing competitive regression results, the additional baseline HOFM is still limited by its linear mathematical form, so approaches based on deep neural networks like RRN, NFM and AFM perform slightly better owing to their non-linear expressiveness. Apart from that, we notice that compared with the performance achieved by the plain FM, other FM-based approaches only shows marginal advantages against it in the regression task. In contrast, with 13% and 7% relative improvements on RRSE over the plain FM, our proposed SeqFM highlights the importance of fully utilizing the sequential dependencies for predictive analytics.

To summarize, the promising effectiveness of SeqFM is thoroughly demonstrated in ranking, classification, and regression tasks. In the comparison with state-of-the-art baselines on a wide range of datasets, the considerable improvements from our model further imply that SeqFM is a general and versatile model that suits different types of temporal prediction tasks.

B. Impact of Hyperparameters (RQ2)

We answer the second research question by investigating the performance fluctuations of SeqFM with varied hyperparameters. Particularly, as mentioned in Section IV-D, we study our model's sensitivity to the latent dimension d , the depth of residual feed-forward network l , the maximum sequence length n^p , as well as the dropout ratio ρ . For each test, based on the standard setting $\{d = 64, l = 1, n^p = 20, \rho = 0.6\}$, we vary the value of one hyperparameter while keeping the others unchanged, and record the new prediction result achieved. To show the performance differences, we demonstrate HR@10 for ranking, AUC for classification, and MAE for regression. Figure 3 lays out the results with different parameter settings.

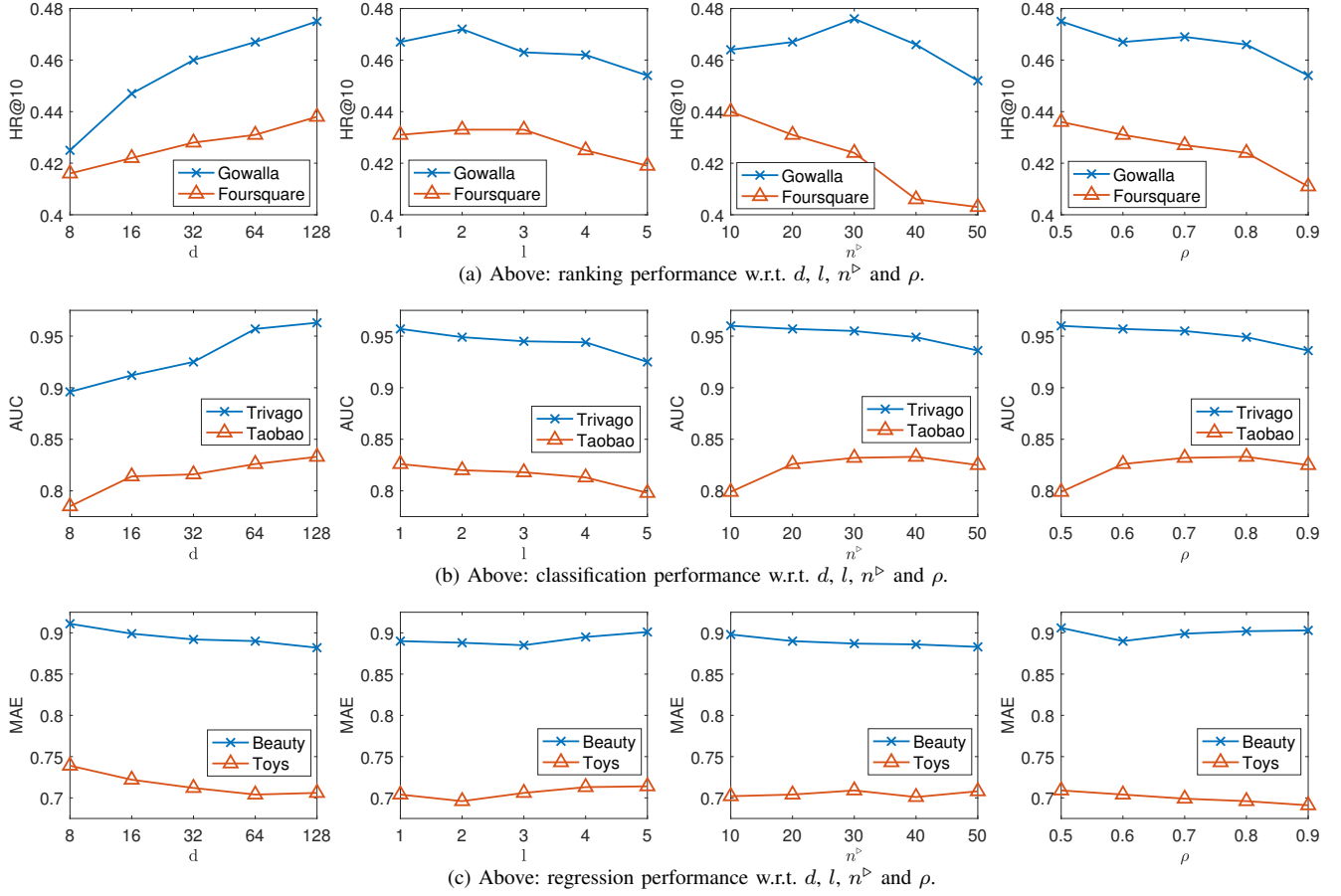


Fig. 3. Parameter sensitivity analysis.

Impact of d . The value of the latent dimension d is examined in $\{8, 16, 32, 64, 128\}$. As an important hyperparameter in deep neural networks, the latent dimension is apparently associated with the model’s expressiveness. In general, SeqFM benefits from a relatively larger d for all types of tasks, but the performance improvement tends to become less significant when d reaches a certain scale (32 and 64 in our case). It is worth mentioning that with $d = 16$, SeqFM still outperforms nearly all the baselines in the temporal prediction tasks, which further proves the effectiveness of our proposed model.

Impact of l . We study the impact of the depth of our shared residual feed-forward network with $l \in \{1, 2, 3, 4, 5\}$. For regression task, there is a slight performance growth for SeqFM as l increases. Though stacking more deep layers in the neural network may help the model yield better performance in some specific applications, for both ranking and classification tasks, SeqFM generally achieves higher prediction results with a smaller l . This is because deeper networks bring excessive parameters that can lead to overfitting, and the information learned by deeper layers may introduce noise to the model.

Impact of n^D . As can be concluded from Figure 3, SeqFM behaves differently on varied datasets when the maximum sequence length n^D is adjusted in $\{10, 20, 30, 40, 50\}$. This is due to the characteristics of sequential dependencies in different datasets. For instance, in Gowalla and Foursquare, users tend to choose the next POI close to their current check-in location (i.e., the previous POI), thus forming sequential

dependencies in short lengths. As a result, a larger n^D will take more irrelevant POIs as the input, and eventually causes the performance decrease. In contrast, in Taobao, users’ clicking behavior is usually motivated by their intrinsic long-term preferences, so a relatively larger n^D can help the model achieve better results in such scenarios.

Impact of ρ . The impact of different dropout ratios is investigated via $\rho \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Overall, the best prediction performance of SeqFM is reached when ρ is between 0.6 and 0.8. From Figure 3 we can draw the observation that a lower dropout ratio is normally useful for preserving the model’s ability to generalize to unseen test data (e.g., Foursquare and Trivago). However, on some datasets, a smaller ρ comes with lower performance (e.g., Taobao and Beauty) because too many blocked neurons may result in underfitting during training.

C. Importance of Key Components (RQ3)

To better understand the performance gain from the major components proposed in SeqFM, we conduct ablation test on different degraded versions of SeqFM. Each variant removes one key component from the model, and the corresponding results on three tasks are reported. Table V summarizes prediction outcomes in different tasks. Similar to Section VI-B, HR@10, AUC and MAE are used. In what follows, we introduce the variants and analyze their effect respectively.

Remove Static View (Remove SV). The attention head in the static view models the interactions among all the static

TABLE V

ABLATION TEST WITH DIFFERENT MODEL ARCHITECTURES. NUMBERS IN BOLD FACE ARE THE BEST RESULTS FOR CORRESPONDING METRICS, AND “↓” MARKS A SEVERE (OVER 5%) PERFORMANCE DROP.

Architecture	HR@10		AUC		MAE	
	Gowalla	Foursquare	Trivago	Taobao	Beauty	Toys
Default	0.467	0.431	0.957	0.826	0.890	0.704
Remove SV	0.455	0.420	0.892↓	0.765↓	0.959↓	0.762↓
Remove DV	0.424↓	0.396↓	0.862↓	0.731↓	0.972↓	0.772↓
Remove CV	0.430↓	0.404↓	0.963	0.754↓	0.935↓	0.763↓
Remove RC	0.457	0.431	0.898↓	0.761↓	0.918	0.719
Remove LN	0.461	0.423	0.933	0.798	0.922	0.720

features. After removing it, a noticeable performance drop has been observed, especially on classification and regression tasks. In our application of SeqFM, the static view directly models interaction between the user and the target object (i.e., POI, link, and item), which is rather important especially when the task relies on mining users’ personal preferences (e.g., the rating prediction task).

Remove Dynamic View (Remove DV). The modelling of the sequential interactions among dynamic features is crucial to the model’s performance in temporal predictive analytics. Hence, a significant (over 5%) performance decrease has appeared in all three tasks. The results verify that the sequence-awareness plays a pivotal role when prediction tasks involve dynamic features. Specifically, the most severe performance drop is exerted in the classification task, including a 10% decrease on Trivago and 12% decrease on Taobao. As these two datasets record users’ clicking behaviors on the product links provided, the entire dynamic feature sequence carries the long-term preference of each user. So, considering the dynamic dependencies can actually help our model accurately capture the rich information from the dynamic features, and eventually yield competitive prediction effectiveness.

Remove Cross View (Remove CV). Similar to the effect of discarding the dynamic view, SeqFM suffers from the obviously inferior performance (over 5% drop) regarding 5 datasets after the cross view with self-attention head is removed. Apparently, in this degraded version of SeqFM, the interactions between static features and dynamic features are discarded, leading to a significant loss of information. This verifies the contribution of the self-attention head in the cross view to our model’s final performance in all three tasks.

Remove Residual Connections (Remove RC). Without residual connections, we find that the performance of SeqFM gets worse, especially on Trivago and Taobao datasets. Presumably this is because information in lower layers (i.e., the output generated by the attention head) cannot be easily propagated to the final layer, and such information is highly useful for making predictions, especially on datasets with a large amount of sparse features.

Remove Layer Normalization (Remove LN). The layer normalization operation is introduced mainly for the purpose of stabilizing the training process by scaling the input with varied data scales for deep layers. Removing the layer normalization also shows a negative impact on the prediction performance, especially in the regression task where the properly normalized features can usually generate better results.

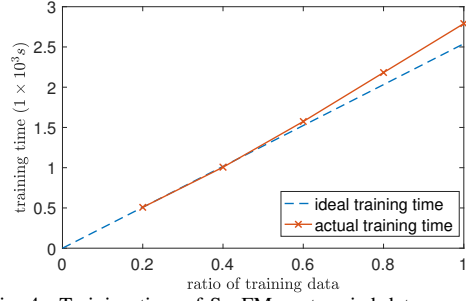


Fig. 4. Training time of SeqFM w.r.t varied data proportions.

D. Training Efficiency and Scalability (RQ4)

We test the training efficiency and scalability of SeqFM by varying the proportions of the training data in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, and then report the corresponding time cost for the model training. It is worth noting that the Trivago dataset is used for scalability test since it contains the most instances. The growth of training time along with the data size is shown in Figure 4. When the ratio of training data gradually extends from 0.2 to 1.0, the training time for SeqFM increases from 0.51×10^3 seconds to 2.79×10^3 seconds. It shows that the dependency of training time on the data scale is approximately linear. Hence, we conclude that SeqFM is scalable to even larger datasets.

VII. RELATED WORK

In a nutshell, the ultimate goal of predictive analytics is to learn an effective predictor that accurately estimates the output according to the input features, where classic predictive methods like support vector machines (SVMs) [10] and logistic regression (LR) [9] have gained extensive popularity. Distinct from the continuous raw features from images and audios, features from the web-scale data are mostly discrete and categorical [11], and are therefore represented by high-dimensional but sparse one-hot encodings. **When performing predictive analytics under the sparse setting, it is crucial to account for the interactions between features [7].** With the core idea of modelling high-order interactions among features, factorization machines (FMs) [14] are widely used for predictive analytics on very sparse data where SVMs fail. Other linear FM-based models are proposed, such as CoFM [15], field-aware FM [4] and importance-aware FM [45]. However, as stated in many literatures [11], [17], [19], [21], these models show limited effectiveness in mining high-order latent patterns or learning quality feature representations.

Another line of research on FM-based models for predictive analytics incorporates deep neural networks (DNNs) [11], [19], [21], [46]. For example, the FM-supported neural network (FNN) [46] as well as the neural factorization machine (NFM) [11] are proposed to learn non-linear high-order feature interactions. They both use the pre-trained factorization machines for feature embedding before applying DNNs. Qu *et al.* proposes a product-based neural network (PNN) [21], which introduces a product layer between embedding layer and DNN layer, and does not rely on pre-trained FM parameters. More recently, hybrid architectures are introduced in Wide&Deep [18], DeepFM [20] and xDeepFM [19] by

combining shallow components with deep ones to capture both low- and high-order feature interactions. Unfortunately, as discussed in Section I, existing FM-based models lack the consideration of sequential dependencies within feature interactions, and most of them ignore the temporal orders within the data. On the contrary, our SeqFM extends FM-based models to the temporal predictive analytics, and utilizes the sequential dependencies within dynamic features to yield superior prediction performance.

VIII. CONCLUSION

In this paper, we propose SeqFM, a sequence-aware factorization machine for temporal predictive analytics. For the first time, we incorporate sequential dependencies into FM-based models by proposing a novel multi-view self-attention scheme to model the interactions between different features. SeqFM is then successfully applied to three different temporal prediction tasks including ranking, classification and regression. The experimental results showcase that SeqFM is a powerful yet general model that can yield superior performance in a wide range of real-world applications.

ACKNOWLEDGMENT

This work is supported by Australian Research Council (Grant No. DP190101985, DP170103954 and DP160104075).

REFERENCES

- [1] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *WSDM*, 2017, pp. 495–503.
- [2] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation," in *SIGKDD*, 2017, pp. 1245–1254.
- [3] T. Chen, H. Yin, H. Chen, L. Wu, H. Wang, X. Zhou, and X. Li, "Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction," in *ICDM*, 2018, pp. 49–58.
- [4] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *RecSys*, 2016, pp. 43–50.
- [5] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *SIGKDD*, 2018, pp. 1059–1068.
- [6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.
- [7] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao, "Deep crossing: Web-scale modeling without manually crafted combinatorial features," in *SIGKDD*, 2016, pp. 255–262.
- [8] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *SIGIR*, 2011, pp. 635–644.
- [9] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.
- [10] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *TIST*, p. 27, 2011.
- [11] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *SIGIR*, 2017, pp. 355–364.
- [12] J. Lian, F. Zhang, M. Hou, H. Wang, X. Xie, and G. Sun, "Practical lessons for job recommendations in the cold-start scenario," in *RecSys Challenge*, 2017, p. 4.
- [13] H. Yin, H. Chen, X. Sun, H. Wang, Y. Wang, and Q. V. H. Nguyen, "Sptf: a scalable probabilistic tensor factorization model for semantic-aware behavior prediction," in *ICDM*, 2017, pp. 585–594.
- [14] S. Rendle, "Factorization machines," in *ICDM*, 2010, pp. 995–1000.
- [15] L. Hong, A. S. Doumith, and B. D. Davison, "Co-factorization machines: modeling user interests and predicting individual decisions in twitter," in *WSDM*, 2013, pp. 557–566.
- [16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [17] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," *IJCAI*, pp. 3119–3125, 2017.
- [18] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *DLRS*, 2016, pp. 7–10.
- [19] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," *SIGKDD*, pp. 1754–1763, 2018.
- [20] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.
- [21] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *ICDM*, 2016.
- [22] S. Rendle, "Factorization machines with libfm," *TIST*, p. 57, 2012.
- [23] P. Campos, F. Díez, and I. Cantador, "Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols," *User Modeling and User-Adapted Interaction*, pp. 67–119, 2014.
- [24] G. Lai, W. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," *SIGIR*, pp. 95–104, 2018.
- [25] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," *ICDM*, pp. 197–206, 2018.
- [26] W. Wang, H. Yin, S. Sadiq, L. Chen, M. Xie, and X. Zhou, "Spore: A sequential personalized spatial item recommender system," in *ICDE*, 2016, pp. 954–965.
- [27] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Quoc Viet Hung, "Streaming session-based recommendation," in *SIGKDD*, 2019.
- [28] R. Pasricha and J. McAuley, "Translation-based factorization machines for sequential recommendation," in *RecSys*, 2018, pp. 63–71.
- [29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
- [31] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
- [33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, pp. 1929–1958, 2014.
- [34] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio, "An empirical analysis of dropout in piecewise linear networks," *arXiv preprint arXiv:1312.6197*, 2013.
- [35] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen, "Adapting to user interest drift for poi recommendation," *TKDE*, pp. 2566–2581, 2016.
- [36] W. Wang, H. Yin, X. Du, Q. V. H. Nguyen, and X. Zhou, "Tpm: A temporal personalized model for spatial item recommendation," *TIST*, vol. 9, no. 6, p. 61, 2018.
- [37] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009.
- [38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2015.
- [39] H. Li, Y. Ge, R. Hong, and H. Zhu, "Point-of-interest recommendations: Learning potential check-ins from friends," in *SIGKDD*, 2016.
- [40] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.
- [41] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *NIPS*, 2016, pp. 3351–3359.
- [42] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li, "Air: Attentional intention-aware recommender systems," in *ICDE*, 2019, pp. 304–315.
- [43] H. Yin, B. Cui, L. Chen, Z. Hu, and C. Zhang, "Modeling location-based user rating profiles for personalized recommendation," *TKDD*, vol. 9, no. 3, p. 19, 2015.
- [44] T. Chen, H. Yin, H. Chen, H. Wang, X. Zhou, and X. Li, "Online sales prediction via trend alignment-based multitask recurrent neural networks," *KAIS*, pp. 1–29, 2019.
- [45] R. J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, and M. Finegold, "Predicting response in mobile advertising with hierarchical importance-aware factorization machine," in *WSDM*, 2014, pp. 123–132.
- [46] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *ECIR*, 2016, pp. 45–57.