Contents

Introduction

The work is dedicated to the accounting of external information in the models for event sequence representations. Sequences of events are a widely used data type that represents an irregular multidimensional time series. Compared with data in CV and NLP tasks, this domain is not as widespread in research papers but can still describe many application areas, for example, bank transactions data or purchases in stores. One of the main tasks for such data is the constructing of representation vectors or embeddings, which can describe sequence at the current time point and can be used further in other applied tasks. Different approaches are used for embedding construction but all of them do not take into account external context. So the main goal of this work is to develop and test different types of external information addition in the construction of event sequence representations.

Relevance

The task of constructing representation vectors for event sequences data is important in many applied areas, including in the field of banking transactions. In this work, we focus on this domain because, firstly, it contains a large amount of data and, secondly, it represents an important for industry tasks, including credit scoring and churn prediction.

Any bank keeps a lot of information about the client's transactions. Each client is represented by its sequence of transactions. It is natural to assume that this big sequential data contains useful for business but hidden information about users. The availability of a large set of such data makes it possible to train various machine learning models [5]. Modern methods especially focus on representation learning models including representations of transactional sequences [2, 24, 27]. The learned embeddings are used later as input in various applied problems [2, 36].

The most relevant for banking sector tasks are churn prediction and credit scoring. Churn prediction includes identifying at-risk users who are likely to stop their work with a particular bank or close their accounts. This problem is important for the bank as it makes it possible to predict the number of users and help to select the way of communication with each particular client. Credit scoring refers to the process of monitoring customer behavior to find out will a particular user pay a loan or not. This problem is one of the most important for banks as its solution helps to detect and prevent spurious actions. Both of these tasks can be solved using only sequences of users transactions.

Although this work examines problems from the banking sector, the proposed approach can be generalized to other areas of data that can be represented as a sequence of events. For example, one can consider the domain of purchasing goods in a store, user interaction on social networks, or the impact of certain events in life on a person.

Main purpose of the research

As it was said before the main purpose of this work is to improve models for the construction of representations of event sequences in general and for sequences of bank clients transactions in particular. Event sequences data is complex data. Each sequence is unique in the set of all sequences and it is important to be able to distinguish two sequences. Also, each sequence itself develops and changes through time. For example, a client in the bank can change its behavior if she decides to leave the bank. It is a significant task to be able to find these changes in sequences using sequence embeddings. So, representations of event sequences have two important properties:

1. Global properties — characterize the sequence as a whole, at the level of the entire set of all sequences.
2. Local properties — are responsible for describing the state of a specific sequence at a specific point in time.

Existing approaches for constructing event sequence representations usually consider only one group of properties, either local or global information. Thus, methods based on the contrastive approach for constructing a representation consider the whole sequence of events [2, 24]. In this approach, the importance of local changes in sequences is blurred and additional techniques are required to overcome this problem. For example, in the work [42] hierarchical loss function is used. On the other hand, the autoregressive approach focuses on the local properties of sequences as it tries to predict the next event and its time. Simultaneously this method does not distinguish different sequences and works equally for each sequence using only events prehistory.

Even though existing models have some global properties, they do not take into account the external context when forming a representation of event sequences data. For example, in the domain of transactional data models don't consider representations of other bank clients or macroeconomic parameters [4]. Accounting for this information is a new task that will potentially improve the quality of the models [1].

Thus, the purpose of this work is to improve the local and global properties of transaction data representations for a bank user using accounting of external information.

Since the external context influences the behavior of sequences, our main assumption is that the external context can be identified through some aggregation of all representation vectors of sequences at a given time point. The addition of this aggregation into representation models can improve the quality of the global and local applied tasks.

For the transactional data proposed approaches will improve the process of creating machine learning models based on this data in the bank, improve the quality of the models, and, consequently, the quality of decisions that are made based on forecasts obtained using such models. Improving local and global properties of representations of sequences of transactions will increase metrics of churn detection and credit scoring tasks.

Scientific novelty

For now, there are two main state-of-the-art methods for constricting embeddings for event sequences. The first one is a contrastive approach [2, 24], in which representations of subsequences from one sequence are getting closer and different sequences are getting further from each other. The second one is the autoregressive approach, in which the model tries to predict the next event and its time using the obtained representation. Naturally, the first approach has better global properties, and the second one has better local properties. But, of course, both of these approaches build an embedding using only one sequence and without considering the interaction of the sequences with each other and therefore the overall context.

On the other side, we propose to aggregate different sequences at the given time point, extract the external context vector for the current moment, and add this vector to the sequence representation. Our solution, with the addition of an external context vector, is built upon two approaches described above. We provide different methods of context aggregation and analyze their performance on applied tasks. This is a completely new approach to improving embedding for sequential data, which has not been previously considered in other works.

Statements for defense

Summarizing all the above, we can deduce the following statements, that we test in this work:

1. The accounting of the external information can improve models for constructing representation vectors for event sequences;

2. The external context representation can be obtained from the aggregation of representations of all sequences at the current time point;

3. The representations obtained using the addition of external context vectors give better quality in applied problems. In particular, we observe that for churn and credit scoring tasks for bank transaction event sequences, the model quality improves significantly.

The purpose of the work and given statements for defense define the main problems and tasks of the project:

1. Development and research of machine learning models to obtain representations of transactional data (in particular contrastive and autoregressive approach);

2. Study of the possibility of improvement for the obtained representations by using external context;

3. Testing the received embeddings to solve several applied problems (in particular in tasks of churn and credit scoring in the bank transactions domain).

To solve these problems, we use existing methods for transactional data, particularly CoLES and autoregressive models, add account for external context, and validate all received representations for several applied problems.

Literature review

This section is dedicated to reviewing the existing methods of learning event sequence representations which possess the desired properties: efficiency when solving both global and local tasks, and external information awareness.

Event sequence representation learning

Neural networks have been shown to perform well when faced with the task of event sequence representation learning, both in local and global problems [2]. Local representations are used for tasks, constrained to one event sequence, such as next event prediction [45] or change point detection [14, 1]. On the other hand, global representations may be applied to compare and classify whole sequences [5, 21].

Neural network solutions are known to be very sensitive to the domain they are applied in, so it is crucial to keep in mind the distinctive features of our domain. Firstly, the data we work with is both temporally and spatially correlated, so preprocessing should be done with care [1]. Secondly, transactional data sequences have some key differences with simple time series:

1. time series usually have a uniform distance between events, which is not true for event series;
2. in contrast to the sequences in a time series dataset, the sequences in an event type dataset don't all necessarily have the same length;
3. transactional data has both categorical features (MCC codes), and continuous features (amount).

Self Supervised Learning Paradigm

Self-supervised learning (SSL) is one of the most promising methods for obtaining a representation of data from various types. It is based on the fact that the marking of similar and dissimilar objects can be obtained directly from data, including sequential ones, without using external markup or involving experts. This allows researchers to skip the costly process of gathering expert annotations, leveraging the large bodies of low-cost raw data. There are two main groups of methods for this approach: contrastive and generative [44].

Contrastive models

Contrastive methods all seek to achieve the following objective: a perfect model would yield close representations for ”similar” objects, and distant representations for ”dissimilar” objects. Similarity measure derived naturally from the domain of the task. For example, in our case, a pair of slices of the same sequence may be marked similar, and any other pair – dissimilar [2]. The way to measure similarity can be very simple, such as the one described in the papers [2, 36, 30], which contrasts between slices from different sequences. More sophisticated options include combining distance and angle metrics [25], as well as learning with hierarchical losses [42].

Contrastive methods first appeared in the field of computer vision: siamese models with triplet loss [20], SimCLR [11], and later DINO [10], BarlowTwins [43] are all methods, capable of learning high-quality image representations.

These methods are also effective when applied to multidimensional event sequences and time series. In [24], the authors have combined neural networks with the Nystrom kernel method and achieved competitive quality. Note, that [24] views different timestamps as separate objects in terms of contrastive learning, which may improve the representations’ generalization ability. Contrastive approaches are also gaining popularity in the field of financial transactions [2, 25].

CoLES method The paper [2] proposes the CoLES method, which uses sequence slices as the contrasted objects. This paper also demonstrated, how such representations can be successfully applied to the tasks of users’ gender classification, age regression, fraud detection, and others. As the CoLES belongs to the class of contrastive approaches [26], it requires sets of positive and negative pairs for training. The authors of the method studied various ways to obtain such pairs. Eventually, they came up with the split strategy that considers two transaction subsequences from the same client to be a positive pair and from different clients — to be a negative one. This strategy does not disrupt the transactions’ order, thereby preserving their temporal structure. Representations of the subsequences were obtained via a Long Short-Term Memory network [19] (LSTM) as an encoder model, which was trained using a contrastive loss [17].

On the other hand, CoLES and some of the other mentioned before models have a bunch of drawbacks. Firstly, CoLES, trained on whole sequences, performs poorly on smaller sequence slices. Secondly, it cannot be used to track the changes in the behavior of a single user: all slices from a common sequence will have close representations. Besides that, it could be argued that similar subsequences need to have close representations, even if they come from different users, which is definitely not the case for CoLES. All in all, CoLES representations show good performance when faced with global, inter-user tasks, but local tasks require a different training procedure.

Generative models

Generative models use different learning approaches to learn the distribution of hidden data. The knowledge gained by them can then be used to generate plausible data. These approaches often originate from neural language processing (NLP) [23, 34].

Autoencoder is a very popular generative model due to its efficiency and simplicity in different domains [39]. The paper [27] learns bank clients representations useful for downstream tasks using variational autoencoder. Masked language models take the autoencoder idea one step further. Such models recover randomly changed tokens and perform well on various benchmarks [23, 18].

According to the latest research, generative methods outperform contrastive ones at missing value prediction [21]. This fact may indicate better local properties. On the other hand, while generative models are supposedly good at the local level, they are known to be less effective at solving global problems. Since there is no obvious leader, we explore both approaches to transactional data, testing the resulting embeddings for local and global properties.

Autoregressive approach In this work we consider autoregressive (AR) modeling which is extensively used in CV [40, 12, 35, 15], as well as in sequential domains like NLP [33, 9, 6] and Audio [31, 7]. Such models are trained to predict the next item in a sequence.

Unlike token embeddings in autoencoders, autoregressive models do not use information about future tokens. Because of this feature autoregressive models are able to capture more complex patterns. It's confirmed by their superior performance on text generation tasks [16].

Accounting for external information

All the mentioned above methods consider only one sequence at a time. It is sometimes beneficial to consider the overall context, formed from the actions of every client [2, 1].

Global context may consist of the behavior of other specific clients, or it may reflect the current macroeconomic state. It has been shown, that the two strongly correlate [38, 4]. This implies that the global context may contain information, useful for model training.

Since choosing macroeconomic indicators is difficult without the proper education, and thus requires bringing in an expert in the field, it makes more sense to try different ways to aggregate the actions of all bank clients (mean, max, etc.). This approach allows extraction of more data from the dataset, without additional annotation.

Sequence embedding evaluation

The process of evaluation and comparison of the proposed approaches is a very important part of research, as it allows the researcher to test the applicability of the considered methods. For our task, we require a very specific set of benchmarks, which would test the local and global properties of sequence representations.

To evaluate the global properties, we shall use the approach from [2]. The subsequences are sampled from the parent sequence in such a way, that each interval corresponds to a single label. This makes it simple to pose a classification problem on the resulting subsequences. The ability to solve such a problem is postulated as indicative of the global properties of used representations.

For testing the local properties, we shall turn to the classic task of next token prediction, which comes from the field of temporal time processes [37]. Additionally, we consider classification task like in the evaluation of global properties but with distributed over time. Details can be found in the methods section.

Conclusions

The field of sequence representation learning has seen a multitude of different results over recent years, which indicates the importance of this field in modern applications. The literature on this subject proposes a multitude of ideas in different domains, including the domain of transactional data

Despite the large amount of work already accomplished in this field, there is a definite lack of models, which perform well both on global and local tasks. Moreover, there still is no common validation procedure for transactional data, which jointly validates the local and global properties of these models. This work aims to solve this novel task and rank the existing approaches accordingly.

Another important contribution of this work lies in taking external information into account when building representations.

The results of this work will help improve the modern solutions of real-world problems, by incorporating the ideas this work contributes and further deepening our understanding of transaction sequence representations.

Methodology

This section describes the methods used to solve problems in the project. We begin this part with a description of specific models from the baseline approach. Next, we describe the approaches used to take into account external representations.

Methods for constructing representations of transactional data

We begin this part with a description of specific models from the baseline approach: contrastive method CoLES and autoregressive method.

Contrastive models for transaction sequences

As contrasting methods for self-supervised learning in this work, we chose the modern approach CoLES, which shows good results working with event sequence data.

CoLES method for transaction sequences

CoLES was used as a starting point in the study of methods for obtaining representations of transactional data. It is a contrastive approach to self-supervised learning, proposed in the work [2]. CoLES shows high quality on a number of tasks of event sequence processing, including transactional data. A comprehensive description of this method can be found in the original article. An overview of this approach can be found below. We mainly focus on its features that can be important in the study dedicated to the local and global properties of event sequence representations.

CoLES (Contrastive Learning for Event Sequences with Self-Supervision) is an approach for obtaining representations of event sequences, including bank customer transactions, without using target data. This method works in the Self-Supervised Learning paradigm and belongs to the class of contrastive approaches [26].

In the case of CoLES, subsets of one bank client's transactions are used as positive pairs, and subsets of transactions obtained from different clients are used as negative pairs. A parametric encoder model is used to obtain representations of these subsequences. This encoder is trained using a special contractive loss function:

$$L_{km} = I_{k=m}\frac{1}{2}d(h^k, h^l)^2 + (1-I_{k=m})\frac{1}{2}max\{0, \rho-d(h^k, h^l)\}^2, \tag{1}$$

where h^k and h^l are the embeddings from k-th and l-th subsequences correspondingly, $I_{k=m}$ is the identifier whether k-th and l-th subsequences come from one sequence, $d(\cdot, \cdot)$ is a distance between embeddings and ρ is a hyperparameter.

The authors of the method provide various ways to obtain positive and negative pairs. However, they stop at the option in which two transaction subsequences of random size obtained from the users' history are used to obtain a positive pair. In this case, transactions are not mixed in terms of time and their temporary structure is preserved. Negative pairs are made up of the same subsequences but from different clients. In this work we use the same approach.

CoLES model structure

The CoLES concept allows to use different encoders to obtain transaction representations. This opportunity is supported by the library pytorch-lifestream, which we used to implement the methods in the current project. In this package, the model structure includes a number of common elements, described below: feature space, transaction encoder, and sequential data processing model.

Feature space for constructing representations. In the original work to study the quality of representations from CoLES, the authors considered different sets of features for different datasets. In this project, we standardize the approach and focus on two main characteristics that are present in all datasets: the MCC code of the transaction and its volume in terms of money — Amount. This solution will not only make it possible to more accurately compare different models, but also unify the process of testing the resulting representations.

Transaction encoder (TrxEncoder). Before a sequence of transactions is put into a parametric model, it is processed using a special transaction encoder. The basic procedure involves obtaining Word2Vec-style representations of MCC codes of some fixed dimension d_{mcc}, where each transaction type is associated with a [32] vector. Simple preprocessing of numerical characteristics is also performed, for example, normalization of the Amount.

Thus, TrxEncoder produces a sequence of dimension $(T, d_{\text{mcc}} + N_{\text{num}})$, where N_{num} is the number of numerical features ($N_{\text{num}} = 1$ in the standard case), and T is the length of the transactions sequence.

Model of sequential data processing (SeqEncoder). The resulting sequence of encoded transactions is fed to the main model, which determines the structure of the considered representations. Any neural network that operates in Sequence-to-Sequence mode can be used as a SeqEncoder. In other words, when the model receives a certain sequence of observations as input, it produces a sequence of representations of the same length. Originally the article proposed to use recurrent neural networks with Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) architecture depending on the given dataset.

Limitations of the original approach

As already mentioned, the limitations of CoLES are strongly connected to the method, which is used to obtain positive and negative pairs of subsequences. This algorithm explicitly encourages the model to ensure that the resulting representations show information about the user as a whole, rather than about his local state at some time point.

In addition, in the article, the authors use the classification of users as the main task to show the quality of their model. This task also shows only global, but not local properties of representations.

This work aims to study not only global but also local properties of transactional data representations. So, the main goal of the proposed improvements to CoLES and other proposed models is to overcome this limitation

Implementation details

In this research, we generally follow the pipeline of the pytorch-lifestream Python package, which contains the implementation of the CoLES model.

CoLES is a one-layer recurrent neural network. We use LSTM with a hidden layer dimension of 1024 for the Churn dataset . For Default, we select GRU with a hidden size of 800. The models were trained for 60 epochs with a batch size of 128. Note that the model takes two features as input: MCCs and amounts of transactions. As MCC is a categorical variable, we use its embedding of size 24 (for Churn) or 16 (for Default).

Autoregressive models

Autoregressive (AR) models predict the next item in a sequence. A recurrent neural network similar to the CoLES model is implemented in our work. The model's last hidden state is used to represent the sequence because it contains complete knowledge about the whole sequence. The model is trained to predict the next transaction, encompassing the MCC code and transaction amount.

Transaction preprocessing and the loss function

To train the AR model we add two linear heads to the recurrent neural network output for amounts and MCC prediction. Since transactions contain both categorical (MCC codes) and continuous (amounts) information, the reconstruction loss is also divided into two parts.

We use cross-entropy for the categorical features and mean squared error for the continuous features. The final loss function is a weighted sum of the two intermediate ones, with the weights being hyperparameters.

Amount preprocessing We found that the preprocessing of transaction amounts defined below is essential for the model to train successfully:

$$f(a) = \text{sign}(a) \ln (1 + a). \tag{2}$$

This transformation allows stabilization of the loss functions for amounts and MCCs without using dramatically different weights for them. Also, it is closer to the human perception. People tend to focus on the order of magnitude, ignoring the exact value.

MCC preprocessing optimal complexity for the training objective is very important for self-supervised approaches [18, 23]. In our case, we found out that prediction of rare MCC code is too complicated and unnecessary task. So, we reduce the number of unique MCC codes to 100 for all datasets. For that we clip all less frequent MCC codes.

Implementation details

The implementation of recurrent neural networks and embedding networks for input features is the same as in the CoLES model. Training details such as the number of epochs and batch size are also the same.

The weights for the amount and MCC part in combined loss are set to five and one respectively.

Usage of external information based on local representations of transactional data

Accounting of external information is a separate important task of this project. We decided to use other users' representations to add context information to the local embeddings of the considered client. In this section, we describe in detail the methodology for solving this problem.

General pipeline for aggregation of external information

An additional context representation vector (we also call it global embedding) is built from the local representation vectors from all or some selected users by aggregating them in various ways.

The global embedding model is built on top of the transaction sequence encoder. Both a pre-trained encoder from the CoLES model and encoders from the autoregressive model are discussed in this work. However, the approach can be extended to the other types of encoders.

The procedure for constructing a context vector at a certain point in time is presented in Figure 1 and is described as follows:

1. We collect a sample of all possible local customer representations: for all users and for each unique moment of the transaction.
2. All local representations from the dataset close to the current time point, but before it are selected.
3. Aggregation is applied to the resulting set of vectors. The resulting vector is the vector of the contextual global representation.

As mentioned earlier, external context vector accounting can improve the quality of models in applied problems. To check this, we concatenate the resulting context embedding vector with the user's local embedding and validate this extended representation.

Classical methods of context vector aggregation

Averaging and maximization were used as classical aggregation methods. In the first case, the vector of global representation is obtained by componentwise averaging of the local representation vectors for all users in the stored dataset. In the second case, by taking the maximum value for each component. For these methods, we can draw an analogy to the Mean and Max Pooling operations in

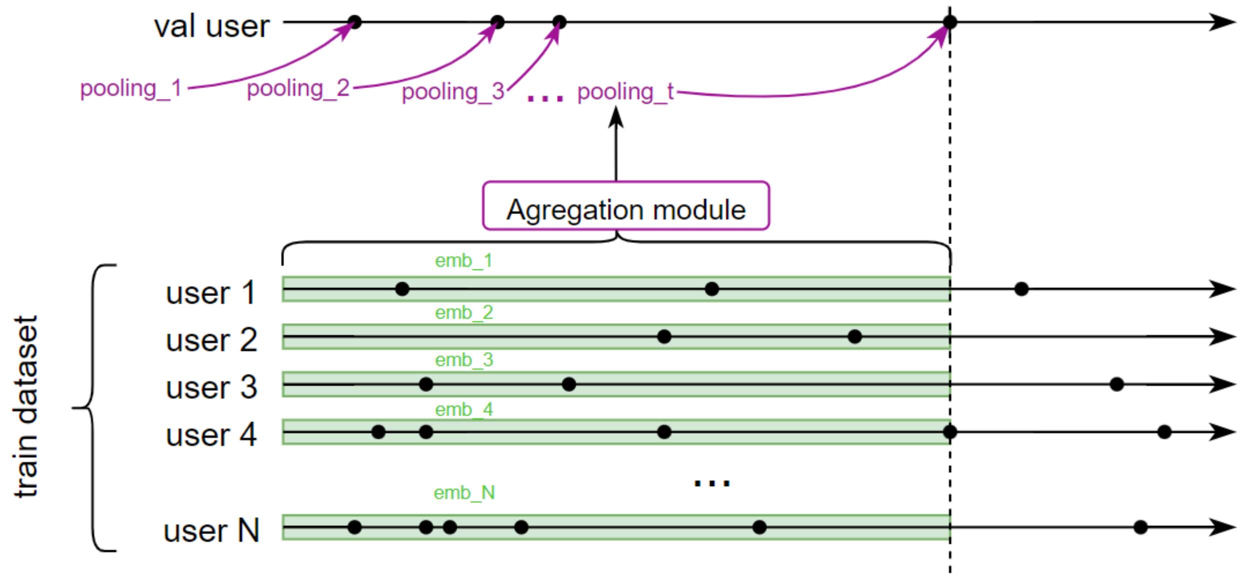

Figure 1: General pipeline for obtaining global representation vectors.

convolutional neural networks [8]. Just like in convolutional networks, such aggregation methods are designed to generalize the environment for the following manipulation.

Methods based on the attention mechanism

The problem with the aggregation methods described in the previous paragraph may be the fact that they do not provide interaction evaluation between the local representations of all users from the dataset and the considered one.

Some users in the dataset may behave more similar to him than others. Similar clients determine the user's closest environment and, correspondingly, help to better describe his behavior.

Thus, we need an aggregation method that can explicitly take into account the similarity of users and, respectively, their local representations. The attention mechanism, which was originally used to describe the similarity of the word embedding vectors in different languages in the machine translation task [41], is ideal for such a problem.

In this work, different variants of the attention mechanism are used.

Simple attention without learnable attention matrix

In the version without a training matrix, the global context vector for a given point in time has the form:

$$\mathbf{B}_t = X \operatorname{softmax}(X^T \mathbf{h}_t), \tag{3}$$

where $\mathbf{h}_t \in R^m$ is a vector of local representation for the considered user, and $X \in R^{m \times n}$ is a matrix, which rows are embeddings of all n users from dataset at a given time point. In this case, the user similarity metric is normalized by the softmax of the scalar product.

Attention with learnable attention matrix

For the method with a trained matrix, the formula is similar:

$$\mathbf{B}_t = X \text{softmax}(X^T A \mathbf{h}_t), \tag{4}$$

here $A \in R^{m \times m}$ – is the matrix to be trained.

In this case, before calculating the scalar product, vectors of representations from the dataset are additionally passed through a trained linear layer. Training of the attention matrix is built into the CoLES and AR model training pipelines.

Attention with symmetric attention matrix

In this approach, we model the attention matrix as a product of one matrix on its transposed version.

$$A = S^T S \tag{5}$$

That makes the resulting formula to be similar to the calculation of kernel dot product between the current representation vector and vectors from the train set with the linear kernel:

$$\mathbf{B}_t = X \text{softmax}(X^T S^T S \mathbf{h}_t) = X \text{softmax}((SX) S \mathbf{h}_t) = X \text{softmax}(< SX, S\mathbf{h}_t >), \tag{6}$$

where $< \cdot, \cdot >$ is the notation for dot product between all the vectors from SX and $S\mathbf{h}_t$ vector combined in one vector.

Kernel attention

We generalize the approach with a symmetric attention matrix and propose a kernel attention method. The main idea here is that we can use the general kernel dot product to calculate attention scores:

$$\mathbf{B}_t = X \text{softmax}(< \phi(X), \phi(\mathbf{h}_t) >), \tag{7}$$

here $\phi(*)$ is some learnable function. In our case, it is parameterized by some neural network.

Methods inspired by Hawkes process

Previous proposed methods do not take into account the time of the embeddings. But of course, this factor can affect the result as it is natural that the events that happened a long time ago should have less influence on the current moment compared to events that happened recently. To take this into consideration we turn to the Multivariate Hawkes processes [13].

The original Hawkes process is described by the equation:

$$\lambda_u(t) = \mu_u(t) + \sum_{v \in [m]} b_{uv} \sum_{e_i \in \mathcal{H}_v(t)} \kappa(t - t_i) = \mu_u(t) + \sum_{v \in [m]} \sum_{e_i \in \mathcal{H}_v(t)} b_{uv} \kappa(t - t_i), \tag{8}$$

where the first term, $\mu_u(t)$, models the current event sequence behavior, and the second term, with b_{uv} models interaction between sequence u and sequence v, $\kappa(t - t_i)$ is some function from time, usually exponential. $e_i \in \mathcal{H}_v(t)$ is an all events from sequence v before time t.

In the original Hawkes process events from one sequence have the same type in our case this is not true. Also as we work not with original sequences but with their representations we can simplify the equation:

$$\lambda_u(t) = \mu_u + \sum_{v} b(\mu_u, \mu_v) \kappa(t - t_v) \tag{9}$$

Now, in new notation, μ_u is a representation vector of the sequence under consideration, μ_v is a representation vector of the sequence from the train dataset, t is a current moment of time and t_v is the time moment of the last event in the v sequence at the current time point, and b and κ is some learnable kernels. The second term in the equation is the searched global representation. In the notation of the previous section of this work the equation can be rewritten:

$$\mathbf{B}_t = b(X, \mathbf{h}_t) \cdot \kappa(t \cdot \mathbf{1} - T), \tag{10}$$

where T is a vector of all last event times in X for current time t.

We simplify the given general formulation to the set of different special cases. Mostly it was done because of the instability in learning of general case. The different variations of the presented formulation of the problem are enumerated below.

Exponential Hawkes

In this simple variation of the Hawkes method we use an exponential kernel for time transformation and identical transformation for matrix X.

$$\mathbf{B}_t = X \exp\left(-(t \cdot \mathbf{1} - T)\right), \tag{11}$$

Actually in this case we are simply weighting the vectors from X with exponential time-dependent weights.

Exponential learnable Hawkes

The learnable transformation was used in this method. Firstly, we concatenate each vector from X with $\mathbf{h}_t$ and get matrix $Y \in R^{2m \times n}$. Secondly, we pass the concatenated matrix through feedforward neural network $\phi_{NN}(\cdot)$ and get the matrix $X' \in R^{m \times n}$.

$$\begin{aligned} \mathbf{B}_t = \phi_{NN}(\text{concatenate}(X, \mathbf{h}_t)) \exp\left(-(t \cdot \mathbf{1} - T)\right) = \\ \phi_{NN}(Y) \exp\left(-(t \cdot \mathbf{1} - T)\right) = \\ X' \exp\left(-(t \cdot \mathbf{1} - T)\right), \end{aligned} \tag{12}$$

This method is similar to the previous one with only one difference. Here, using a neural network, we additionally take into account the dependencies between the current embedding vector and embedding vectors from the dataset.

Learnable Hawkes

This variation of the method is fully learnable. The embeddings transformation works exactly the same way as in the previous exponential learnable Hawkes method, but here we also add learnable transformation for times deltas using feedforward neural network $\kappa_{NN}(\cdot)$.

$$\mathbf{B}_t = \phi_{NN}(\text{concatenate}(X, \mathbf{h}_t))\kappa_{NN}(t\mathbf{1} - T) \tag{13}$$

This method shows instability during learning. Losses tend to go to the infinity. So this method requires more in-depth study and results for it are not presented in this work.

Attention Hawkes

In this method, we decided to combine the usual exponential Hawkes and the attention methods:

$$\mathbf{B}_t = X\text{softmax}(X^T\mathbf{h}_t)\exp\left(-(t\cdot\mathbf{1} - T)\right), \tag{14}$$

This approach has double weighting: the first one accounts similarity of users and the second one accounts the time delta.

Implementation details

The learnable attention matrix A from the learnable attention method, learnable matrix S from the symmetrical attention method, and the learnable function $\phi(*)$ from the kernel attention method were trained in the CoLES and AR learning pipelines with fixed encoders.

For the $S \in R^{r\times m}$ matrix internal size r was taken to be equal to 100. For the learnable function $\phi(*)$ we use a two-layer neural network with a hidden size equal to 100.

The learnable transformation ϕ_{NN} from exponential learnable Hawkes was trained in the CoLES and AR learning pipelines with fixed encoders. For this transformation we use two-layer neural network with with hidden size equal to 100.

All the learnable elements were trained for 60 epochs with a batch size of 128 in the CoLES and AR pipeline.

A model using external information requires quite a lot of computing resources since it needs to store local representations of all users from the training set for all time points. To obtain these local representations, we use the encoder from the CoLES trained model.

Due to available memory requirements, we store only a random part of the set of local representations: for the Churn dataset, the number of clients to train in all experiments was 1000, and for the Default dataset, 300.

Results and Discussion

We start this chapter with methods for validating the resulting embeddings in terms of global and local properties and discussion of used datasets and metrics. Next, the chapter provides obtained results of different methods used to take into account external representations. The discussion of results and the proposition for further development of the project are also presented in this part.

All experiments were carried out in the Python programming language using the models and hyperparameters described in the methods chapter.

Approaches to measure the quality of received representations

One of the objectives of this work is to study the local and global properties of transactional data representations obtained using various methods.

From our point of view, global properties characterize the behavior of the client as a whole, throughout the entire history of his transactions. In contrast, local properties show the nature of the client's current state at a particular point in time. Moreover, different clients may be similar to each other at some point in time, in which case their local representations should also be similar. In addition, the local properties of the same client can change over time and the neural network must respond to this change.

Procedures for measuring the quality of representations in terms of their global and local properties are described below.

Methods for validation of the global property

To assess the quality of the obtained representations in terms of global properties, we followed the approach used in the paper [2], in which the CoLES model was proposed.

We used the Churn and Default datasets to solve the problem of binary classification of users on clients who left the bank and clients who did not repay the loan, respectively. This procedure consists of three steps, which are described below.

For an initial sequence of transactions of length T_i related to the ith user, we obtain the representation $\mathbf{H}_i \in R^d$, which characterizes the entire sequence of transactions as a whole.

Given fixed representations $\mathbf{H}_i$, we predict the binary target label $y_i \in \{0, 1\}$ using gradient boosting. As a specific implementation, the LightGBM [22] model is used, which works fast enough for large data samples and allows obtaining results of sufficiently high quality. The specific hyperparameters of the gradient boosting model are fixed (corresponding to [2]) and are the same for all base models under study.

The quality of the solution of a binary classification problem is measured using a standard set of metrics described in a metrics section.

This procedure allows you to evaluate how well the representations capture the client's "global" pattern across its history.

Local validation methods

Measurement of the quality of the obtained representations in terms of local properties is studied in various formulations.

In all cases, a sliding window procedure of size w is used to obtain local representations. To do this, for the ith user at time $t_j \in [t_w, T_i]$ the subsequence of his transactions $\mathbf{S}^i_{j-w:j}$. Next, this interval is passed through the encoder model under consideration to obtain a local representation $\mathbf{g}^i_j \in R^d$. An illustration of this approach can be found in Figure 2.

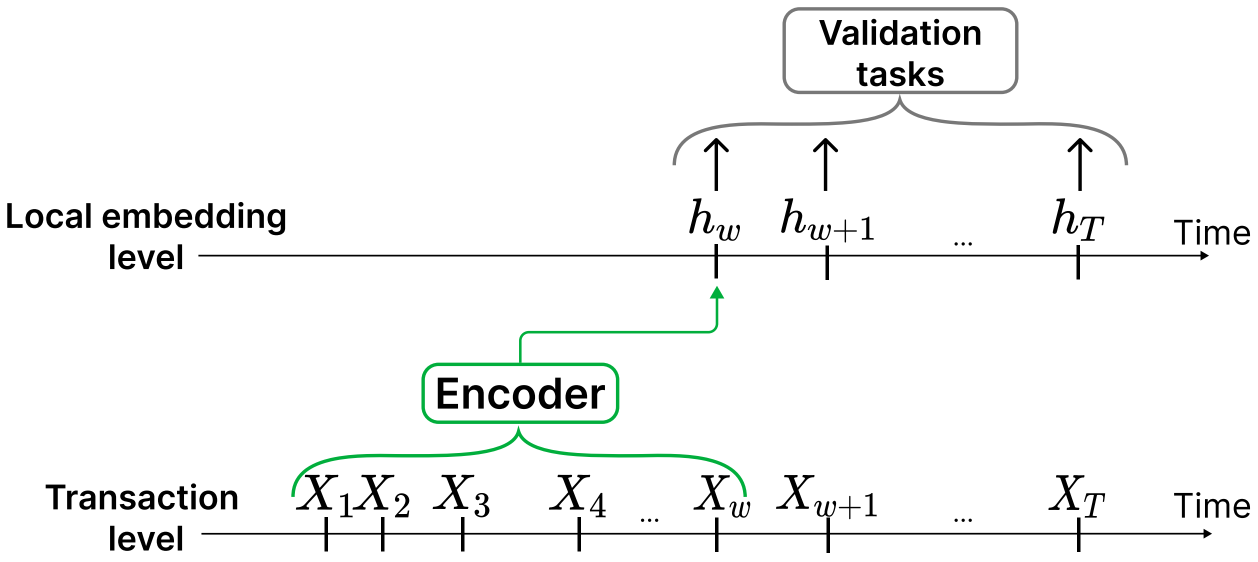

Figure 2: Usage of the sliding window for local validation approaches

The longer time interval the model uses, the greater the risk that the data it relies on will become outdated. Our artificial limitation allows us to reduce this effect for all models and also make their local properties stronger only due to the "relevance" of the data.

An obvious limitation of this approach is that it does not allow obtaining local representations at times $t \leq t^i_w$. In addition, the window size w is an additional hyperparameter that must be chosen, taking into account the fact that for small values of w the resulting representations do not contain enough information to solve local problems and for large values of w the representations lose local properties and become global in the limit $w \to T_i$.

As local problems, we explored different approaches, ranging from solving applied problems to studying the dynamics of representations. Below is a detailed description of each method.

Applied problems of transaction classification.

For the Churn and Default samples studied in the project, it is proposed to consider the corresponding local binary classification tasks (downstream tasks).

In the Churn dataset, target (global) labels correspond to those clients who stopped using the bank's services at some time horizon. It is logical to assume that the behavior of customers who are about to stop using a bank card begins to change in advance: they make fewer transactions, do not top up the card, transactions become less frequent, etc.

Given this, we created local binary labels $c_j^i \in \{0, 1\}$ for all transactions in the sequence. For empirical reasons, it was proposed to select the "early outflow" horizon equal to one month. That is, all transactions of a user who left the bank during the month before his last transaction were marked with the 1 tag, and the rest were marked with the 0 tag.

For the Default sample, similar local binary labels were created, which reflected the client's transition to the "pre-default" state.

Thus, to measure the local properties of embeddings, the problems of their binary classification were considered: the label c_j^i was predicted from the local representation $\mathbf{h}_j^i$. This problem is solved using a simple neural network model; in particular, in this work, we use a single-layer perceptron. Accordingly, the better the embedding describes the behavior of the client "at the moment", that is, locally, the better a simple classifier model will be able to solve applied problems.

Predict the MCC code of the next transaction.

This validation approach was inspired by the work [45], in which it was proposed to predict the type of the next event based on the history of observations — in our case, the MCC code of the next transaction.

Formally, in this case, the multiclass classification problem is solved: using the local representation $\mathbf{h}_j^i$, the MCC code of the transaction of the ith client, completed at time t_{j+1}, was predicted.

Note that there are a lot of types that are rare in datasets. From a business perspective, such categories are often less interesting and meaningful. Therefore, to simplify the task, in this procedure for testing local properties, it was decided to leave only transactions that correspond to the 100 most popular codes. For more information on the distribution of transaction types, see section .

Implementation details

For global validation we follow the procedure described in [2] and use the same hyperparameters for the boosting model. For local validation we follow the procedure described in the corresponding section in the methods chapter and use the following hyperparameters: window size equal to 32, stride equal to 16, batch size equal to 512, and a maximum number of epochs equal to 10.

Metrics

A complete validation of the proposed methods is provided using a set of classical metrics. As stated above, our validation approached involve binary and multiclass classification tasks.

Classification metrics For classification problems, we utilize the standard metrics: Accuracy, ROC-AUC, and PR-AUC [36, 28, 29]. A better model produces higher values for these three metrics.

Accuracy is defined as the ratio of correct predictions to all predictions made:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{15}$$

where TP — true positive, TN — true negative, FP — false positive, and FN — false negative predictions.

ROC-AUC, or the area under the Receiver Operating Characteristic (ROC) curve, represents the relationship between true positive rate (TPR or Recall) and false positive rate (FPR) for different thresholds. Similarly, the PR-AUC is the area under the Precision-Recall curve, which describes the connection between Precision and Recall (TPR). Formally,

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{TP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}. \tag{16}$$

For multiple classes, we use micro-averaging for Accuracy and weighted averaging for ROC-AUC and PR-AUC. For such formulation contribution of each class to the result is proportional to the number of its members. It makes these metrics to be insensitive to class imbalance.

Datasets

This section describes the datasets used to compare the models and methods under study.

General overview of the data. EDA

In this work, we work with two open samples of transactional data: Churn [1] and Default [2]. Descriptions of these datasets are given below. The main characteristics of the samples are given in Table 1.

Churn.

This dataset contains transactional data of bank customers. For global validation, it is proposed to solve the problem of binary classification of clients depending on whether the client left the bank or not. At the same time, the classes are almost balanced. The data was used previously, for example, in the work [2].

Default.

The dataset also contains transactional data of bank customers. For global validation, it is proposed to solve the problem of binary classification of clients depending on whether the client was able to repay the loan to the bank. There is a significant class imbalance in this sample. This dataset was offered to participants at one of the competitions of a large bank and is available in the repository `pytorch-lifestream`. Unlike most other open transaction data in this dataset, the target variable is close to real business problems, as it corresponds to a credit scoring problem.

Data preprocessing

For more convenient work with data, we carried out minimal preprocessing. We brought all-time data to one format, removed unnecessary columns, and encoded MCC (Merchant category code) codes by frequency (i.e. 1 is the most frequent MCC code, 345 is the rarest). The last change is done so that there are no gaps in the categorical task, and to make it technically easier to take into account rare MCC codes (as, for example, in problems with generative models).

[1]https://boosters.pro/championship/rosbank1
[2]https://boosters.pro/championship/alfabattle2

Table 1: Basic statistics of dataset used in experiments

	Churn	Default
Number of transactions	490 513	2 124 000
Number of clients	5 000	7 080
Min. length of sequences	1	300
Max. length of sequences	784	300
Median length of sequences	83	300
Number of MCC codes	344	309
Class Balance	0.55 : 0.45	0.04 : 0.96

To solve the local problem, it was also necessary to add local targets: ”will the client leave within a month” in the case of Churn, and ”will the client go bankrupt within a month” in the case of Default. These labels are generated using the original target labels available in the datasets.

Conclusions

In this project, we consider specific for the banking industry transactional data. We will conduct experiments on two datasets, Churn and Default.

The presented datasets have sufficiently large sizes and high quality. This ensures the reliability of statistical conclusions and also reduces the risk of overfitting the neural network models used in the project with a large number of parameters. With a high probability, the obtained results will be reproduced on larger datasets that are used to build models in the bank.

In addition, these data have been used previously in other studies. This allows us to correctly compare the resulting models with those previously built for these samples.

Thus, the selected datasets allow us to fully evaluate the quality of the developed methods. Since they are large and have been used previously, the conclusions drawn from these data will accurately reflect the effectiveness and applicability of the proposed methods in real-world settings.

Results

In this section, we describe the results of experiments on obtaining an external context representation and using it to improve existing models.

In the experiments, we investigated the following types of aggregation of representations to obtain a context vector: averaging (**Mean**), maximization (**Max**), attention mechanism without a learning matrix (**Attention**), with a learning matrix (**Learnable attention**), attention with symmetric matrix (**Symmetrical attention**), kernel attention (**Kernel attention**), exponential Hawkes (**Exp Hawkes**), exponential learnable Hawkes with (**Exp learnable Hawkes**), and Hawkes with attention (**Attention Hawkes**). We learn all the learnable parts of these methods using CoLES and AR pipelines with fixed original encoders and compare the results with a conventional CoLES and AR encoder without adding external context information (**Without context**).

Models using external information were tested in the same way as other models for obtaining representations of transactional data. The experimental results for the Churn and Default datasets are presented in Figures 3 and 4 and in Tables 2 and 3. All results were averaged across 3 different pre-trained encoder models. Also the table 4 contains the ranks of all methods averaged by the mean ROC-AUC metric for both datasets.

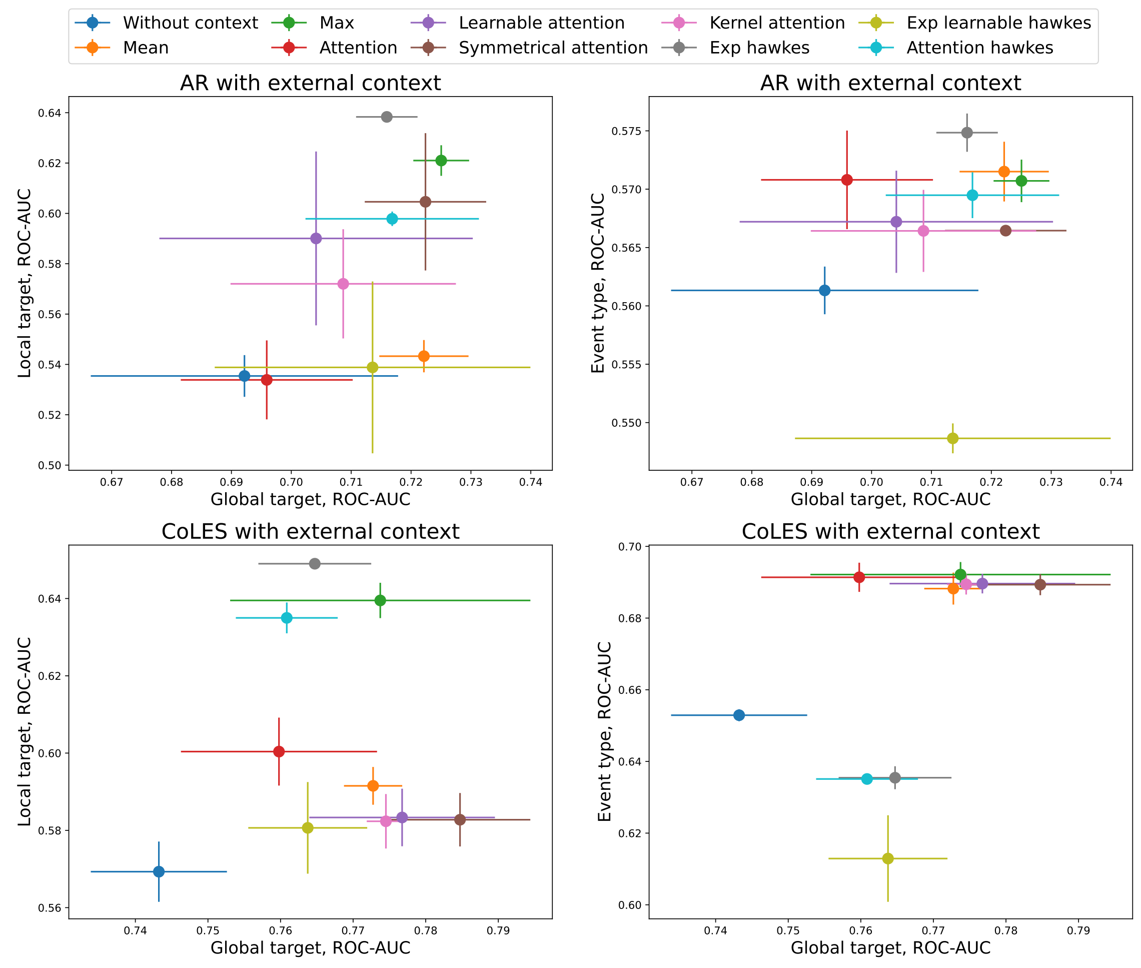

Figure 3: Quality of the models regarding their global and local properties on the *Churn* dataset. The x-axis at each graph corresponds to global validation ROC-AUC, while the y-axis shows the local target or next MCC prediction ROC-AUC. Thus, the upper and righter the dot is, the better model it represents

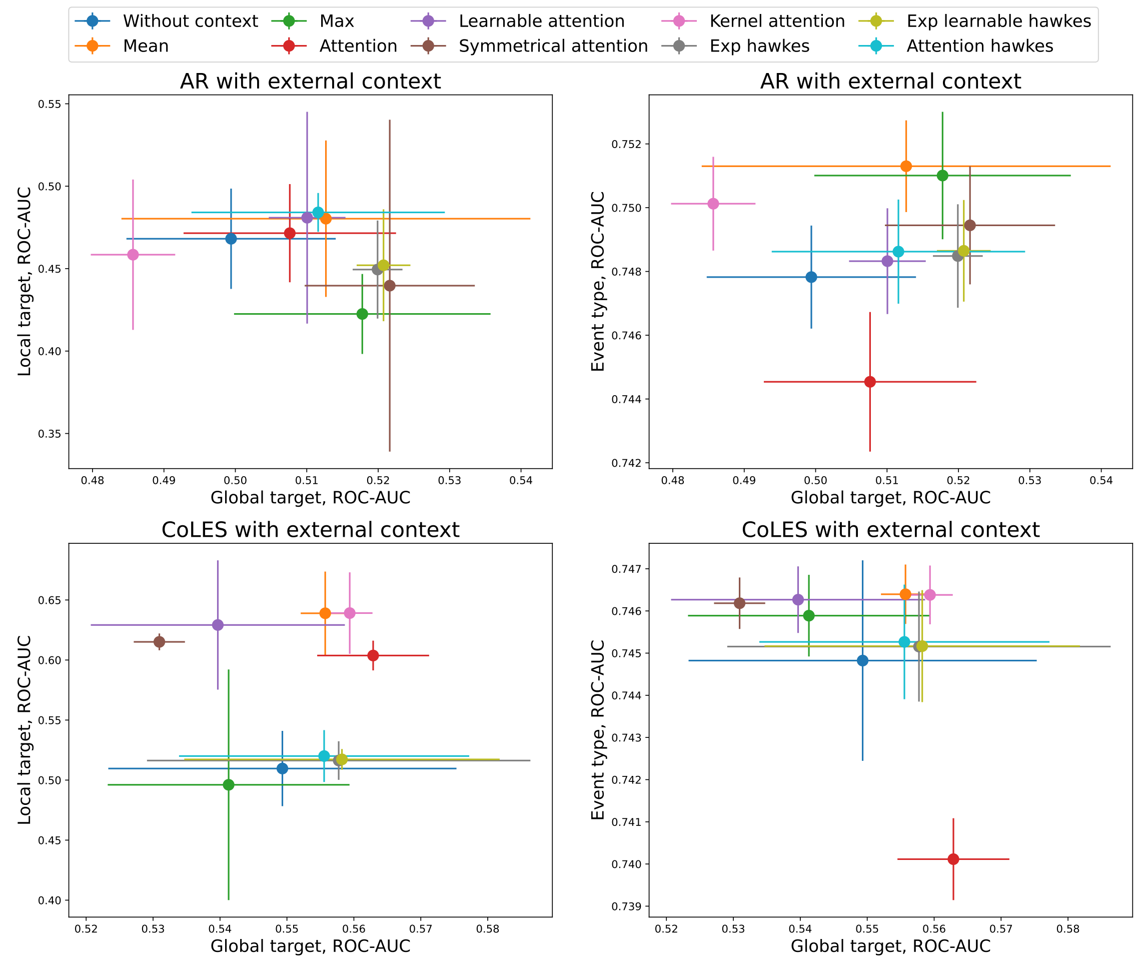

Figure 4: Quality of the models regarding their global and local properties on tje *Default* dataset. The x-axis at each graph corresponds to global validation ROC-AUC, while the y-axis shows the local target or next MCC prediction ROC-AUC. Thus, the upper and righter the dot is, the better model it represents

Discussion of the results

Experimental results show that using external context improves metrics in most cases. This is especially noticeable in the balanced Churn sample, for which the metrics improve across most testing procedures. On the unbalanced Default sample, contextual representations also help models solve local and global problems, but this is not as explicit as in the case of a balanced dataset.

Comparison of different methods results

Among all the above approaches, the attention methods with different learnable parts (Learnable attention, Symmetrical attention, Kernel attention) can be distinguished: they most often end up

Table 2: Quality metrics for global and local embedding validation results on the *Churn* dataset. All metrics in the Table should be maximized. The results are averaged by three runs and are given in the format $mean \pm std$. The best values are **highlighted**, and the second-best values are underlined.

	CoLES			AR		
	ROC-AUC ↑	PR-AUC ↑	Accuracy ↑	ROC-AUC ↑	PR-AUC ↑	Accuracy ↑
	Global target validation					
Without context	0.743 ± 0.009	0.792 ± 0.014	0.689 ± 0.005	0.692 ± 0.025	0.734 ± 0.032	0.657 ± 0.013
Mean	0.773 ± 0.004	0.828 ± 0.003	**0.715**± 0.010	0.722± 0.007	0.776 ± 0.005	0.653 ± 0.008
Max	0.774 ± 0.021	0.818± 0.032	0.701 ± 0.004	**0.725** ± 0.005	**0.777** ± 0.002	0.664± 0.012
Attention	0.760 ± 0.014	0.808 ± 0.017	0.691± 0.010	0.696 ± 0.014	0.744 ± 0.017	0.644 ± 0.022
Learn. attention	0.777 ± 0.013	0.830 ± 0.013	0.699 ± 0.006	0.704 ± 0.026	0.751 ± 0.020	0.655 ± 0.009
Sym. attention	**0.785** ± 0.010	**0.835** ± 0.005	0.703 ± 0.017	0.722 ± 0.010	0.769 ± 0.004	**0.671** ± 0.009
Kernel attention	0.775 ± 0.003	0.824 ± 0.002	0.693 ± 0.009	0.709 ± 0.019	0.760 ± 0.003	0.655 ± 0.014
Exp Hawkes	0.765 ± 0.008	0.814 ± 0.009	0.699 ± 0.012	0.716 ± 0.005	0.767 ± 0.013	0.661 ± 0.011
Exp learn. Hawkes	0.764 ± 0.008	0.812 ± 0.008	0.703 ± 0.006	0.714 ± 0.025	0.758 ± 0.020	0.665 ± 0.024
Attention Hawkes	0.761 ± 0.007	0.796 ± 0.009	0.702 ± 0.007	0.717 ± 0.014	0.751 ± 0.023	0.667 ± 0.006
	Local target validation					
Without context	0.569 ± 0.008	0.321 ± 0.003	0.732 ± 0.000	0.535 ± 0.008	0.299 ± 0.011	0.732 ± 0.000
Mean	0.592 ± 0.005	0.342 ± 0.005	0.732 ± 0.000	0.543 ± 0.006	0.312 ± 0.006	0.732± 0.000
Max	0.640 ± 0.005	**0.400** ± 0.006	0.732 ± 0.000	0.621 ± 0.006	0.256 ± 0.008	0.732 ± 0.000
Attention	0.600 ± 0.009	0.348 ± 0.010	0.732 ± 0.000	0.534 ± 0.016	0.301 ± 0.007	0.732 ± 0.000
Learn. attention	0.583 ± 0.007	0.330 ± 0.008	0.732 ± 0.000	0.590 ± 0.035	0.338 ± 0.025	0.732 ± 0.000
Sym. attention	0.583 ± 0.007	0.329 ± 0.007	0.732 ± 0.000	0.605 ± 0.027	0.350 ± 0.020	0.731 ± 0.002
Kernel attention	0.582 ± 0.007	0.329 ± 0.007	0.732 ± 0.000	0.572 ± 0.021	0.330 ± 0.023	0.732 ± 0.000
Exp Hawkes	**0.649** ± 0.0004	0.366 ± 0.003	0.732 ± 0.000	**0.638** ± 0.001	**0.351** ± 0.001	0.732 ± 0.0002
Exp learn. Hawkes	0.581 ± 0.012	0.322 ± 0.013	0.732 ± 0.000	0.539 ± 0.034	0.293 ± 0.025	0.730 ± 0.005
Attention Hawkes	0.635 ± 0.004	0.359 ± 0.005	0.732 ± 0.000	0.598 ± 0.003	0.331 ± 0.001	0.732 ± 0.000
	Next event type validation					
Without context	0.653 ± 0.002	0.168 ± 0.001	0.239 ± 0.003	0.561 ± 0.002	0.111 ± 0.002	0.237 ± 0.002
Mean	0.688 ± 0.004	0.193 ± 0.003	0.242 ± 0.001	0.572 ± 0.003	0.119 ± 0.003	0.238 ± 0.004
Max	**0.692** ± 0.003	0.192 ± 0.001	0.239 ± 0.002	0.570 ± 0.002	0.115 ± 0.001	0.231 ± 0.002
Attention	0.691 ± 0.004	**0.194** ± 0.004	0.243 ± 0.001	0.571 ± 0.004	0.117 ± 0.002	0.236 ± 0.005
Learn. attention	0.690 ± 0.003	**0.194** ± 0.004	0.241 ± 0.001	0.567 ± 0.004	0.116 ± 0.001	0.238 ± 0.003
Sym. attention	0.689 ± 0.003	0.193 ± 0.004	0.241± 0.0004	0.566 ± 0.0003	0.115 ± 0.002	0.238 ± 0.003
Kernel attention	0.689 ± 0.003	0.193 ± 0.004	0.241 ± 0.0004	0.566 ± 0.004	0.115 ± 0.001	0.238 ± 0.002
Exp Hawkes	0.635 ± 0.003	0.161 ± 0.002	0.244 ± 0.002	**0.575** ± 0.002	**0.122** ± 0.001	**0.244** ± 0.001
Exp learn. Hawkes	0.613 ± 0.012	0.153 ± 0.007	**0.257** ± 0.011	0.549 ± 0.001	0.113 ± 0.002	0.241 ± 0.005
Attention Hawkes	0.635 ± 0.001	0.159 ± 0.002	0.246 ± 0.001	0.569 ± 0.002	0.118 ± 0.001	0.241 ± 0.003

among the leaders or on a par with them, especially in local validation tasks as local target and next event type prediction. This is not surprising since approaches based on the attention mechanism help to well identify local patterns in sequences [41]. Also among the leaders for local tasks we can find approaches inspired by the Hawkes process (Exp Hawkes, Exp learnable Hawkes, Attention Hawkes). This can be explained by the fact that these methods take into account the temporal distance of sequences embeddings and because of this they respond better to local temporal changes in the data, wich help in local tasks.

Mean and Max methods show good performance in both local and global validation tasks. But sometimes these methods fail in local target prediction task. For example, the Max method on the Default dataset and the Mean method on the Churn dataset. This can be explained as follows:

Table 3: Quality metrics for global and local embedding validation results on the *Default* dataset. All metrics in the Table should be maximized. The results are averaged by three runs and are given in the format $mean \pm std$. The best values are **highlighted**, and the second-best values are underlined

	CoLES			**AR**		
	ROC-AUC ↑	PR-AUC ↑	Accuracy ↑	ROC-AUC ↑	PR-AUC ↑	Accuracy ↑
	Global target validation					
Without context	0.549 ± 0.026	**0.058** ± 0.002	0.962 ± 0.000	0.499 ± 0.015	0.054 ± 0.003	0.962 ± 0.000
Mean	0.556 ± 0.004	0.056 ± 0.009	0.962 ± 0.000	0.513 ± 0.029	0.065 ± 0.019	0.961 ± 0.001
Max	0.541 ± 0.018	0.049 ± 0.005	0.962 ± 0.000	0.518 ± 0.018	0.065 ± 0.017	0.962 ± 0.000
Attention	**0.563** ± 0.008	0.050 ± 0.005	0.962 ± 0.000	0.508 ± 0.015	**0.067** ± 0.026	0.962 ± 0.000
Learn. attention	0.540 ± 0.019	0.073 ± 0.032	0.962 ± 0.000	0.510 ± 0.005	0.050 ± 0.008	0.961 ± 0.001
Sym. attention	0.531 ± 0.004	0.055 ± 0.008	0.962 ± 0.000	**0.522** ± 0.012	0.063 ± 0.015	0.962 ± 0.000
Kernel attention	0.559 ± 0.003	0.057 ± 0.003	0.962 ± 0.000	0.486 ± 0.006	0.055 ± 0.010	0.962 ± 0.000
Exp Hawkes	0.558 ± 0.029	0.071 ± 0.039	0.962 ± 0.000	0.520 ± 0.003	0.061 ± 0.009	0.962 ± 0.000
Exp learn. Hawkes	0.558 ± 0.024	0.061 ± 0.025	0.962 ± 0.000	0.521 ± 0.004	0.062 ± 0.010	0.962 ± 0.000
Attention Hawkes	0.556 ± 0.022	0.063 ± 0.024	0.962 ± 0.000	0.512 ± 0.018	0.064 ± 0.015	0.962 ± 0.000
	Local target validation					
Without context	0.510 ± 0.031	0.006 ± 0.0003	0.994 ± 0.000	0.468 ± 0.030	0.005 ± 0.0002	0.994 ± 0.000
Mean	**0.639** ± 0.035	0.008 ± 0.001	0.994 ± 0.000	0.480 ± 0.047	**0.006** ± 0.001	0.994± 0.000
Max	0.496 ± 0.096	0.006 ± 0.001	0.994 ± 0.000	0.422 ± 0.024	0.005 ± 0.0002	0.994 ± 0.000
Attention	0.604 ± 0.012	0.008 ± 0.001	0.994 ± 0.000	0.472 ± 0.030	0.005 ± 0.0001	0.994 ± 0.000
Learn. attention	0.629 ± 0.054	0.008 ± 0.001	0.994 ± 0.000	0.481 ± 0.064	0.005 ± 0.001	0.994 ± 0.000
Sym. attention	0.615 ± 0.007	0.008 ± 0.0003	0.994 ± 0.000	0.440 ± 0.101	0.005 ± 0.01	0.994 ± 0.000
Kernel attention	**0.639** ± 0.034	0.008 ± 0.001	0.994 ± 0.000	0.458 ± 0.046	0.005 ± 0.001	0.994 ± 0.000
Exp Hawkes	0.516 ± 0.016	0.006 ± 0.0004	0.994 ± 0.000	0.449 ± 0.030	0.005 ± 0.0004	0.994 ± 0.000
Exp learn. Hawkes	0.517 ± 0.009	0.006 ± 0.0002	0.994 ± 0.000	0.452 ± 0.034	0.005 ± 0.0003	0.994 ± 0.000
Attention Hawkes	0.520 ± 0.022	0.006 ± 0.001	0.994 ± 0.000	**0.484** ± 0.012	0.005 ± 0.0002	0.994 ± 0.000
	Next event type validation					
Without context	0.745 ± 0.002	0.260 ± 0.002	0.333 ± 0.004	0.745 ± 0.002	0.273 ± 0.0001	0.248 ± 0.002
Mean	0.746 ± 0.001	0.261 ± 0.001	0.331 ± 0.001	**0.751** ± 0.001	**0.276** ± 0.001	**0.350** ± 0.0002
Max	0.746 ± 0.001	0.260 ± 0.001	0.331 ± 0.001	**0.751** ± 0.002	0.275 ± 0.001	0.348 ± 0.003
Attention	0.740 ± 0.001	0.253 ± 0.002	**0.335** ± 0.002	0.745 ± 0.002	0.269 ± 0.002	0.347 ± 0.002
Learn. attention	0.746 ± 0.001	**0.262** ± 0.001	0.333 ± 0.001	0.748 ± 0.002	0.274 ± 0.001	0.349 ± 0.001
Sym. attention	0.746 ± 0.001	0.261 ± 0.001	0.331 ± 0.001	0.749 ± 0.002	0.275 ± 0.001	0.348 ± 0.003
Kernel attention	**0.747** ± 0.001	0.261 ± 0.001	0.331 ± 0.001	0.750 ± 0.001	**0.276** ± 0.001	0.349 ± 0.003
Exp Hawkes	0.745 ± 0.001	0.261 ± 0.001	0.333 ± 0.0003	0.748 ± 0.002	0.274 ± 0.001	0.348 ± 0.001
Exp learn. Hawkes	0.745 ± 0.001	0.261 ± 0.001	0.334 ± 0.0002	0.749 ± 0.002	0.274 ± 0.001	0.348 ± 0.001
Attention Hawkes	0.745 ± 0.001	0.261 ± 0.001	0.333 ± 0.001	0.749 ± 0.002	0.275 ± 0.001	0.348 ± 0.001

these two methods provide some average representation of all users, which can help in global classification tasks, but because of this they can also smooth out the local representations of individual users, which negatively affects the quality of local tasks.

Comparison of models based on CoLES and AR

For models trained using the AR algorithm, classical methods are often the leaders in local and global validation results (Mean, Max). This may be due to the fact that AR models are initially better suited to take into account local features of sequences, while Mean and Max work better with global problems. This combination of opposites gives an increase in quality. Also, Exp Hawkes

Table 4: Average ranks (by ROC-AUC metric) for both datasets. All metrics in the Table should be minimized. The best values are **highlighted**, the second-best values are $\underline{\text{underlined}}$, and the third-best values are $\underline{\underline{\text{double underlined}}}$

	Global target validation		*Local target validation*		*Next event type validation*		*Mean*
	CoLES	AR	CoLES	AR	CoLES	AR	
Without context	8.5	9.5	9.5	7	7.25	9.25	8.5
Mean	$\underline{\underline{5.25}}$	$\underline{\underline{3.75}}$	**3.25**	5	4.75	**1.75**	**3.958**
Max	6	$\underline{2.5}$	6	6	**2.25**	$\underline{2.75}$	$\underline{4.25}$
Attention	5	8.5	$\underline{4.5}$	7	6	6.25	6.208
Learn. attention	5.5	7.5	$\underline{\underline{4.75}}$	$\underline{3.5}$	$\underline{\underline{3.25}}$	6.75	5.208
Sym. attention	5.5	**1.75**	5.25	6	4	6.25	$\underline{\underline{4.792}}$
Kernel attention	**2.5**	8.5	$\underline{\underline{4.75}}$	6	$\underline{2.75}$	5.25	4.958
Exp Hawkes	$\underline{4.75}$	4	$\underline{4.5}$	$\underline{\underline{4.5}}$	8	$\underline{\underline{4.25}}$	5
Exp learn. Hawkes	$\underline{\underline{5.25}}$	4	8	7.5	8.75	7.5	6.833
Attention Hawkes	6.75	5	$\underline{4.5}$	**2.5**	8	5	5.292

usually work better with AR models. Actually, this method is close to the classical methods as it is nothing more than a weighted sum of the vectors from the training dataset.

For CoLES models, the best results are often shown by various variations of attention-based methods (Attention, Learnable attention, Symmetrical attention, Kernel attention). As in the case of AR models, this can be explained by the unity of opposites: the original CoLES model aims to highlight global properties due to the nature of its contrastive training, and the attention mechanism takes more into account some local features of sequences.

Further development

Although this work presents a broad comparative analysis of various methods of addition of context information, there are still a number of issues that can be considered by researchers.

First of all, it is interesting to propose some new methods for external context analysis. Continuation of the development of methods based on Hawkes process can be especially complex. It was already mentioned that, for example, a fully learnable analog is not presented in this work as learning of it is unstable.

Secondly, it is intriguing to dive deeper into the investigation of methods based on the attention. For example, we can compare the attention matrices obtained in the Learnable attention and Symmetrical attention approach. Such research can shed light on exactly how different embeddings interact with each other.

Conclusion

In this work, we consider the task of creating representations for transaction sequences, which are non-uniform event sequences. We adapted existing approaches: CoLES based on contrastive learning and AR based on the prediction of the next event. Also one of the main tasks for the project was the creation of the validation which can show both local and global properties of representations.

The main contribution of this work is that we introduced novel methods for incorporating external contextual information, thereby enhancing the quality of existing approaches and foundational solutions. We prove that accounting of the external information can improve models for constructing representation vectors for event sequences. Also, we demonstrate that the external context representation can be obtained from the aggregation of representations of all sequences at the current time point. We propose a lot of different methods for such aggregation based on pooling, attention mechanism, and inspired by Hawkes process. The representations obtained using the addition of external context vectors give better quality at applied problems in particular on bank transactions data domain and churn and credit scoring tasks.

We analyzed the obtained results and came up with the conclusion that the most effective results were achieved using an approaches based on a trainable attention mechanism, which identifies the closest clients influencing the user under consideration, and the Hawkes process which considers time difference for embeddings. Mean and Max aggregations also show outperforming results in the global task. Also, we found out that different aggregation methods are suitable for different training pipelines, and usually dissimilar properties of the original encoder and aggregation type work well.

Author contribution

All experiments, processing of results, and writing of the text in this work were performed by the author. Evgenia Romanenkova, Alexandra Bazarova, Ilya Kuleshov, Alexander Stepikin, Alexandr Yugay, and Alexey Zaytsev took part in the development of the baseline models and validation pipelines.

Publications

Bazarova, A., Kovaleva, M., Kuleshov, I., Romanenkova, E., Stepikin, A., Yugay, A., Mollaev, D., Kireev, I., Savchenko, A., and Zaytsev, A. Universal representations for financial transactional data: embracing local, global, and external contexts. *arXiv preprint arXiv:2404.02047* (2024)

Acknowledgements

I would like to express my gratitude to my scientific advisor Alexey Zaytsev and the team from the Laboratory of applied research Skoltech-Sberbank that worked with me on this project and writing the article: Evgenia Romanenkova, Alexandra Bazarova, Ilya Kuleshov, Alexander Stepikin, Alexandr Yugay.

Bibliography

[1] Ala'raj, M., Abbod, M. F., Majdalawieh, M., and Jum'a, L. A deep learning model for behavioural credit scoring in banks. *Neural Computing and Applications* (2022), 1–28.

[2] Babaev, D., et al. Coles: Contrastive learning for event sequences with self-supervision. In *Proceedings of the 2022 International Conference on Management of Data* (2022).

[3] Bazarova, A., Kovaleva, M., Kuleshov, I., Romanenkova, E., Stepikin, A., Yugay, A., Mollaev, D., Kireev, I., Savchenko, A., and Zaytsev, A. Universal representations for financial transactional data: embracing local, global, and external contexts. *arXiv preprint arXiv:2404.02047* (2024).

[4] Begicheva, M., and Zaytsev, A. Bank transactions embeddings help to uncover current macroeconomics. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2021).

[5] Bin Sulaiman, R., Schetinin, V., and Sant, P. Review of machine learning approach on credit card fraud detection. *Human-Centric Intelligent Systems 2*, 1-2 (2022), 55–68.

[6] Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. GPT-NeoX-20B: An Open-Source autoregressive language model. In *Proceedings of BigScience Episode – Workshop on Challenges Perspectives in Creating Large Language Models* (Stroudsburg, PA, USA, 2022), Association for Computational Linguistics.

[7] Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., et al. AudioLM: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 31* (2023), 2523–2533.

[8] Boureau, Y.-L., Ponce, J., and LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 111–118.

[9] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *Advances in neural information processing systems 33* (2020), 1877–1901.

[10] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 9650–9660.

[11] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (2020), PMLR, pp. 1597–1607.

[12] Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. PixelSNAIL: An improved autoregressive generative model. In *International Conference on Machine Learning* (2018), PMLR, pp. 864–872.

[13] De, A., Valera, I., Ganguly, N., Bhattacharya, S., and Gomez Rodriguez, M. Learning and forecasting opinion dynamics in social networks. *Advances in neural information processing systems 29* (2016).

[14] Deldari, S., et al. Time series change point detection with self-supervised contrastive predictive coding. In *Proceedings of the Web Conference 2021* (2021).

[15] Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (June 2021), IEEE, pp. 12873–12883.

[16] Ethayarajh, K. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), Association for Computational Linguistics.

[17] Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 2, pp. 1735–1742.

[18] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 16000–16009.

[19] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[20] Hoffer, E., and Ailon, N. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3* (2015), Springer, pp. 84–92.

[21] Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. A survey on contrastive self-supervised learning. *Technologies 9*, 1 (2020), 2.

[22] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems 30* (2017), 3146–3154.

[23] Kenton, J. D. M. W. C., and Toutanova, L. K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT* (2019).

[24] Li, T., Kou, G., and Peng, Y. A new representation learning approach for credit data analysis. *Information Sciences 627* (2023), 115–131.

[25] Li, Z., Liu, G., and Jiang, C. Deep Representation Learning With Full Center Loss for Credit Card Fraud Detection. *IEEE Transactions on Computational Social Systems 7*, 2 (Apr. 2020), 569–579.

[26] Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering 35*, 1 (2023), 857–876.

[27] Mancisidor, R. A., Kampffmeyer, M., Aas, K., and Jenssen, R. Learning latent representations of bank customers with the variational autoencoder. *Expert Systems with Applications 164* (2021), 114020.

[28] Marceau, L., Qiu, L., Vandewiele, N., and Charton, E. A comparison of deep learning performances with other machine learning algorithms on credit scoring unbalanced data. *arXiv preprint arXiv:1907.12363* (2019).

[29] Moscato, V., Picariello, A., and Sperlí, G. A benchmark of machine learning approaches for credit score prediction. *Expert Systems with Applications 165* (Mar. 2021), 113986.

[30] Moskvoretskii, V., Osin, D., Shvetsov, E., Udovichenko, I., Zhelnin, M., Dukhovny, A., Zhimerikina, A., Efimov, A., and Burnaev, E. Self-supervised learning in event sequences: A comparative study and hybrid approach of generative modeling and contrastive learning. *arXiv preprint arXiv:2401.15935* (2024).

[31] Oord, A., et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[32] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[33] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. Tech. rep., OpenAI, 2018.

[34] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9.

[35] Razavi, A., Van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with VQ-VAE-2. *Proceedings of the 33rd International Conference on Neural Information Processing Systems 32* (Dec. 2019), 14866–14876.

[36] Romanenkova, E., et al. Similarity learning for wells based on logging data. *Journal of Petroleum Science and Engineering* (2022).

[37] Shchur, O., Biloš, M., and Günnemann, S. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations* (2019).

[38] Thomas, L. C. A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International journal of forecasting 16*, 2 (2000), 149–172.

[39] Tschannen, M., Bachem, O., and Lucic, M. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069* (2018).

[40] Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *Proceedings of the 33rd International conference on machine learning* (June 2016), vol. 48, PMLR, pp. 1747–1756.

[41] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[42] Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2022), vol. 36, pp. 8980–8987.

[43] Zbontar, J., et al. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning* (2021), PMLR.

[44] Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. Self-attentive hawkes process. In *International conference on machine learning* (2020), PMLR, pp. 11183–11193.

[45] Zhuzhel, V., et al. Continuous-time convolutions model of event sequences. *arXiv preprint arXiv:2302.06247* (2023).