Expert Systems With Applications 209 (2022) 118177

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Customer churn prediction for web browsers

Xing Wu[a,b,*], Pan Li[a], Ming Zhao[c], Ying Liu[d,**], Rubén González Crespo[e], Enrique Herrera-Viedma[f,g]

[a] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
[b] Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China
[c] CSSC Ocean Exploration Technology Research Institute Co., Ltd., WuXi, 214000, China
[d] School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China
[e] Department of Computer Science, Universidad Internacional de La Rioja, Logroño, La Rioja 26006, Spain
[f] Andalusian Research Institute in Data Science and Computational Intelligence, Granada 18071, Spain
[g] Department of Electrical and Computer Engineering, Faculty of Engineering Abdulaziz University, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Keywords:
Churn prediction
MBST
Attention mechanism
Tree-based models
Sequence models

ABSTRACT

In the competitive web browser market, identifying potential churners is critical to decreasing the loss of existing customers. Churn prediction based on customer behaviors plays a vital role in customer retention strategies. However, traditional churn prediction algorithms such as Tree-based models cannot exploit the temporal characteristics of browser customers behaviors, while sequence models cannot explicitly extract the information between multiple behaviors. To meet this challenge, we propose a novel model named Multivariate Behavior Sequence Transformer (MBST) with two complementary attention mechanisms to explore the temporal and behavioral information separately. Furthermore, a Tree-based classifier is attached for churn prediction instead of using the multilayer perceptron. Extensive experiments on a real-world Tencent QQ browser dataset with over 600,000 samples demonstrate that the proposed MBST achieves the F-score of 82.72% and the Area Under Curve (AUC) of 93.75%, which significantly outperforms state-of-the-art methods in terms of churn prediction.

1. Introduction

As the most significant source of revenue, customers are the lifeblood of the business in web browser marketplaces. In the saturated market, however, attracting a new customer costs 5–10 times more than keeping an existing one (Jamalian & Foukerdi, 2018). Therefore, companies focus marketing efforts on customer retention rather than customer acquisition. Building an effective and trustworthy customer churn prediction model is necessary for client retention.

The accuracy of churn prediction models is particularly critical in implementing customer retention strategies, especially in industries with large numbers of customers. Typically, web browser applications have a large user base, such as the Tencent QQ browser, one of the most popular web browsers in China and has more than 89 million users. Therefore, the prediction model's performance is crucial since a slight increase in accuracy can retain a large number of customers and recover huge profits. However, it is a rather challenging problem embodied in both data and algorithms.

From the data perspective, researchers tend to collect data in many tasks over a long period of time to get a more thorough and detailed grasp of patterns while enhancing algorithm performance simultaneously. However, new customers may churn before their long-term behavior history is gathered. It will be difficult to collect sufficient user behavior information before the churn prediction is made. What is more, web browsers are prohibited from collecting users' personal identification information to protect privacy. Each anonymous user is identified by their unique ID, not related to identity information like age and sex. Only the behavior information is available (e.g., which domain they access and how long they use the application on a specific day). As a result, one of our main issues is how to extract more useful information from the limited data.

In this work, observable user history data is restricted to 14 days for timely predictions, and hybrid features are introduced as our algorithm's input for higher prediction accuracy on limited data. Specifically, hybrid features consist of two different feature categories: *static*

* Corresponding author at: School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China.
** Corresponding author.
E-mail addresses: xingwu@shu.edu.cn (X. Wu), indeed@shu.edu.cn (P. Li), nudtzhaoming08@163.com (M. Zhao), liuy@ucas.ac.cn (Y. Liu), ruben.gonzalez@unir.net (R.G. Crespo), viedma@decsai.ugr.es (E. Herrera-Viedma).

https://doi.org/10.1016/j.eswa.2022.118177
Received 16 December 2021; Received in revised form 15 June 2022; Accepted 14 July 2022
Available online 25 July 2022

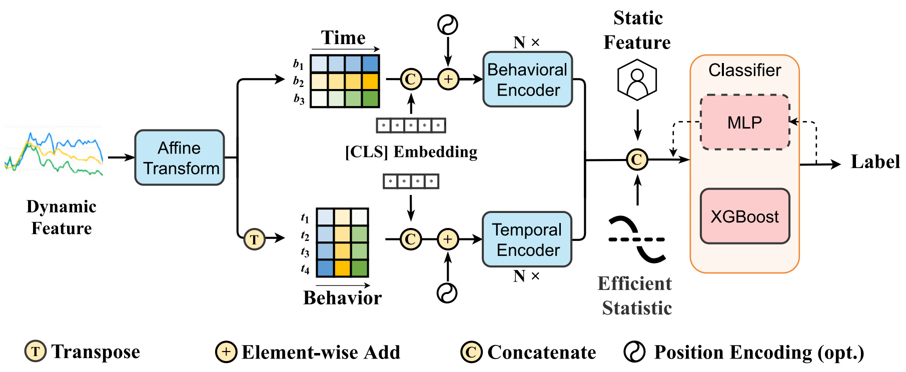

Fig. 1. Overview of the proposed architecture.

features and *dynamic* features from behavior information. The former is a set of weights assigned by users' preferences, such as education, news, etc. The latter are multivariate behavioral sequences whose values change over time (e.g., the number of accessed domains and the usage time). The hybrid features will be specifically described in Section 4.1.

From the algorithmic perspective, SVM-based, Tree-based and NN-based methods are available for prediction tasks. The Tree-based ensemble learning methods based on bagging or boosting, such as Random Forests (RF) (Biau & Scornet, 2016) and Gradient Boosting Decision Tree (GBDT) (Friedman, 2001) have better generalization performance for structured static data. They outperform SVM-based and NN-based approaches in user profile analysis applications with static data. However, Tree-based ensemble learning methods treat sequential features as the same as static features, ignoring the temporal nature of customer behavior. Previous works primarily recognize customer dynamic behavior patterns by NN-based sequence models (Alboukaey, Joukhadar, & Ghneim, 2020; Kristensen & Burelli, 2019; Tan et al., 2018). However, these methods focus on modeling the temporal relationships and dismiss the behavioral relationships, which is distinct for churn prediction in web browsers (i.e., there are crucial potential correlations between different customer behaviors). For instance, the number of domains accessed by users is often positively correlated with the users' usage time. That is to say, it is of great importance to model the temporal and behavioral correlations.

On the one hand, it is essential to investigate the combination of static and dynamic features for mining informative knowledge with limited data. Tree ensemble models outperform SVM-based and NN-based approaches for structured data in many real-world applications, whereas sequence models have better performance with dynamic data. It is natural to employ a sequence model for dynamic data representation and a tree ensemble model for static data representation and aggregated feature classification. On the other hand, it is essential to model the temporal and behavioral correlations. The Transformer (Vaswani et al., 2017) architecture contains an attention mechanism to model relationships between sequences, and it has provided quantitative breakthroughs in natural language processing (Devlin, Chang, Lee, & Toutanova, 2018). Promising results have been obtained for other tasks, including computer vision (Liu et al., 2021) and time series forecasting (Zhou et al., 2021). This work proposes a Transformer-based architecture to model temporal and behavioral dependency. We are the first to apply the Transformer architecture to churn prediction and convert it to a novel model named Multivariate Behavior Sequence Transformer (MBST). MBST uses two complementary attention mechanisms to explore the modeling of temporal and behavioral dependencies. Furthermore, an external Tree-based classifier is attached for churn prediction instead of using the multilayer perceptron as the classification head. An overview of our architecture is depicted in Fig. 1, and it will be introduced in Section 3.

The main contributions of this study are as follows:

- We investigate the effect of different categories of data and combine static and dynamic data collected from users to dig informative knowledge regarding user behavior.
- We propose the Multivariate Behavior Sequence Transformer (MBST) with two complementary attention mechanisms and attach an external Tree-based classifier leading to a higher prediction accuracy.
- We conduct extensive experiments to demonstrate the superiority of our approach, and visualize the attention weights to explore the interpretability of churn prediction.

The remainder of this paper is organized as follows. Section 2 summarizes the related work of churn prediction. Then, the MBST is introduced in Section 3. Section 4 presents the experimental results in detail. Finally, conclusions and future work are drawn in Section 5.

2. Related work

This section introduces the churn prediction research in various applications based on different data types. Moreover, a comprehensive summary of the current research state and applications are presented for the two primary prediction approaches (Tree-based methods and NN-based sequence models).

2.1. Churn prediction

Churn prediction is one of the most crucial stages in customer relationship management (CRM) (Ahn, Hwang, Kim, Choi, & Kang, 2020; Arivazhagan & Sankara, 2020) hence developing an accurate and effective client churn prediction model is key for customer retention.

In recent decades, extensive researches have been conducted on this issue using static features, especially in industries with large numbers of customers, such as the telecommunication (Amin et al., 2019; Lu, Lin, Lu, & Zhang, 2012), banking (Bilal Zorić, 2016; Keramati, Ghaneei, & Mirmohammadi, 2016), insurance (Dela Llave, López, & Angulo, 2019; Sundarkumar & Ravi, 2015), social networking (Backiel, Verbinnen, Baesens, & Claeskens, 2015; Farquad, Ravi, & Raju, 2014; Óskarsdóttir et al., 2017), credit card (Keramati et al., 2016; Rajamohamed & Manokaran, 2018), and online game industries (Milošević, Živić, & Andjelković, 2017; Periáñez, Saas, Guitart, & Magne, 2016; Tamassia et al., 2016).

In addition, many researchers build churn prediction models based on the dynamic feature. Alboukaey et al. (2020) represented customer's activities as a multivariate time series and proposed four models to predict daily churn, including RFM-based, statistics-based, CNN-based models, and LSTM-based model. Óskarsdóttir, Van Calster, Baesens, Lemahieu, and Vanthienen (2018) proposed a novel method to extract time-series data from call networks to represent dynamic customer behavior and use the similarity forests method together with some of the proposed extensions to predict churn. Leung and Chung (2020) used trend modeling to capture the change in customer behavior, and the results showed that data from multiple time periods helped improve precision and recall.

However, few kinds of research focus on churn prediction in the scenario of web browsers handling diverse data types. As a result, our primary purpose is to make up for gaps in the field despite the fact that data and algorithms are restricted.

2.2. Tree-based learning

The decision tree is a traditional method for classification problems, including churn prediction (De Caigny, Coussement, & De Bock, 2018; Nie, Rowe, Zhang, Tian, & Shi, 2011). Its superior strength is the efficient selection of global features with the most statistical information gain (Grabczewski & Jankowski, 2005) and high interpretability. However, A single decision tree model tends to produce an overly complex model with poor generalization performance, especially for high-dimensional data (e.g., the user data of our dataset have over 100 dimensions). Therefore, ensemble learning (bagging and boosting) based on the decision tree is a typical method to reduce the variance for improving classification performance.

Bootstrap aggregation, also known as bagging, employs a randomly selected portion of the training data and assigns equal weights to each model in the ensemble voting. For example, the random forest algorithm (Ho, 1998) combined random decision trees with bagging to promote model variance.

Boosting is progressively forming an ensemble by training each new model instance to emphasize the previous models' misclassified training instances. Boosting is more accurate than bagging in certain instances, but it is also more prone to over-fitting the training data. AdaBoost (Freund, Schapire, et al., 1996) is the first adaptive boosting algorithm as it dynamically changes its parameters to the data based on the actual performance in the current iteration. Gradient Boosting Decision Tree (GBDT) (Friedman, 2001) identifies difficult observations by large residuals computed in the previous iterations (Mayr, Binder, Gefeller, & Schmid, 2014). The boosting technique that optimizes the empirical risk using the steepest gradient descent in function space significantly developed the statistical aspect of boosting. XGBoost (Chen & Guestrin, 2016) is a recent ensemble decision tree approach that dominates most recent data science competitions. According to some studies (Ahmad, Jafar, & Aljoumaa, 2019), using the XGBoost algorithm for churn prediction yielded the best results.

Our experimental results show that ensemble learning methods can outperform sequence models when the representation capacity is improved by attention and linear transformation.

2.3. Sequence modeling

Under a discrete stochastic modeling framework, Hidden Markov Model (HMM) is an effective method to deal with sequential signals (Rabiner & Juang, 1986). However, it is a memoryless model that does not make effective use of contextual information. Deep learning methods have been very popular in recent years, and Recurrent Neural Networks (RNN) provide another alternative for incorporating temporal dynamics. However, it cannot combat the vanishing or exploding gradient problem (Medsker & Jain, 2001). Long Short-Term Memory Model (LSTM), a variant of RNN, has become a definitive benchmark for sequential data, not only on its ability to ameliorate the vanishing or exploding gradient problem but also on its ability to capture long term dependencies allowing the model to keep long term explanatory observations to make classifications and predictions in memory (Van Houdt, Mosquera, & Nápoles, 2020). Yang, Shi, Jie, and Han (2018) proposed a parallel LSTM framework with an attention mechanism to perform churn prediction in a social application by utilizing dynamic user data and achieved excellent results.

Recently, researches with the vanilla transformer architecture shows significantly more parallelization than the RNN-based sequence model in different domains. The vanilla transformer architecture is an encoder–decoder architecture proposed by Vaswani et al. (2017). The encoder is composed of a stack of multiple identical layers, where each layer containing two sublayers. The first sublayer is a multi-headed self-attention mechanism followed by residual connections, and the second is a simple-wise fully connected feed-forward networks. The decoder also consists of a stack of layers. The first two layers are the same as the encoder layers, and the third is multi-head attention over the output of the encoder stack. Its variants are used in many fields for process sequential data, including financial, medical, transportation applications, etc. For example, Informer enhances the prediction capacity in the long sequence time-series forecasting problem as a variant of transformers (Zhou et al., 2021). Non-Autoregressive Spatial–Temporal Transformer (NAST) building a bridge by a learned temporal influence map to fill the gaps between the spatial and temporal attention so that spatial and temporal dependencies can be processed integrally (Chen et al., 2021).

In this work, the Transformer architecture is applied to churn prediction for the first time, and we primarily focus on modeling both behavioral and temporal dependencies. However, off-the-shelf variants of Transformer architecture, which model both spatial and temporal dependencies (Aksan, Cao, Kaufmann, & Hilliges, 2020; Plizzari, Cannici, & Matteucci, 2021; Xu et al., 2020), are not suitable for churn prediction since the semantic space of spatial (behavioral) dimension is inconsistent (i.e., every behavioral sequence expresses different meanings), so they cannot be transformed via the same linear mapping layer. As a result, an affine transform block and two different attention mechanisms are introduced to address this issue.

3. Methodology

In this section, we first provide the formal definition of the churn prediction problem. Then the overview of the Multivariate Behavior Sequence Transformer is presented in Section 3.2. In addition, we will go through each of the MBST's major components in depth. The optimization method for our model is provided in the end.

3.1. Problem formulation

We pose churn prediction as a classic binary classification task where the problem becomes accurately classify a specific user $\boldsymbol{x}_i$ will terminate the service or not in the near future. Supposing a set of N samples $\mathbb{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, a prediction model can be defined as a function $\hat{y}_i = f(\boldsymbol{x}_i)$ where i denotes the order number, $f(\cdot)$ is the model operation. $\hat{y}_i \in \mathcal{Y} = \{0, 1\}$ is a binary value where the positive value represents that the customer will leave in the near future.

Each $\boldsymbol{x}_i$ is composed of three primitive sub-components: static features $\boldsymbol{s}_i$ and dynamic features $\boldsymbol{U}_i$. Finally, we can formulate this problem as follows:

$$p(y_i = 1|\boldsymbol{s}_i, \boldsymbol{U}_i; \boldsymbol{\Phi}) \tag{1}$$

where $\boldsymbol{\Phi}$ denotes parameters of the model, static features $\boldsymbol{s}_i \in \mathbb{R}^{S}$ and dynamic features $\boldsymbol{U}_i \in \mathbb{R}^{B\times D}$, S represents the dimension of static features, D and B represent the numbers of total days and behaviors respectively.

3.2. Overview of the architecture

As shown in Fig. 1, we improve the vanilla Transformer architecture to adapt to the churn prediction task in three ways: an affine transformation, the bidirectional attention mechanism, and a Tree-based classification head. The overall process of the Multivariate Behavior Sequence Transformer (MBST) is described as follows.

Firstly, the dynamic feature is organized in two distinct orders after affine transformation (i.e., the original version $\boldsymbol{U} = [\boldsymbol{u}^1, \boldsymbol{u}^2, \dots, \boldsymbol{u}^B] \in \mathbb{R}^{B\times D}$ and the transpose version $\boldsymbol{V} = [\boldsymbol{v}^1, \boldsymbol{v}^2, \dots, \boldsymbol{v}^B] \in \mathbb{R}^{D\times B}$). Each vector of the original data version reflects the user's daily activities, allowing the attention mechanism to learn temporal relations. Each vector in the transposed version indicates the number of times the same activity occurs on various days, allowing the attention mechanism to learn the relations between multiple behaviors. These two versions of the dynamic feature are concatenated with classification token and added with optional position encoding before they are fed into the encoder block.

Since the transformer encoder embeds the input and outputs it in the same dimension, an additional vector, also called the CLASS token, is concatenated to the original input tensor with the same dimension as the other sequence vectors for classification. The learnable CLASS token embeddings ($\boldsymbol{u}^0_{class}$ and $\boldsymbol{v}^0_{class}$)gather information from all the sequences using multi-headed self-attention. Except for classification, CLASS token is treated the same as input sequences, and only the hidden output from the CLASS token ($\boldsymbol{u}^L_{class}$ and $\boldsymbol{v}^L_{class}$) are fed into the classification head serving as a part of the ultimate representation $\boldsymbol{r}$.

For sequential data, the location information of the sequence is particularly critical. Since Transformer architecture contains no recurrence and no convolution, we need to inject some information about the relative or absolute position of the tokens in the sequence. There are many choices to encode, and we have tried multiple ways of position encoding, including the learned version and the sinusoidal version of position encoding (Eq. (2)). Comprehensive experiments are conducted to discuss the effects of different positional coding (see Section 4.4 for more details).

$$\boldsymbol{E}^{pos}_{2i} = \sin\left(pos/10000^{2i/d}\right), \qquad \boldsymbol{E}^{pos}_{2i+1} = \cos\left(pos/10000^{2i/d}\right) \tag{2}$$

where pos is the position, d is the dimension of the input sequence vectors, and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid.

In summary, the following equation can be used to describe these two stages: class token embedding and position encoding.

$$\boldsymbol{Z}^0_u = [\boldsymbol{u}^0_{class}; \boldsymbol{u}^0_1, \boldsymbol{u}^0_2, \dots, \boldsymbol{u}^0_D] + \boldsymbol{E}^{pos}_u, \qquad \boldsymbol{E}^{pos}_u \in \mathbb{R}^{B\times(D+1)} \tag{3}$$

$$\boldsymbol{Z}^0_v = [\boldsymbol{v}^0_{class}; \boldsymbol{v}^0_1, \boldsymbol{v}^0_2, \dots, \boldsymbol{v}^0_B] + \boldsymbol{E}^{pos}_v, \qquad \boldsymbol{E}^{pos}_v \in \mathbb{R}^{D\times(B+1)} \tag{4}$$

The embed matrices $\boldsymbol{Z}^0_u$ and $\boldsymbol{Z}^0_v$ are fed to two decoupled encoder blocks after the embedding procedure as mentioned above. These two encoder modules, the Temporal Encoder (TE) and the Behavioral Encoder (BE), offer different modes of computation for modeling temporal and behavioral relational dependencies. The behavioral and temporal information in the dynamic feature will fully interact through the bidirectional attention mechanism. Moreover, the classification head vectors $\boldsymbol{u}^L_{class}$ and $\boldsymbol{v}^L_{class}$ will aggregate all valuable temporal or behavioral information, and they have a low dimensionality compared to the original features. The equation can be described as follows:

$$\boldsymbol{Z}^l_u = \mathrm{BE}(\boldsymbol{Z}^{l-1}_u), \quad \boldsymbol{Z}^l_v = \mathrm{TE}(\boldsymbol{Z}^{l-1}_v), \qquad l = 1, 2, \dots, L \tag{5}$$

where L is the encoder's maximum number of layers.

The vanilla transformer architecture contains a vital module, namely layer normalization, which directly estimates the normalization statistics from the summed inputs to the neurons within a hidden layer. The layer normalization creates an embedding in the space of spheres that conforms to a conventional Gaussian distribution. The original statistical information of the mean and variance would be lost, which is crucial for customer analysis. Such statistical information, for example, contains a customer's average usage duration. As a result, the mean and variance statistics value of the original behavioral sequence $\boldsymbol{v}$ are retained as a part of the final representation. The equation can be described as follows:

$$\boldsymbol{r} = \boldsymbol{s} \oplus \boldsymbol{u}^L_{class} \oplus \boldsymbol{v}^L_{class} \oplus \boldsymbol{\mu} \oplus \boldsymbol{\sigma}, \qquad \boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^B \tag{6}$$

where $\oplus$ is the concatenation operation, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent the mean and variance statistics value of the original behavioral sequence, respectively.

Finally, an external Tree-based classifier is attached for churn prediction instead of using the multilayer perceptron as the classification head. A pseudo-code version description (Algo. 1) is supplied to aid in the explanation of the complete procedure. Except for input and output, subscripts denoting the order of samples are removed for better understanding.

Algorithm 1 the Framework of MBST

```
Input: User static data $\boldsymbol{s}_i$ and dynamic data $\boldsymbol{U}_i$
Output: Customer churn prediction result $\hat{y}_i$ (probability of churning)
1: Compute the mean and standard deviation of each behavior sequence of dynamic data $\boldsymbol{U}_i$ as $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.
2: Feed dynamic data $\boldsymbol{U}_i$ into the affine transformation block for adjusting the scale of each behavior vector.
3: Transpose the dynamic feature matrix as $\boldsymbol{V}_i = \boldsymbol{U}^T_i$.
4: Concatenate the classification token and add the position encoding to original version of feature matrix as $\boldsymbol{Z}^0_u = [\boldsymbol{u}^0_{class}; \boldsymbol{u}^1, \boldsymbol{u}^2, \cdots, \boldsymbol{u}^D] + \boldsymbol{E}^{pos}_u$.
5: Concatenate the classification token and add the position encoding to transposed version of feature matrix as $\boldsymbol{Z}^0_v = [\boldsymbol{v}^0_{class}; \boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^B] + \boldsymbol{E}^{pos}_v$.
6: Apply multi-layer behavioral and temporal attention mechanisms to $\boldsymbol{Z}^0_u$ and $\boldsymbol{Z}^0_v$ for extracting different dependency information, respectively.
7: Aggregate all extracted feature as $\boldsymbol{r} = \boldsymbol{s} \oplus \boldsymbol{u}^L_{class} \oplus \boldsymbol{v}^L_{class} \oplus \boldsymbol{\mu} \oplus \boldsymbol{\sigma}$.
8: Feed the ultimate representation $\boldsymbol{r}$ into the Tree-based XGBoost classifier.
9: return $\hat{y}_i$
```

3.3. Learnable affine transformation

Neural networks are notoriously sensitive to scale, and layer normalization (Ba, Kiros, & Hinton, 2016) is the critical algorithm for convergence in the Transformer architecture. In our application scenario, however, the feature values representing various actions on the same day vary greatly (e.g., a user spends 500 min on the browser in a day, but the number of domains viewed is only 30). Moreover, there are behavioral features that are binary values rather than continuous values (e.g., whether the user has installed other browsers on a given day). Therefore, in the temporal encoder, if the layer normalization is applied to the behavioral features $\boldsymbol{u}$, the individual feature values will affect the mean and variance of the standardization procedure, (i.e., some features will dominate the model while some are completely ignored resulting in a large bias in the model). To make layer normalization more effective, we introduce a learnable affine transformation:

$$\mathrm{Affine}(\boldsymbol{X}) = \mathrm{Diag}(\lambda_1, \lambda_2, \dots, \lambda_n)\boldsymbol{X} + \boldsymbol{B} \tag{7}$$

where $\lambda_1, \lambda_2, \dots, \lambda_n$ and $\boldsymbol{B}$ are learnable parameters, $\boldsymbol{X}$ can be the multivariate behavioral sequence and all elements in each column of the matrix $\boldsymbol{B}$ have only one shared value.

Because some features may dominate the objective function, preventing the estimator from learning from other features as expected, the affine transformation is applied to scale the different behavioral sequences at the beginning, making layer normalization more effective.

In summary, this operation has several advantages. First, it can compensate for the lack of layer normalization. Second, as opposed to batch normalization, the operator does not depend on batch statistics. Last but not least, This module allows us to resize different behavioral features in a learnable way instead of pre-processing the data and act as an abstract feature selection process.

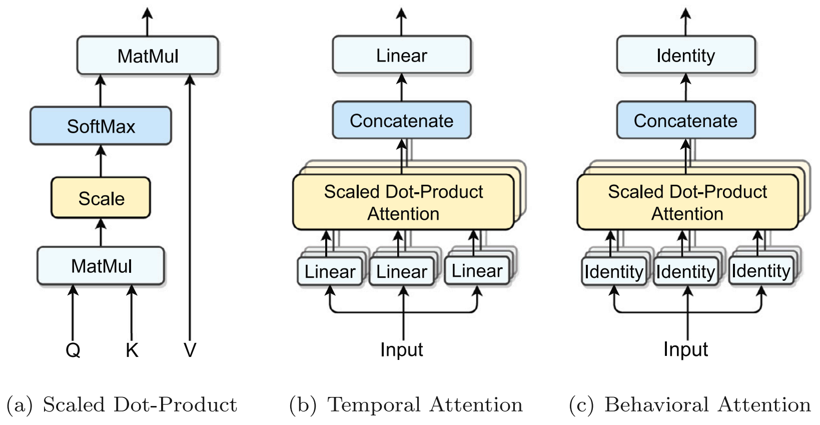

Fig. 2. Attention mechanisms.

3.4. Bidirectional attention mechanism

According to the computational paradigm of the attention mechanism, the input data consists of multiple vectors and the correlation is computed with linear mapping and dot product operation. For example, the natural language processing task's input is sequence vectors, and each vector represents a word. The relationship between words can be computed by the attention mechanism. Thus, the bidirectional attention mechanism is proposed for learning both temporal and behavioral relationships. When the original dynamic data version is fed into the encoder, each vector reflects the user's activities throughout the day, allowing the attention mechanism to learn temporal relations. Each vector in the transposed version indicates the number of times the specific action occurs on various days, allowing the attention mechanism to discover the correlation between multiple behaviors.

This work applies two decoupled but complementary encoder blocks to implement the bidirectional attention mechanism. In both of them, we use the scaled dot-product attention (Fig. 2(a)), requiring *query* $\boldsymbol{Q}$, *key* $\boldsymbol{K}$ and *value* $\bar{\boldsymbol{V}}$ of dimension d. If the *query* and the *key* are similar (i.e., high attention weight), then the corresponding *value* is relevant. We compute the dot products of the *query* with all *keys*, and the result of the attention operation is the weighted sum of the *value* $\bar{\boldsymbol{V}}$. The softmax function has incredibly small gradients because the dot products expand in magnitude as d increases. To counteract the effect, the dot products are scaled by $\frac{1}{\sqrt{d}}$:

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\bar{\boldsymbol{V}}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\bar{\boldsymbol{V}} \tag{8}$$

In the temporal encoder block (Fig. 3(a)), we divide the input sequence vector according to the time dimension, (i.e., each element of the same input sequence $(\boldsymbol{u}^1,\boldsymbol{u}^2,\dots,\boldsymbol{u}^D)$ represents different behaviors). We use a multi-head attention mechanism (Fig. 2(b)) to enable the model to attend to information from several representation subspaces simultaneously, and the temporal attention can be expressed as follows:

$$\begin{aligned}\text{TemporalAttention}(\boldsymbol{X}) = {} & \text{Concat}(\text{head}_1,\dots,\text{head}_h)\boldsymbol{W}^O \\ \text{where, head}_i = {} & \text{Attention}\left(\boldsymbol{X}\boldsymbol{W}_i^Q,\ \boldsymbol{X}\boldsymbol{W}_i^K,\boldsymbol{X}\boldsymbol{W}_i^{\bar{V}}\right)\end{aligned} \tag{9}$$

In the behavioral encoder block (Fig. 3(b)), the input sequence vectors are divided based on the behavioral dimension (i.e., each sequence represents a kind of behavior). Each element of the same input sequence $(\boldsymbol{v}^1,\boldsymbol{v}^2,\dots,\boldsymbol{v}^B)$ represents a distinct moment in time. Because they are in different semantic spaces, they cannot share the same linear mapping layer as the vanilla attention mechanism. Empirically, the dot product of the sequence itself is enough to affect the attention weight. For example, the number of domains accessed by users is often positively correlated with the users' usage time (i.e., the dot product value of such two sequence vectors can be high). Moreover, we find that increasing the number of attention heads (Fig. 2(c)) in the behavioral encoder does not improve performance. Therefore, we drop the linear mapping in attention and get the weights directly through the dot product, and the behavioral attention can be expressed as follows:

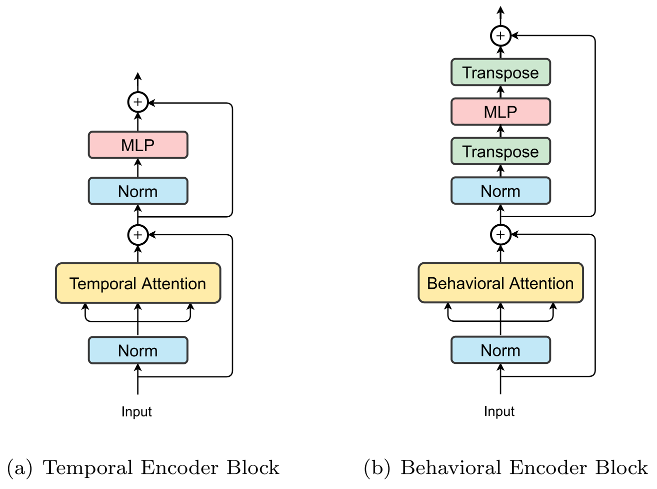

Fig. 3. Temporal and behavioral encoder blocks.

$$\text{BehavioralAttention}(\boldsymbol{X}) = \text{Softmax}\left(\frac{\boldsymbol{X}\boldsymbol{X}^T}{\sqrt{d}}\right)\boldsymbol{X} \tag{10}$$

In addition to attention sub-layers, both two categories of encoders contain a fully connected feed-forward network, which is applied to each sequence separately and identically. These consist of two linear transformations with a ReLU activation in between:

$$\text{FFN}(\boldsymbol{x}) = \text{ReLU}\left(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1\right)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{11}$$

where the linear transformations are the same across different positions. It is noteworthy that the embedding representation output by the attention module should be transposed to adapt to the sharing of linear layer parameters in the behavioral encoder.

In summary, two encoder modules, temporal and behavioral encoder, are proposed using different attention mechanisms to handle different dependencies. In the temporal encoder, the multi-headed attention mechanism in the vanilla Transformer is used to analyze the relationship between times. In the behavioral encoder, the linear mapping is abandoned, and the dot product is used directly to calculate the similarity between behavioral sequences to decrease the parameters

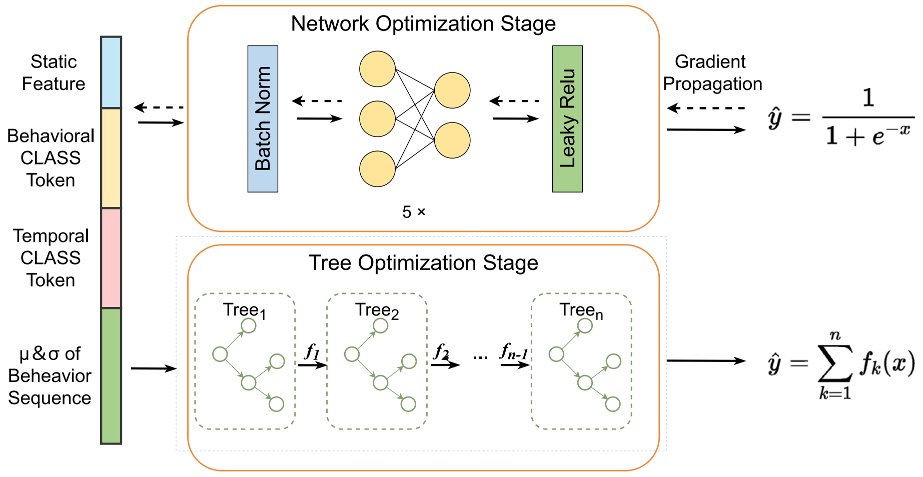

Fig. 4. Two optimization stages.

and achieve better performance. The combination of these two modules enables a bidirectional attention mechanism that can model both temporal and behavioral dependencies.

3.5. Multiple classification heads

The Tree-based methods' computational paradigm is non-differentiable. Thus they cannot update the encoders' weight through gradient propagation. The multilayer perceptron is vastly over-parametrized and not applicable for structured user data. To train the encoders and maximize the use of aggregated feature information for classification, two different classification heads (Fig. 4) are used to achieve respective goals.

For training the network, as tree methods are non-differentiable functions, we use the multilayer perceptron (MLP) to implement back-propagation which follows a standard tower pattern in which the bottom of the network is most comprehensive, and each hidden layer is followed by a batch normalization layer and LeakyReLU activation function:

$$\mathrm{LeakyReLU}(x) = \max(0, x) + s \times \min(0, x) \tag{12}$$

where s controls the angle of the negative slope. In contrast to the Rectified Linear Unit (ReLU) activation function, it assigns a non-zero slope to all negative values to retain some information.

The XGBoost algorithm is attached as the classification head to maximize the use of aggregated feature information. The gradient boosted decision tree is used as its base estimator.

There are several reasons why we use the Tree-based method instead of the multilayer perceptrons:

1. The multilayer perceptron is vastly over parametrized, and the lack of appropriate inductive bias often causes them to fail to find optimal solutions.
2. The ensemble Tree-based learning method is representationally efficient for decision manifolds with approximately hyperplane boundaries.
3. The XGBoost algorithm adds a regularization term to regulate the complexity of the model and avoid overfitting

In summary, the XGBoost algorithm is independent of the data magnitude, minimizes the loss function with fewer parameters, and thus has better generalization performance for classifying the aggregated feature. On the other hand, a multilayer perceptron is necessary for training the network since the existing optimizer cannot back-propagate the gradient for Tree-based classifiers.

3.6. Optimization

The optimization loss function for network training can be formulated as:

$$l(\hat{y}_i, y_i) = \sum -y_i \log \hat{y}_i - \left(1 - y_i\right) \log\left(1 - \hat{y}_i\right) \tag{13}$$

where y_i represents the ground-truth churn label and $\hat{y}_i$ is the predicted churn label for the user.

In order to train the tree model, L1 regular penalty and L2 regular penalty are used to control the complexity of the base classifier. The optimization loss function can be formulated as:

$$\mathcal{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right) \quad where\ \Omega(f) = \alpha T + \frac{1}{2}\lambda\|w\|^2 \tag{14}$$

where α and λ denote the coefficients of the L1 regularization penalty and the L2 regularization penalty, T denotes the number of leaf nodes in the tree, and $\|w\|^2$ is the L2 penalty of leaf scores.

4. Experiments

4.1. Dataset

Detailed experiments are conducted on the Tencent QQ dataset to prove the proposed method's effectiveness. This real-world dataset contains anonymous data from more than 600,000 users, including the user behavior data from November 29 to December 19, 2016. Following that, we will take a closer look at the QQ browser and the dataset.

QQ Browser delivers not only desktop but also mobile online browsing services. It ranks top 2 on the desktop browser and top 5 on mobile web browsers in China, with 303 million monthly active users from Jun 2018 to Jun 2019. Thus the retention of these users is the prime concern, and churn prediction enables important insights and action cues on retention.

There are two types of data in the dataset: static data and dynamic data. The static data are tags labeled according to users' browsing history in terms of interests, videos, games, etc. The interest category tag contains dozens of sub-categorical tags, as demonstrated in Table 1. The tags are assigned different weights according to the user's browsing preferences. For example, if the weights of a user's entertainment tag and education tag are 80 and 5, respectively, this user prefers entertainment to education. The dynamic data consists of multiple behavior sequences with a sampling period of one day, including usage time, the number of different domains visited, launch time of the browser and related applications. The observable time is limited to 14 days. After preprocessing, including feature selection and outlier processing,

Table 1
The description of static data.

Interest category	Tag
News	Comprehensive news Military news Financial news
...	...
Education	Comprehensive education Preschool education Vocational education

28 static features and 22 different dynamic behavior sequences are preserved, as shown in Table 2 for prediction.

In the end, it is crucial to determine whether or not a user is a churner. After the professional discussions with domain experts, we identified the following selection criteria to distinguish active users and churners to address this problem. Users who do not utilize the browser in 20 days are defined as churners, and those who use the browser for more than seven days are defined as active users. Other users who employ the browser for less than seven days are excluded. To summarize, the dataset's customer churn rate reaches 34.93%.

4.2. Experiment setup

Training details. Our experiments are all performed on a workstation with Nvidia RTX 3090 GPU. Twenty percent of samples are reserved for testing. Experiments follow default settings for some hyperparameters of all involved models, such as AdamW weight decay, AdamW β_1 parameter, etc. Grid searches are conducted for other parameters like learning rate and batch size to find the best-performing parameter within a specific range. The implementation of the MBST follows this set, in which the number of temporal and behavioral attention encoder layers is set to 3. In the encoder block, the number of multi-heads is set to 2, and the number of neurons in the fully connected layer is 32. The multilayer perceptron heads have five layers in total. Table 3 summarizes the considered default values and the meaning of each parameter. While training, we choose AdamW (Loshchilov & Hutter, 2017) as the optimizer and set the batch size to 512. We use a cosine learning rate decay strategy according to (He et al., 2019) as follows:

$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)\eta \tag{15}$$

where η_t is the learning rate at batch t, the initial learning rate η is set to 0.01 in our experiments, and T is the total number of batches.

Baseline approaches. We compare our proposed model with various Tree-based methods and sequence models to assess the performance. Tree-based models include Random Forest (RF), Adaptive Boosting (AdaBoost), Gradient Boosting Decision Tree (GBDT), and Extreme Gradient Boosting (XGBoost). Recurrent Neural Networks (RNN), Long Short-Term Memory Model (LSTM), and the vanilla Transformer represent various state-of-the-art sequence models.

Evaluation metrics. In order to evaluate the churn prediction performance of our model, we adopt four metrics described as follows:

- **Log Loss**: The Log Loss is the most crucial classification metric based on probabilities. The Log Loss represents how close the forecast probability is to the actual/true number. Minimizing the Log Loss is the same as increasing the classifier's accuracy. Eq. (13) describes the specific formula of log loss.
- **F-score**: The F-score is computed by dividing the number of true positive results by the total number of positive results, including those incorrectly recognized. The recall is the number of genuine positive findings divided by the total number of samples that should have been detected as positive. To measure performance, we utilize a balanced F-score (F1 score), the harmonic mean of accuracy and recall.

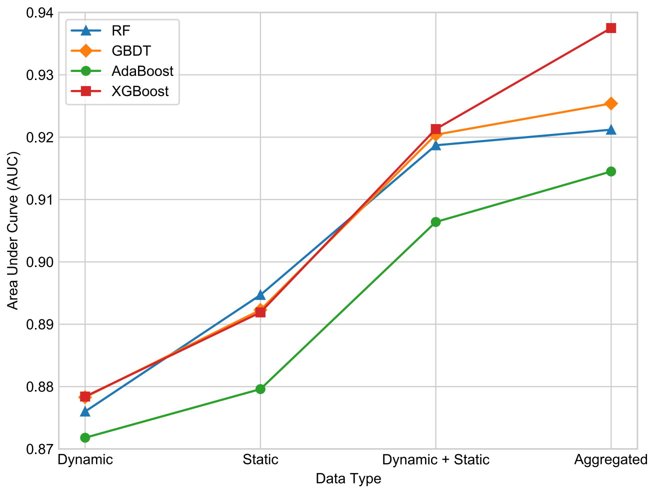

Fig. 5. Comparison results (AUC) across different data.

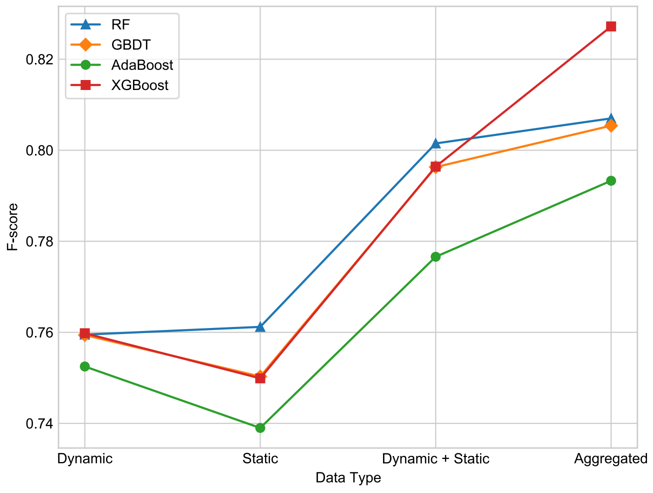

Fig. 6. Comparison results (F-score) across different data.

- **AUC**: Area Under the ROC Curve (AUC) measures the entire area underneath the receiver operating characteristic (ROC) curve, measuring how accurately the model can distinguish between two things. This score gives an overall performance metric across all potential classification thresholds.
- **P-value of Delong Test**: Delong Test (DeLong, DeLong, & Clarke-Pearson, 1988) is a nonparametric approach to analyzing areas under correlated ROC curves using the theory of generalized U-statistics to generate an estimated covariance matrix. The experiments follow the implementation by Sun and Xu (2014) to compare the ROC curves of all comparative and ablation trial models with our proposed model.

4.3. Main results

In Table 4, we compare MBST with various models and MBST substantially outperforms state-of-the-art methods in Log Loss, F-score, AUC and p-value of the Delong Test. All methods use the same dynamic and static data as input for a fair comparison. It is worth noting that tree-based models cannot directly deal with sequence data, and sequence models cannot directly deal with static data. As a result, the sequence data is flattened for tree-based methods and a MLP is

Table 2
The description of dynamic data.

Numerical data	Categorical data
Number of direct launches (QB desktop shortcuts)	Whether to start Kingsoft AntiVirus
Number of direct launches (navigation shortcuts)	Whether to start 360 Total Security
Number of direct launches (QB start menu shortcut)	Whether to start Sogou Browser
Number of launches through QQ	Whether to start 360 Extreme Browser
Number of domain names visited	Whether to start Cheetah Browser
…	…
Usage time (s)	Whether to start 360 Security Browser

Table 3
Main considered parameters and related default values.

Description	Parameter	Value
Batch size	T	512
Number of epochs	N_e	100
Learning rate	η	0.01
AdamW β_1 parameter	β_1	0.9
AdamW β_2 parameter	β_2	0.999
AdamW ϵ parameter	ϵ	10^{-8}
AdamW weight decay	ω	0.01
LeakeyReLU negative slope	s	0.01
XGBoost learning rate	η	0.3
Maximum depth of the tree	d	6
XGBoost L1 penalty	α	1
XGBoost L2 penalty	λ	1

Table 4
Comparison results across different models.

Model	Model type	Log loss	F-score	AUC	P-value
RF	Tree	0.6767	0.8015	0.9187	4.953e−15
AdaBoost	Tree	0.6764	0.7766	0.9064	6.755e−34
GBDT	Tree	0.3323	0.7963	0.9204	1.196e−18
XGBoost	Tree	0.3320	0.7964	0.9213	1.454e−25
RNN	Sequence	0.3175	0.8085	0.9253	3.699e−11
LSTM	Sequence	0.3116	0.8124	0.9278	1.277e−21
Transformer	Sequence	0.3116	0.8119	0.9278	3.009e−14
MBST	**Tree & Sequence**	**0.2904**	**0.8272**	**0.9375**	–

Table 5
Important components for MBST.

Aff.	T.E.	B.E.	Stat.	Cls.	Log Loss	F-score	AUC	P-value
✗	✓	✓	✓	MLP	0.3109	0.8142	0.9283	2.177e−4
✓	✗	✓	✓	MLP	0.3081	0.8150	0.9292	1.327e−2
✓	✓	✗	✓	MLP	0.3123	0.8145	0.9279	3.207e−3
✓	✓	✓	✗	MLP	0.3120	0.8145	0.9282	4.782e−4
✓	✓	✓	✓	MLP	0.3084	0.8165	0.9297	3.631e−2
✓	✓	✓	✓	RF	0.6733	0.8070	0.9212	4.127e−3
✓	✓	✓	✓	AdaBoost	0.6752	0.7933	0.9145	3.449e−5
✓	✓	✓	✓	GBDT	0.3193	0.8054	0.9254	2.164e−2
✓	✓	✓	✓	**XGboost**	**0.2904**	**0.8272**	**0.9375**	–

Table 6
Different version of position encoding for MBST.

Module	P.E.	Log loss	F-score	AUC
Temporal encoder	Sinusoidal	0.3107	0.8148	0.9291
	Learnable	**0.3084**	**0.8165**	**0.9297**
	None	0.3101	0.8160	0.9287
Behavioral encoder	Sinusoidal	0.3112	0.8161	0.9286
	Learnable	0.3099	0.8119	0.9286
	None	**0.3084**	**0.8165**	**0.9297**

employed to take static data input for sequence models. Experimental results show that traditional Tree-based learning algorithms have shown competitive performance. The experimental results of RF and AdaBoost on Log Loss are not good because they do not use the corresponding log-likelihood loss as the optimization objective function but classify the nodes based on information gain and use voting to decide the classification result. The advantages of the Transformer for long-term dependency modeling are not well exploited in short sequences, and the result is that the Transformer achieves almost the same performance as the LSTM. Our MBST takes advantage of both Tree-based and sequence models based on deep learning and improves the vanilla Transformer's attention mechanism to understand more relational dependencies. As a result, the proposed approach MBST achieves the Log Loss of 0.2904, the F-score of 82.72% and the Area Under Curve (AUC) of 93.75%.

In Figs. 5 and 6, we conduct comparative experiments across different data types to prove the efficiency of data and the advantages of aggregated representation $\boldsymbol{r}$ (Eq. (6)) obtained by the fusion module of MBST. We can find that the effect of using only dynamic and static data is insufficient, and the simultaneous use of dynamic and static data can significantly enhance the model performance. In addition, the aggregated feature representation embedded by MBST is better than the hybrid feature.

The receiver operating characteristic (ROC) curve of such four Tree-based models based on different data types is described in Fig. 7. It shows the robustness of proposed methods under different threshold settings by the evidence that the model outperforms the others at various thresholds based on aggregated representation obtained by the fusion module of MBST.

4.4. Ablation studies

Extensive ablation experiments are conducted to demonstrate the indispensability of each module. Table 5 demonstrates different model variants as adding or removing essential components, including Affine Transformation (Aff.), Temporal Encoder (T.E.), Behavioral Encoder (B.E.), efficient statistic (Stat.). The impact of the behavioral encoder on the model performance has exceeded the temporal encoder of the vanilla Transformer, which also proves the effectiveness of our proposed component. Experiments with other Tree-based methods are conducted to demonstrate the importance of the classification heads (Cls.). The experimental results show that the classifier used in MBST stands out among all tree classifiers and that the aggregation feature improves the performance of any integrated tree classifiers.

In Table 6, we test the impact of different position encoding. We add the sinusoidal version (Eq. (2)) and the learned version of position encoding to the temporal and behavioral encoder and test the performance, respectively. The empirical results show that learnable encoding is optimal for the temporal encoder, while adding any position encoding will reduce the performance of the behavioral encoder. The analysis is as follows: in processing temporal sequences, the model needs to learn fewer parameters because the input length is short and fixed, so the self-learning method is better than the sinusoidal version method; in processing behavior sequences, the sequences are divided according to different behaviors. Therefore, unlike the temporal sequences, which are strictly sequential, the location relationships between the behavioral sequences are not as important.

In Fig. 8, we explore the effect of the multiple heads number on the effect. In the temporal encoder, it can be observed that the model improves in both AUC and F-score when the number of attention heads increases, and the performance is best when the number of

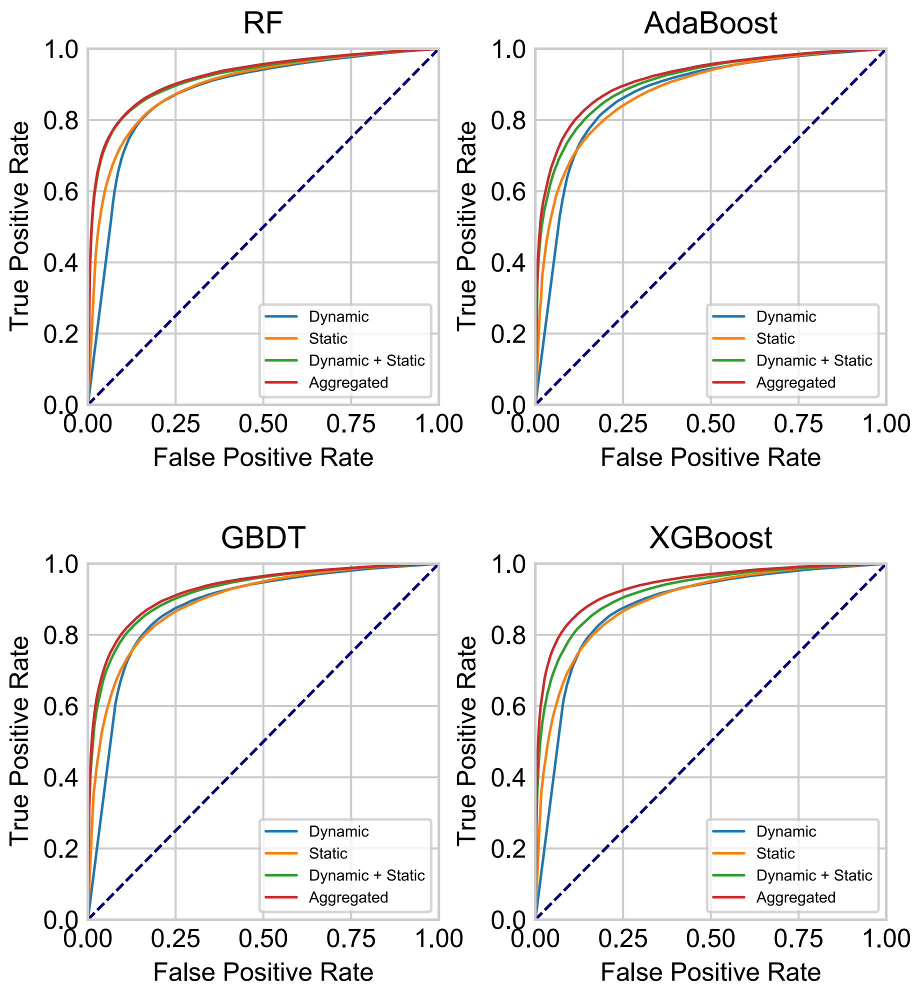

Fig. 7. The receiver operating characteristic (ROC) curve.

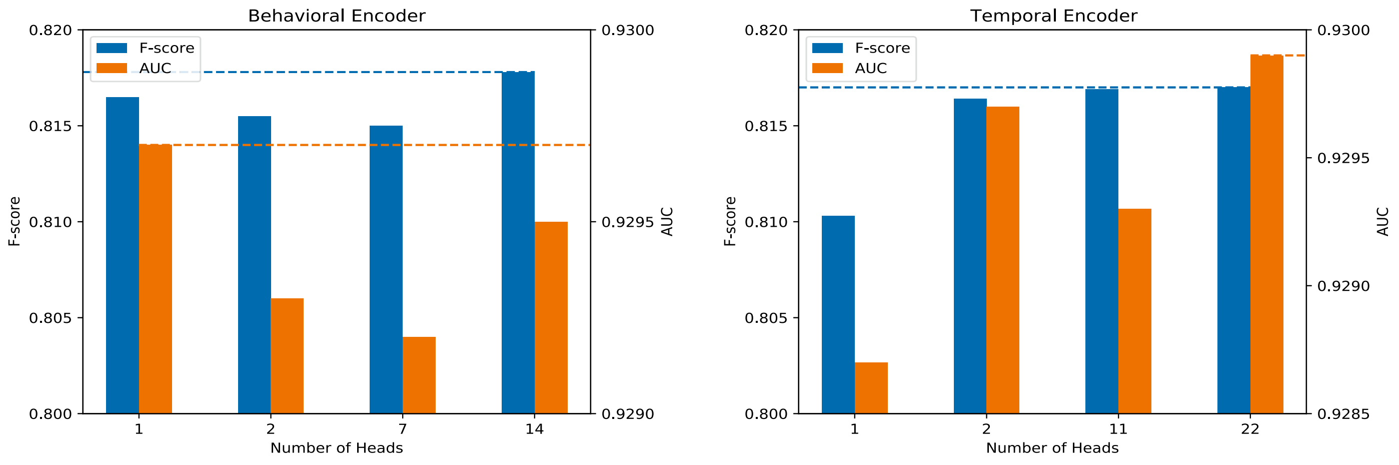

Fig. 8. The effect of heads number.

attention heads reaches the maximum (i.e., equal to the dimensionality of a single time series), which illustrates the heterogeneity between different behavioral relationships in a single time series. While in the behavioral encoder, we observe an objective pattern, where the model's performance decreases as the number of attention heads increases. Although the performance on a metric is excellent when the number of attention heads is maximized, more heads tend to mean multiply computation, so we use attention heads of 2 and 1 for the temporal and behavioral encoders, respectively, in our practical experiments.

4.5. Visualization and discussion

We compute the attention weights obtained by each encoder layer and present them in a heat map after average pooling. The attention weight distribution of the behavioral and temporal encoders is shown in Figs. 9 and 10 respectively. The first row (column) represents a class token sequence in the heat map, and the other rows (columns) represent behavior sequences or time sequences. The attention weight varies with the brightness of the color in the cell. In the heat map of the behavioral

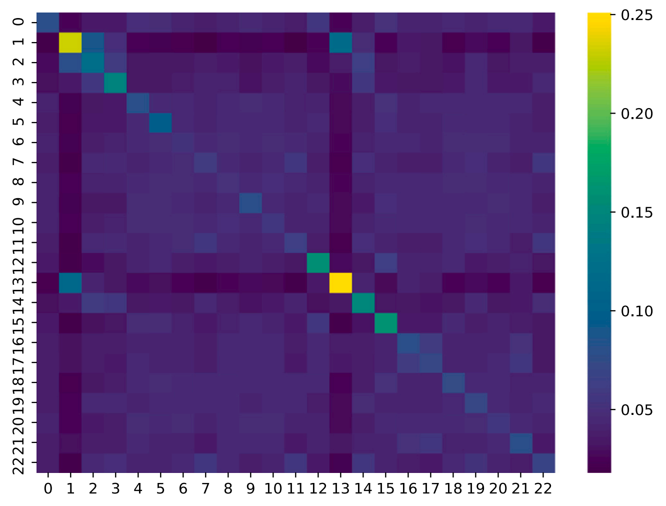

Fig. 9. Distribution of attention weights in the behavioral encoder.

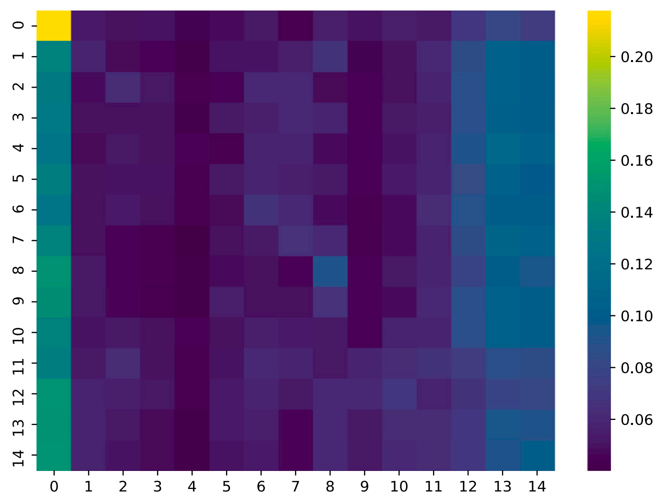

Fig. 10. Distribution of attention weights in the temporal encoder.

encoder, the attention weight matrix is symmetric because we drop the linear mapper. We find that the correlation between the second 2nd and the 14th sequence is high, representing the number of websites visited and the total use time in one day, respectively. In the heat map of the temporal encoder, counter-intuitively, the value of the attention weight matrix on the diagonal is not high. Each sequence is highly correlated with the sequence of the last three days. Intuitively, if a user has no record in the last few days, it is highly likely to become a churner. We know from the heatmap that the last days of data are given more attention, which indicates that the model captures this pattern well.

5. Conclusion and future work

This paper introduces the Multivariate Behavior Sequence Transformer, a new churn prediction model (MBST). The model enhances the prediction by two orthogonal but complementary attention mechanisms and combines the Tree-based learning method and sequence model. A comparison to state-of-the-art methods demonstrates the superiority of our methodology. In addition, extensive ablation experiments are carried out to demonstrate each component's indispensability. The depiction of attention weights aids the model's interpretability.

From the data perspective, MBST can combine static and dynamic data to obtain a comprehensive representation and improve the performance of downstream tasks. Data is the soul of both customer management and machine learning, and introducing more data from managerial insights is a key way to improve prediction performance.

From the algorithmic perspective, MBST draws on both the learning capability of the tree ensemble model for structured data and the learning capability of the sequence model for time series. However, it is not yet a universal model so far. The attention mechanism is highly customizable to the extent that its application is limited. Furthermore, because the MBST focus on the classification problem of short time series, the model cannot perform well when dealing with large time series due to the squared computational complexity of the attention mechanism, long time step input memory bottleneck, etc. As a result, we aim to make MBST applicable to more scenarios in the future.

CRediT authorship contribution statement

Xing Wu: Conceptualization, Methodology, Funding acquisition, Project administration, Supervision, Writing – review & editing. **Pan Li:** Writing – original draft, Data curation, Software, Implementation, Investigation, Formal analysis, Visualization. **Ming Zhao:** Supervision, Writing – review & editing. **Ying Liu:** Supervision, Writing – review & editing. **Rubén González Crespo:** Supervision, Writing – review & editing. **Enrique Herrera-Viedma:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62172267), the National Key R&D Program of China (Grant No. 2019YFE0190500), the Natural Science Foundation of Shanghai, China (Grant No. 20ZR1420400), the State Key Program of National Natural Science Foundation of China (Grant No. 61936001).

References

Ahmad, A. K., Jafar, A., & Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*, *6*(1), 1–24.
Ahn, J., Hwang, J., Kim, D., Choi, H., & Kang, S. (2020). A survey on churn analysis in various business domains. *IEEE Access*, *8*, 220816–220839.
Aksan, E., Cao, P., Kaufmann, M., & Hilliges, O. (2020). A spatio-temporal transformer for 3D human motion prediction. arXiv preprint arXiv:2004.08692.
Alboukaey, N., Joukhadar, A., & Ghneim, N. (2020). Dynamic behavior based churn prediction in mobile telecom. *Expert Systems with Applications*, *162*, Article 113779.
Amin, A., Shah, B., Khattak, A. M., Moreira, F. J. L., Ali, G., Rocha, A., et al. (2019). Cross-company customer churn prediction in telecommunication: a comparison of data transformation methods. *International Journal of Information Management*, *46*, 304–319.
Arivazhagan, B., & Sankara, S. (2020). Customer churn prediction model using regression with Bayesian boosting technique in data mining. *Ijaema. Com*, *12*(V), 1096–1103.
Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.
Backiel, A., Verbinnen, Y., Baesens, B., & Claeskens, G. (2015). Combining local and social network classifiers to improve churn prediction. In *Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015* (pp. 651–658). ACM.
Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*(2), 197–227.
Bilal Zorić, A. (2016). Predicting customer churn in banking industry using neural networks. *Interdisciplinary Description of Complex Systems: INDECS*, *14*(2), 116–124.
Chen, K., Chen, G., Xu, D., Zhang, L., Huang, Y., & Knoll, A. (2021). NAST: Non-autoregressive spatial-temporal transformer for time series forecasting. arXiv preprint arXiv:2102.05624.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

De Caigny, A., Coussement, K., & De Bock, K. W. (2018). A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *European Journal of Operational Research*, *269*(2), 760–772.

Dela Llave, M. A., López, F. A., & Angulo, A. (2019). The impact of geographical factors on churn prediction: an application to an insurance company in madrid's urban area. *Scandinavian Actuarial Journal*, *2019*(3), 188–203.

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 837–845.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Farquad, M. A. H., Ravi, V., & Raju, S. B. (2014). Churn prediction using comprehensible support vector machine: An analytical CRM application. *Applied Soft Computing*, *19*, 31–40.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml, vol. 96* (pp. 148–156). Citeseer.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 1189–1232.

Grabczewski, K., & Jankowski, N. (2005). Feature selection with decision tree criterion. In *Fifth international conference on hybrid intelligent systems (HIS'05)* (pp. 6–pp). IEEE.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 558–567).

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

Jamalian, E., & Foukerdi, R. (2018). A hybrid data mining method for customer churn prediction. *Engineering, Technology & Applied Science Research*, *8*(3), 2991–2997.

Keramati, A., Ghaneei, H., & Mirmohammadi, S. M. (2016). Developing a prediction model for customer churn from electronic banking services using data mining. *Financial Innovation*, *2*(1), 10.

Kristensen, J. T., & Burelli, P. (2019). Combining sequential and aggregated data for churn prediction in casual freemium games. In *2019 IEEE conference on games (CoG)* (pp. 1–8). IEEE.

Leung, H. C., & Chung, W. (2020). A dynamic classification approach to churn prediction in banking industry.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030.

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Lu, N., Lin, H., Lu, J., & Zhang, G. (2012). A customer churn prediction model in telecom industry using boosting. *IEEE Transactions on Industrial Informatics*, *10*(2), 1659–1665.

Mayr, A., Binder, H., Gefeller, O., & Schmid, M. (2014). The evolution of boosting algorithms. *Methods of Information in Medicine*, *53*(06), 419–427.

Medsker, L. R., & Jain, L. (2001). Recurrent neural networks. *Design and Applications*, *5*, 64–67.

Milošević, M., Živić, N., & Andjelković, I. (2017). Early churn prediction with personalized targeting in mobile social games. *Expert Systems with Applications*, *83*, 326–332.

Nie, G., Rowe, W., Zhang, L., Tian, Y., & Shi, Y. (2011). Credit card churn forecasting by logistic regression and decision tree. *Expert Systems with Applications*, *38*(12), 15273–15285.

Óskarsdóttir, M., Bravo, C., Verbeke, W., Sarraute, C., Baesens, B., & Vanthienen, J. (2017). Social network analytics for churn prediction in telco: Model building, evaluation and network architecture. *Expert Systems with Applications*, *85*, 204–220.

Óskarsdóttir, M., Van Calster, T., Baesens, B., Lemahieu, W., & Vanthienen, J. (2018). Time series for early churn detection: Using similarity based classification for dynamic networks. *Expert Systems with Applications*, *106*, 55–65.

Periáñez, A., Saas, A., Guitart, A., & Magne, C. (2016). Churn prediction in mobile social games: Towards a complete assessment using survival ensembles. In *2016 IEEE international conference on data science and advanced analytics (DSAA)* (pp. 564–573). IEEE.

Plizzari, C., Cannici, M., & Matteucci, M. (2021). Spatial temporal transformer network for skeleton-based action recognition. In *Pattern recognition. ICPR international workshops and challenges: virtual event, january 10–15, 2021, proceedings, part III* (pp. 694–701). Springer.

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *Ieee Assp Magazine*, *3*(1), 4–16.

Rajamohamed, R., & Manokaran, J. (2018). Improved credit card churn prediction based on rough clustering and supervised learning techniques. *Cluster Computing*, *21*(1), 65–77.

Sun, X., & Xu, W. (2014). Fast implementation of delong's algorithm for comparing the areas under correlated receiver operating characteristic curves. *IEEE Signal Processing Letters*, *21*(11), 1389–1393.

Sundarkumar, G. G., & Ravi, V. (2015). A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance. *Engineering Applications of Artificial Intelligence*, *37*, 368–377.

Tamassia, M., Raffe, W., Sifa, R., Drachen, A., Zambetta, F., & Hitchens, M. (2016). Predicting player churn in destiny: A hidden markov models approach to predicting player departure in a major online game. In *2016 IEEE conference on computational intelligence and games (CIG)* (pp. 1–8). IEEE.

Tan, F., Wei, Z., He, J., Wu, X., Peng, B., Liu, H., et al. (2018). A blended deep learning approach for predicting user intended actions. In *2018 IEEE international conference on data mining (ICDM)* (pp. 487–496). IEEE.

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model.. *Artificial Intelligence Review*, *53*(8), 5929–5955.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., et al. (2020). Spatial-temporal transformer networks for traffic flow forecasting. arXiv preprint arXiv:2001.02908.

Yang, C., Shi, X., Jie, L., & Han, J. (2018). I know you'll be back: Interpretable new user clustering and churn prediction on a mobile social application. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 914–922).

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*.