An Approach for Complex Event Streams Processing and Forecasting

Viktor Morozov, Mikhail Petrovskiy
Lomonosov Moscow State University
Moscow, Russia
v.morozov.ml@gmail.com, michael@cs.msu.su

Abstract—**Building complex event representation and event forecasting are two important problems which are usually solved separately. In this work we propose general approach that allows to incorporate all available information about events, such as numeric, categorical and binary features and even some text descriptions. We propose ways to build latent representation of complex events using one of dimensionality reduction methods (matrix factorization, neural autoencoders) together with forecasting further occurrences of these events. The experimental results compare combinations of different methods for event representation and forecasting.**

I. INTRODUCTION

Event flow processing and forecasting tasks can be found almost in any area nowadays. In practice these tasks are solved when predicting the prices in stock markets, forecasting the traffic flows, amount of further purchases, number of taxi orders and almost any timestamped data processing. In this work we consider *event* as set of features and timestamp. Mainly we focus on *complex* events, i.e. events with complex structure. Event is considered complex if it is described by features of different types: numeric, categorical, binary and text descriptions.

As another example we could mention forecasting real world events. Usually these events are described by several categorical features (country, city, etc.), numerical ones and text descriptions. Efficient use of all available information might significantly improve forecasting results.

It can be noticed that the aforementioned description of event does not require equal time intervals between any two events. It means that we cannot directly apply common well-studied approaches for time series forecasting. Also, taking into account complex structure of such events is not clear how to aggregate event categorical features or text descriptions over some time interval, e.g. day, week or month.

Taking all this into account, following challenge arises: we should be able to transform complex object descriptions into a numerical latent feature space with minimal information loss and use historical information about changes of these latent features to perform forecasting.

To handle this challenge we propose general approach, which allows to take into account all available event features. This approach includes two main parts: building latent event representation together with its aggregation and forecasting further events. We could mention two desired properties of event representation: it should be possible to interpret and aggregate latent event descriptions. Then we represent original complex features in latent space and aggregate them in any way we want, obtaining multivariate time series. Finally we can perform event forecasting based on this latent representation, which is well-studied task of multivariate time series forecasting.

In this work we focus on solving two forecasting tasks: predicting probability of event occurring during some time interval and predicting number of events that will occur during some period. Since predicting probability of unspecified event does not make sense, we would like to notice, that during solving these tasks we split events in two parts: *events of interest* and *other events*. It means that we are interested in forecasting events with specific properties (e.g. price increase in stock market, not just any price behaviour). Further in forecasting tasks we denote these events of interests as just *events* for simplicity.

We define the task of forecasting probability as predicting probability p of the event during the next aggregation time interval. Since predicted probability lies in $[0, 1]$, this task could be naturally reduced to the *classification*.

The task of forecasting number of events is similar to previous one, but instead of predicting probability p, we will predict number of events that will occur during the next aggregation time interval. Therefore our target variable is event counter and it lies in $[0, +\infty)$. From the definition it follows that we can consider this task a *regression*.

II. RELATED WORKS

Forecasting, especially for time series, has a good study history in the literature. While solving this task, authors often focus on forecasting or modeling multivariate time series. This kind of data has numeric values (also known as *observations*) and equal time intervals between the observations. Such regular structure does not require any special latent representation. In case if authors are solving forecasting task using less regular structured data, they mainly focus on creating handcrafted features based on expert knowledge.

First subtask that authors have to solve when predicting further events is representation of these events. In case if data itself is time series it is not necessary and authors focus on research of forecasting approaches. Most common and well-studied approach for time series modeling is using methods like ARIMA for univariate time series or Vector ARIMA for multivariate ones [1]. Recently deep learning also has been applied to time series forecasting, mostly utilizing advantages of recurrent neural networks (RNN) and especially

Long short-term memory networks (LSTM) [2]. Laptev et al. incorporate LSTM ability of learning both long-term and short-term patterns while predicting completed taxi trips during periods such as holidays or weekends [3].

Recent area of deep learning research also includes approaches that use Convolutional neural networks together with RNN or LSTM networks. Shengdong et al. propose framework for traffic flow forecasting using the aforementioned combination of neural network architectures [4]. Guokun et al. also propose using this general idea in more common task of modeling temporal patterns in time series [5].

Tasks which include data more complex than time series require developing special representation for the data, or at least performing feature extraction in order to use well-studied forecasting approaches. It is important to notice that if features are handcrafted, it is not always possible to apply them to another task.

Gupta et al. use handcrafted features based on domain expertise together with historical features for predicting popularity trend of events in microblogging platforms such as Twitter [6] . Objects (events) considered in this work are short text messages. After extracting features, authors reduce prediction task to classification task with several classes representing future behaviour of popularity trend. Similar approach based on replacing prediction task with classification is presented in [7].

To incorporate text data Dey et al. classify text to one of predefined categories using LSTM and use classifier scores in order to obtain multivariate time series from texts [8]. After the processing is done, these numeric features could be easily concatenated with any other features, although this approach, especially categorizing text, is task-specific.

It is also important to consider anomaly detection task as a subtype of event forecasting. In case if our events to forecast are extremely rare, we could assume them to be anomalies and apply approaches used in anomaly detection field. Anomaly detection is well-studied area, including both classic methods, such as one-class classifiers and more recent ones, such as neural autoencoders [9], [10]. Ranjan considers the problem of predicting rare sheet break events for paper manufacturing [10]. Author proposes propose to train an autoencoder model on non-anomalistic data. After training is finished, author declares event as abnormal if the reconstruction error of the autoencoder is big enough. Main assumption is that distribution of anomalies differs from distribution of regular events.

III. Proposed approach

In order to handle complex structure of objects, we propose the approach, which consists of two main parts. The first one is aimed at transforming complex-structured objects to low-dimensional numeric latent space, so that we are able to aggregate latent representations of objects over the time. The second part of the approach allows us to perform event forecasting. This part aggregates latent features of events over the time, thus obtaining multivariate time series, and performs its further forecasting. General structure and main steps of the proposed approach are depicted in Fig. 1. Along with these parts we propose general technique that allows us to interpret latent representation.

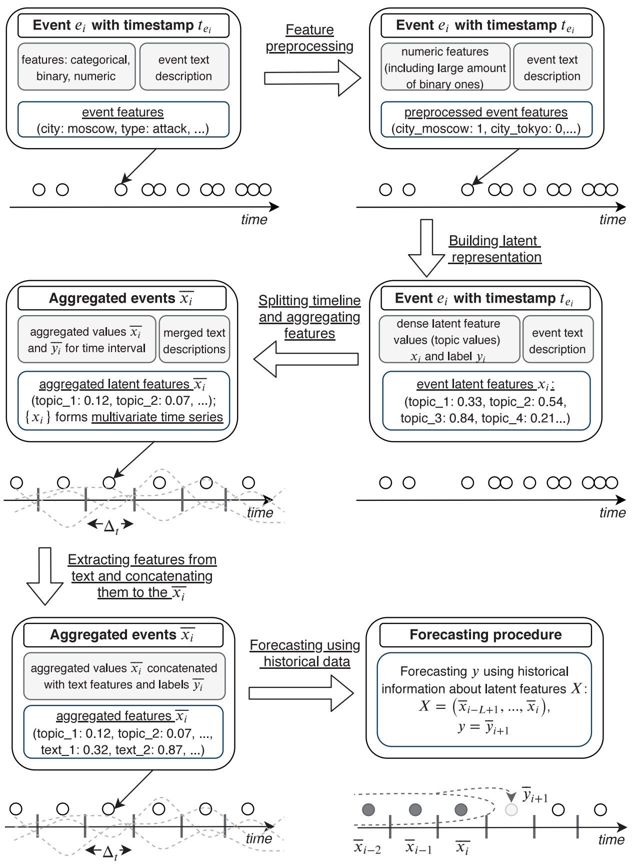
Event e_i with timestamp t_{e_i}
features: categorical, binary, numeric
event text description
event features (city: moscow, type: attack, ...)
time
Feature preprocessing
Event e_i with timestamp t_{e_i}
numeric features (including large amount of binary ones)
event text description
preprocessed event features (city_moscow: 1, city_tokyo: 0,...)
time
Building latent representation
Event e_i with timestamp t_{e_i}
dense latent feature values (topic values) x_i and label y_i
event text description
event latent features x_i: (topic_1: 0.33, topic_2: 0.54, topic_3: 0.84, topic_4: 0.21...)
time
Splitting timeline and aggregating features
Aggregated events $\overline{x_i}$
aggregated values $\overline{x_i}$ and $\overline{y_i}$ for time interval
merged text descriptions
aggregated latent features $\overline{x_i}$ (topic_1: 0.12, topic_2: 0.07, ...); $\{x_i\}$ forms multivariate time series
time
Δ_t
Extracting features from text and concatenating them to the $\overline{x_i}$
Aggregated events $\overline{x_i}$
aggregated values $\overline{x_i}$ concatenated with text features and labels $\overline{y_i}$
aggregated features $\overline{x_i}$ (topic_1: 0.12, topic_2: 0.07, ..., text_1: 0.32, text_2: 0.87, ...)
time
Δ_t
Forecasting using historical data
Forecasting procedure
Forecasting y using historical information about latent features X: $X = (\overline{x}_{i-L+1}, ..., \overline{x}_i)$, $y = \overline{y}_{i+1}$
$\overline{y}_{i+1}$
$\overline{x}_{i-2}$
$\overline{x}_{i-1}$
$\overline{x}_i$
time

Fig. 1. Structure of the proposed approach

Further in this section we describe specific parts of the approach in more details. Firstly we formulate developing latent event representation for complex events, and then discuss the interpretability of the proposed representation. Then we present an approach how to make any representation interpretable by given text descriptions for at least some of events. Finally, we introduce forecasting techniques which are applicable using any of the aforementioned event representations.

A. Building Latent Representation

Given the temporal irregularity of events, we require that it is possible to aggregate latent representation over arbitrary time intervals. Before building this latent representation, we preprocess features of the complex object in the following way. We keep numeric and binary feature as-is, each categorical feature is converted into one or more binary feature using one-hot encoding technique. Text descriptions can be transformed into numeric features in several ways such as tf-idf, LDA, word2vec, LSTM embeddings; in this work we choose Latent Dirichlet allocation. As a result, all kinds of features are transformed into numeric after such preprocessing.

It is important to mention, that after feature processing we obtain a set of numeric features (*initial features*), which is suitable representation itself. However, a lot of binary features generated with one-hot encoding technique are redundant because each of them corresponds to the specific value of the categorical feature. Categorical features (e.g. city or country) can take a tens or hundreds of different values, resulting in a large number of binary features. We assume that such dimensionality reduction approach as decomposition or autoencoder are capable of extracting meaningful topics from initial features obtained after preprocessing, simultaneously reducing dimension of initial feature space.

1) Matrix decomposition: Let n be the number of events and d be the number of features for each event. Let V be the original event description matrix, $V \in \mathbf{R}^{n \times d}$. Each row of the matrix V corresponds to some event and describes this event with d features. Decomposition task is defined as finding such matrices $W \in \mathbf{R}^{n \times k}, H \in \mathbf{R}^{k \times d}$ that minimize $||V - W \cdot H||$ for given V, $k \ll d$. It is important to notice that matrix H shows correspondence between original features and new latent features. Matrix W describes events in terms of new latent features. These latent features also known as *topics*, so further we call the matrix W the $event - topic$ matrix and H the $topic - feature$ matrix. As a baseline idea of developing suitable representation we use Non-negative matrix factorization (NMF) [11] and Principal Component Analysis (PCA) [12] dimensionality reduction methods. Both of them are able to find decomposition for matrix V, although the first one is easier to interpret, because its topics are essentially non-negative linear combinations of the original features.

NMF solves decomposition task in iterative fashion by non-negative initialization of W, H and updating both matrices H and W by the following rules:

$$H^{n+1}_{[i,j]} = H^{n}_{[i,j]} \frac{((W_n)^T V)_{[i,j]}}{((W^n)^T W^n H^n)_{i,j]}}$$

$$W^{n+1}_{[i,j]} = W^{n}_{[i,j]} \frac{(V(H^{n+1})^T)_{[i,j]}}{(W^n H^{n+1}(H^{n+1})^T)_{[i,j]}}$$

As soon as the process converges, we obtain matrices W, H.

To derive representation $z' \in \mathbf{R}^k$ for any event e' using NMF we should multiply its feature vector $x' \in \mathbf{R}^d$ by $H^T \in \mathbf{R}^{d \times k}$.

2) Autoencoder latent representation: Another way to find hidden features is to train an autoencoder neural network [13].

An autoencoder is a neural network architecture, where input and output layers have same dimensionality, with one or more internal (hidden) layers connecting them. The internal layer represents low-dimensional code of the input data, enforcing autoencoder to find reasonable hidden representation for the input instead of just remembering its input signal. Hidden representation that preserves most relevant features of the input allows to at least approximately reconstruct the input. Autoencoder consists of two parts: encoder, that performs mapping of the input to the code and decoder, that maps the code to the output, obtaining reconstruction of the input.

While solving our task of building latent representation, we use deep feedforward autoencoder with multiple layers that transforms an object $x \in \mathbf{R}^d$ into $z \in \mathbf{R}^k$ in a non-linear fashion and then attempts to reconstruct x from z. Given the reconstruction $x' \in \mathbf{R}^d$, autoencoder is trained to minimize reconstruction loss $L(x, x')$ (e.g. $L(x, x') = ||x - x'||$). The main goal of the autoencoder is to find an efficient encoding for set of objects and that makes autoencoder model suitable for our task of building latent representation.

After training neural autoencoder it is possible to use its first part (encoder) to obtain code z^* from an arbitrary input x^* and use z^* as a latent representation.

Non-linear input transformations leads to complex latent space, but in this case it is also quite complicated to find meaningful interpretation for each topic (i.e. for each element of z^*). We discuss this problem in III-B.

When using autoencoder to build latent space, we incorporate knowledge from anomaly detection field, using reconstruction error as an additional feature. Main assumption is that abnormal events fit poorly in input distribution, hence factorization or autoencoder will not reconstruct input accurate enough. Although reconstruction error is available in matrix decomposition as well, autoencoder is much more flexible because of its non-linearity and its reconstruction error is more meaningful.

B. Interpretability

As we mentioned above, important property of latent event representation is its interpretability. Given this property, it is possible to analyse forecasting results and understand data transformation better.

By its nature, NMF and PCA latent features are just linear combinations of the original event features. NMF by virtue of its non-negativity could be understood in the most natural way: bigger weights on certain original features mean bigger influence of these features to the corresponding topic. In case if feature weights do not have to be positive (as in PCA), negative contribution of feature to the topic is a bit harder to understand intuitively.

More formally, found latent features (topics) can be interpreted using matrix $H \in \mathbf{R}^{k \times d}$. To interpret i-th topic we select $K \in \mathbf{Z}$, $0 < K < d$ largest values from i-th row of matrix H and state that i-th topic depends on K corresponding features. Also we can measure individual contribution of the original j-th feature in i-th topic according value of $H_{[i,j]}$. In this way we obtain topic interpretation based on the contribution values of the original features.

When more complex approaches for building latent representation are used (e.g. neural autoencoders), it is not clear how to interpret latent features. Here we propose human-friendly way to do so by extracting keywords from at least partly available text descriptions, and then using them to interpret latent features. Description of this algorithm is provided below.

At the very beginning we select only events that contain text descriptions. Then we apply classic text preprocessing techniques to these descriptions such as conversion to lower case, removing punctuation and stop-words, stemming or normalizing form. After such preprocessing we obtain normalized keywords for each text description, then we compose dictionary W of size $|W|$ that is a union of all extracted keywords. Following step is to create $|W|$ (one per keyword) additional binary features for each event, representing presence or absence of the corresponding keyword in the event description. After building these features we calculate value $v_{i,j}$ for i-th topic T_i and j-th keyword binary feature vector B_j over all selected events. The value of $v_{i,j}$ is calculated as follows:

$$v_{i,j} = \frac{r_{i,j}}{\max\limits_{k \neq i} r_{k,j}}, \text{ where } r_{i,j} = 1 - \text{corr}(T_i, B_j)^2,$$

and it inversely proportional to how j-th keyword is important for i-th topic. Finally for each topic we use top-K keywords that have the smallest values $w_{i,j}$ for this topic as keyword interpretation of this topic. Details of the algorithm are shown in Algorithm 1. As shown in IV, even less than 20% events with text description is enough to get reasonable keywords and interpretive topics.

Algorithm 1 Topic interpretation algorithm

```
**Input.** Matrix with latent features for each event $T = \{t_{i,j}\}, \in \mathbf{R}^{k \times n}$, event text descriptions $d'_j$, number of desired keywords for each topic $K$
**Output.** set of $K$ keywords for each topic $\theta_i,\ i = 1..k$
initialize: $\theta_i = \varnothing,\ i = 1..k$;
**for** $j = 1$ **to** $n$ **do**
    $d_j \leftarrow preprocess(d'_j)$;
    $kw_j \leftarrow extract\_keywords(d_j)$;
**end for**
$W \leftarrow \bigcup_{j=1}^{n} kw_j,\ W = \{w_i\}$;
$B \leftarrow \{b_{i,j}\},\ b_{i,j} = \mathbf{I}_{w_i \in d_j},\ B \in \mathbf{R}^{|W| \times n}$;
**for** $i = 1$ **to** $k$ **do**
    **for** $j = 1$ **to** $|W|$ **do**
        $r_{i,j} \leftarrow 1 - \text{corr}(T_i, B_j)^2$;
        $v_{i,j} \leftarrow \frac{r_{i,j}}{\max_{k \neq i} r_{k,j}}$;
    **end for**
**end for**
**for** $k = 1$ **to** $K$ **do**
    $j = \arg\min_{\{j : w_j \notin \theta_i\}} v_{i,j}$
    $\theta_i \leftarrow$ add $w_j$
**end for**
```

This approach does not rely on any latent space structure, hence it is applicable for any hidden representation. And as auxiliary result we can obtain keyword description for any event as a combination of topic keyword descriptions if we select keywords from each topic description proportionally to the values of topics for the event.

C. Event Forecasting

Event forecasting task can be encountered when solving almost any forecasting problem. Usually, there are one of two most important sub-tasks is solved: forecasting probability of the event or forecasting number of events that will happen.

Naturally, the task of forecasting probability of the event during given time interval is a binary classification task with positive (event will happen, $y = 1$) and negative (event will not happen, $y = 0$) classes.

The task of forecasting number of events is a regression task, where the aim is to forecast value of the event counter ($y \in \mathbf{N}_0$).

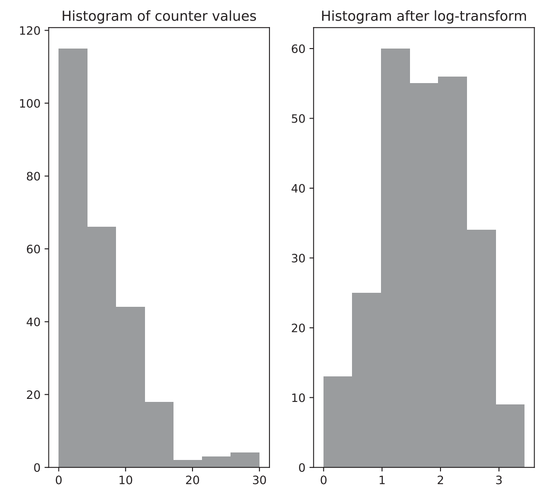

Fig. 2. Logarithmic transform of the target variable

In the regression task our target value is a counter, and in real world it has skewed distribution, which is often close to either Poisson or log-normal distribution. In order to reduce the skewness we propose using logarithm as a link function for the regression and forecast not the number of events y, but its logarithm $y' = \log(y + 1)$ instead.

Example of how log-transform affects the skewness of the target distribution is depicted in Fig. 2

Forecast of y' is $\mu = \mathbb{E}[\log(y + 1)]$, and the proper reconstruction of the $\mathbb{E}[y]$ is computed as follows [14]:

$$\mathbb{E}[y] = e^{\mu + \frac{\sigma^2}{2}} - 1,$$

where σ is a standard deviation of the y'. Estimate of the σ could be computed as a mean squared error over the validation set.

Let us define all events as E_{all} and events of interest (i.e. events to be forecasted) as E_{in}. First of all we should generate labels y_e for each event e:

$$y_e = \begin{cases} 1, & \text{if } e \in E_{in} \\ 0, & \text{otherwise} \end{cases}$$

Further reasoning is based on the assumption that each event e has the timestamp t_e and the set of latent features x_e that describes the e.

As soon as event representation is built, we obtain latent features for each event. These features are numerical, hence we can aggregate them over any time period, obtaining multivariate time series. Firstly, we choose aggregation time interval duration Δt, splitting timeline into intervals of size Δt. Then for each time interval T_i we get averaged latent feature values and aggregated label values of events:

$$\bar{x_i} = \frac{1}{|\{j : t_{e_j} \in T_i\}|} \sum_{\{j : t_{e_j} \in T_i\}} x_{e_j}$$

$$\bar{y}_i = \begin{cases} \sum\limits_{\{j:t_{e_j} \in T_i\}} y_{e_j}, & \text{for regression task} \\ \max\limits_{\{j:t_{e_j} \in T_i\}} y_{e_j}, & \text{for classification task} \end{cases}$$

To work with text descriptions when they are partly presented we can use two different approaches:

1) Calculate embeddings for each text description of the event e_i and then concatenate these embeddings with numeric features x_i
2) Calculate embeddings for merged text description from each time interval T_i and then concatenate it with aggregated numeric features $\bar{x}_i$

Due to sparsity in the text description presence, we use the second approach.

In order to add explicit autoregressive component we propose two additional numeric features, the total number of events A_{all_i} and number of events of interest A_{in_i} occurred during previous time interval. More formally, these features defined as follows:

$$A_{in_i} = \sum_{\{j:t_{e_j} \in T_{i-1}\}} y_{e_j},$$

$$A_{all_i} = |\{j : t_{e_j} \in T_{i-1}\}|$$

Such kind of features provide our model with additional historical information. Although these features can be called hand-crafted, they are not task-dependent, hence applicable in solving any other problem.

1) Problem formulation: After such procedure we have multivariate time series data, where $\bar{x}_0, \bar{x}_1, ...$ are observations. Since we forecast target variable given some historical data, we use L previous time intervals during predicting process. We formulate problem as forecasting $\hat{y}_{i+1}$ given $\bar{x}_{i-L+1}, \bar{x}_{i-L+2}, ..., \bar{x}_i$, where $\hat{y}_{i+1}$ is a predicted value of $\bar{y}_{i+1}$.

We propose two ideas of how to perform this forecast. First and baseline one is to concatenate features $\bar{x}_{i-L+1}, \bar{x}_{i-L+2}, ..., \bar{x}_i$ in a single vector, add features mentioned in III-A and III-C if desired and then use linear or tree-based machine learning model (e.g. Random Forest, Linear/Logistic Regression).

Second way is to build Long short-term memory neural network (LSTM) [15].

LSTM neural network architecture is based on the idea of Recurrent neural networks. In short, RNN is a class of neural networks used in sequence processing tasks. Their main advantage is loops in the architecture and reuse of its internal state. Due to these particular qualities, RNN networks can process sequences of variable length, but they struggle to handle long sequences due to exploding and vanishing gradients problems. It means that gradients that are back-propagated through time can vanish or explode, i.e. tend to zero or to infinity correspondingly. LSTM architecture was developed to overcome these problems, introducing *cell* and 3 gates: *input gate*, *output gate* and *forget gate*.

Cell in LSTM is a memory part, intuitively it allows to keep information about dependencies in sequence. The input gate controls the part of new sequence element information that we will consider, the output gate controls which information in the cell is used to compute the output and the forget gate controls which information in the cell will be erased (forgotten). LSTM block architecture allows gradients back-propagating through cell remaining unchanged, thus solving aforementioned gradient-related problems.

Usage of LSTM model differs from our baseline models in forecasting task. In LSTM, we sequentially pass feature vectors into the model starting from $\bar{x}_{i-L+1}$ to $\bar{x}_i$. Then we train the model in a way that its output $\hat{y}_{i+1}$ on final layer is as close as possible to the desired $\bar{y}_{i+1}$.

To perform qualitative evaluation of regression forecasts we use *mean absolute scaled error* (MASE) [16]. MASE is often applied in time series forecasting tasks because of the desirable features of this metric. Among them are invariance to the data scale, adequate behavior when $y \to 0$ and symmetry: penalties for underestimating and overestimating of y are equal.

Mean absolute scaled error metric is defined as follows:

$$MASE(y, \hat{y}) = \frac{MAE(y, \hat{y})}{\frac{1}{N-1} \sum\limits_{i=2}^{N} |y_i - y_{i-1}|},$$

$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|,$$

where y is ground truth event counter value and $\hat{y}$ is predicted counter value.

For the classification task we use *area under ROC curve* (ROC AUC) metric [17]. ROC AUC is invariant to the class imbalance and more representative than simple *accuracy* metric.

IV. Experiments

In this section, we use the real Global Terrorist Database for the evaluation of the proposed approach. We also provide comparison of different approaches of building latent representation and forecasting, including both classic models and novel neural network architectures.

A. Data description

For the experiments we use the Global Terrorism Database (GTD), it is the most comprehensive unclassified database of terrorist attacks in the world. It is an open-source database, which contains information on domestic and international terrorist events around the world since 1970, and now includes more than 190,000 events. For each event, a wide range of information is available, including the date and location of the incident, the weapons used, nature of the target, the number of casualties, and when identifiable the group or individual responsible. Details of the experimental dataset are described in Table I.

We select events that occurred between 2002 and 2008, size of aggregation time interval Δt equal to 3 days and define events of interest as Bombing/Explosion events that took place in Iraq. The size of aggregation time interval $\Delta t = 3$ was chosen empirically, but its value is highly dependent on the

TABLE I. DATASET DESCRIPTION

Dataset	Global Terrorism Database
Data type	Tabular
Attributes	country, city, attacktype1, nkill, nwound...
Attribute types	Numeric, categorical, binary, text
Time Span	01/01/1970 - 31/12/2015
Records	More than 190'000
Location	Worldwide

nature of the task. In case if events of interest are rare, it is reasonable to increase Δt. And if the goal is, for example, to forecast events for the next week, the value of Δt should be equal to 7.

Train/test split performed according to temporal structure of data, training part has first 70% of data and testing part has last 30%. The 70/30 proportion is a classic way to split data into training and testing sets, it provides enough data for training yet not too little amount of data for validation.

It is important to notice that we did not aim at the solving a particular problem with greatest possible accuracy, the main goal was to show that proposed general approach is applicable for solving forecasting task. Hyperparameters such as interval Δt, number of topics k and others were chosen empirically, and there is strong possibility of increasing forecasting accuracy with thorough selection of these hyperparameters.

In order to perform processing of the dataset with complex events, we do the following transformations:

- Replace every rarely-presented feature $attr$ (less than 5% of events have it) with special binary feature $has\_attr = 1$, if feature is presented in the event description and 0 otherwise.
- Use decision tree-based impute technique to fill missing features $nkill$ and $nwound$ based on known attributes.
- Replace categorical features using one-hot encoding technique
- Discard features we decided to be non-necessary or redundant (e.g. text decoding of such features as attack type id)
- Transform text descriptions into numeric vectors using Latent Dirichlet allocation.

After we obtain numeric features x_i for each event. Corresponding labels y_i are extracted as said in III-C.

B. Building latent representations

Further we denote number of events n, number of latent features k, number of original features d and event-feature matrix $V \in \mathbf{R}^{n \times d}$. After processing our dataset we had $n = 35834,\ d = 414$. In this work we evaluated performance using number of latent features (topics) k equal to 10 and 50. Both baseline and autoencoder approaches are firstly trained on training set, then using approach-specific way we derive latent representation for events from testing set.

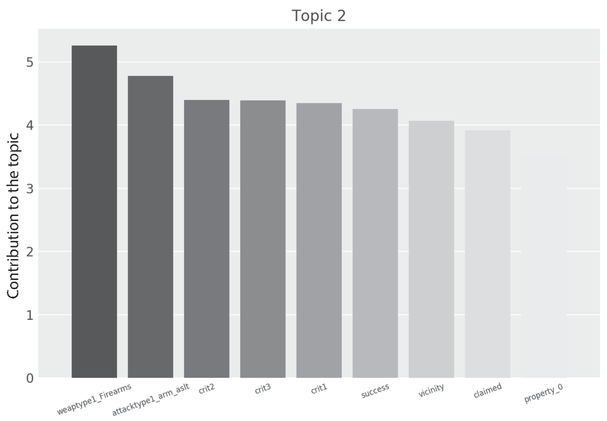

Fig. 3. Example of NMF topic interpretation

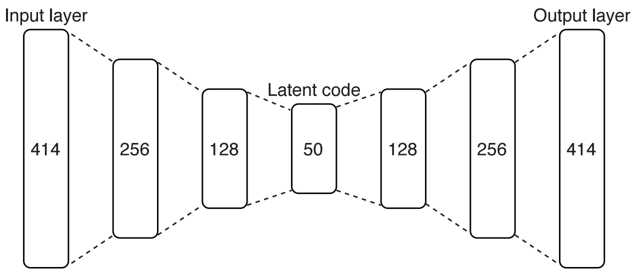

Fig. 4. Autoencoder architecture details

As a baseline approach we use non-negative matrix factorization. An example of how we could interpret NMF features using 9 larges feature contributions is depicted in Fig. 3. Along the abscissa axis original features are depicted. The ordinate values are obtained based on 2nd row of matrix H and they represent contribution of each of the original features to the topic (e.g. original feature weaptype1_Firearms is included in the linear combination (topic) with the largest coefficient, it means that this feature has the strongest influence on topic values).

Moving on to more complex methods of building latent representation, we describe autoencoder model. We built feed-forward deep autoencoder model using classic architecture with 3 layers in encoder and 3 layers in decoder. All of the activation functions are Rectified linear unit (ReLU), more detailed description of the autoencoder architecture is presented in Fig. 4. During training procedure we pass into autoencoder input feature vectors $x \in \mathbf{R}^d$, obtain their reconstruction $x' \in \mathbf{R}^d$ and minimize reconstruction mean absolute error (MAE) loss function:

$$MAE(x, x') = |x - x'|$$

Between encoder and decoder we obtain latent representation of the input signal also known as *code*: $z \in \mathbf{R}^k$. Training takes 100 epochs, i.e. 100 iterations over whole training set. Leaning rate is set to $2 \cdot 10^{-3}$, and exponential weight decay is applied as follows: after every epoch learning rate is multiplied by 0.96.

TABLE II. EXAMPLES OF KEYWORD EXTRACTED FOR LATENT TOPICS

Topic	Keywords
0	used, sources, available, majority, listed, database, ...
1	shot, civilian, distributed, bomb, detonated, ...
2	taliban, shrine, perpetrators, tamil, rebels, group, ...
3	released, hostage, kidnapping, status, abduction, ...
4	indicated, military, police, civilian, detonating, ...

C. Keyword topic description

From the training data 19335 events with text descriptions are selected. According to III-B, we obtain dictionary of size 2311 and a set of keywords for each of topics. An example of the acquired topic keywords is presented in Table II.

D. Forecasting

We use aggregation time interval Δt equal to 3 days, and after aggregating data according to III-C we get 569 time intervals in training set and 256 in testing one.

1) Forecasting probability of event: Probability of event occurring is a classification task with labels *will occur* ($y = 1$) and *will not occur* ($y = 0$). To solve this task we use both baseline machine learning classifier models such as RandomForest [18], LogisticRegression, and a novel LSTM neural network.

During experiments the following LSTM architecture is used: sequence length varies from 10 to 30, being exactly equal to L. Dimensions of the hidden layer and the memory cell are equal to dimensionality of latent space and vary from 10 to 50. We use softmax activation function applied to LSTM output values to get probability estimates. During training we use learning rate equal to 10^{-3} and number of epochs equal to 50.

Feature generation and classifier training are performed as said in III-C1, for baseline models we perform grid-search over the hyperparameters. Results of classifier performance depending on latent representation building technique are presented in Table III for experiments without text and in IV for experiments with text.

In the table III and all subsequent ones, 'NMF10' means that NMF feature extraction method is used with the number of topics $k = 10$, 'AE50' stands for autoencoder feature extraction approach with number of topics $k = 50$. '+A' means additional autoregressive features introduced in III-C, '+R' means additional reconstruction error feature introduced in III-A, '+AR' means that both autoregressive and reconstruction features are used. RFC stands for Random Forest Classifier, LogReg represents Logistic Regression and LSTM represents Long Short-term Memory network, described in III-C1. An example of classification model output is depicted in Fig. 5

2) Forecasting amount of events: To solve the task of forecasting amount of the events we use RandomForest (RFR), LinearRegression (LinReg), and LSTM models. For LSTM we use the same architecture as described in IV-D1, but instead of sigmoid activation after output layer we apply linear activation. We also follow the process described in III-C1, for baseline models we performed grid-search over hyperparameters.

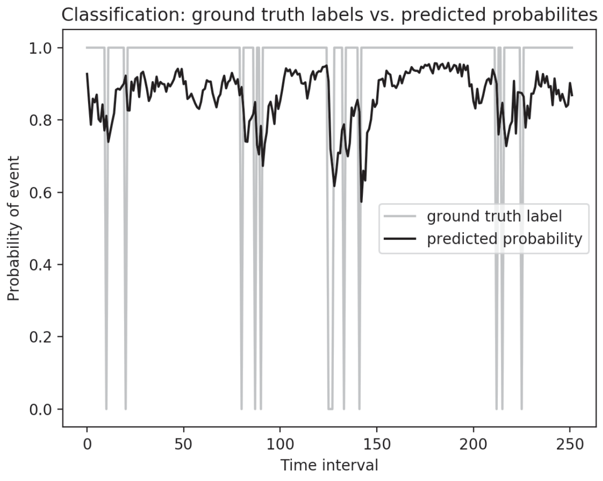

Fig. 5. Example of classification model performance

TABLE III. CLASSIFICATION: WITHOUT USING TEXT (ROC AUC)

	LSTM L=10	LSTM L=30	LogReg L=10	LogReg L=30	RFC L=10	RFC L=30
AE10	0.616	0.641	0.439	0.417	0.484	0.509
AE10+A	0.747	0.749	0.654	0.578	0.583	0.598
AE50	0.651	0.655	0.567	0.574	0.656	0.637
AE50+A	0.743	0.752	0.655	0.552	0.658	0.578
AE10+R	0.696	0.656	0.480	0.608	0.509	0.566
AE10+AR	0.764	0.744	0.653	0.572	0.609	0.615
AE50+R	0.547	0.544	0.605	0.588	0.443	0.623
AE50+AR	0.761	0.760	0.655	0.551	0.659	0.606
NMF10	0.618	0.605	0.577	0.526	0.637	0.606
NMF10+A	0.746	0.746	0.653	0.561	0.701	0.670
NMF50	0.691	0.711	0.537	0.446	0.650	0.533
NMF50+A	0.740	**0.783**	0.658	0.580	0.638	0.671

TABLE IV. CLASSIFICATION: USING TEXT (ROC AUC)

	LSTM L=10	LSTM L=30	LogReg L=10	LogReg L=30	RFC L=10	RFC L=30
AE10	0.597	0.657	0.505	0.563	0.679	0.361
AE10+A	0.764	0.747	0.642	0.599	0.673	0.665
AE50	0.655	0.678	0.523	0.550	0.624	0.631
AE50+A	0.752	0.745	0.645	0.493	0.681	0.614
AE10+R	0.608	0.628	0.523	0.564	0.424	0.474
AE10+AR	**0.795**	0.758	0.642	0.576	0.644	0.619
AE50+R	0.505	0.553	0.523	0.552	0.575	0.616
AE50+AR	0.736	0.759	0.644	0.523	0.674	0.504
NMF10	0.612	0.743	0.476	0.563	0.652	0.615
NMF10+A	0.601	0.753	0.644	0.544	0.674	0.567
NMF50	0.776	0.688	0.484	0.547	0.575	0.498
NMF50+A	0.741	0.753	0.645	0.569	0.688	0.625

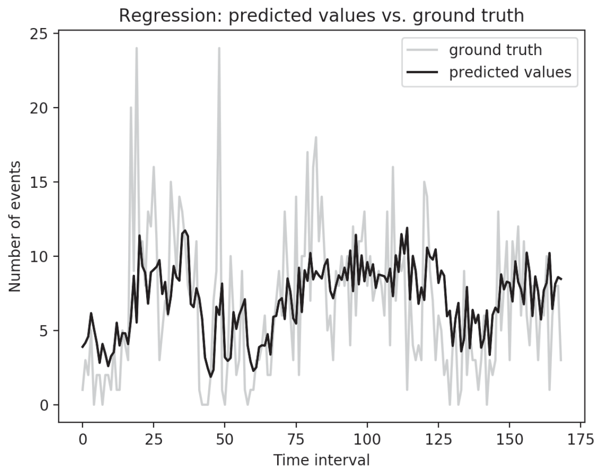

Fig. 6. Example of regression model performance

TABLE V. Regression: not using log-transform, not using text (MASE)

	LSTM L=10	LSTM L=30	LinReg L=10	LinReg L=30	RFR L=10	RFR L=30
AE10	1.218	1.204	1.119	1.191	1.241	1.265
AE10+A	1.050	1.050	0.909	1.074	0.890	0.909
AE10+R	1.092	1.096	1.121	1.193	1.239	1.287
AE10+AR	1.054	1.072	0.911	1.102	0.886	0.910
AE50	1.119	1.155	1.180	1.657	1.244	1.257
AE50+A	1.084	1.055	1.632	1.616	0.883	0.904
AE50+R	1.040	1.039	1.209	1.580	1.243	1.278
AE50+AR	1.059	1.056	1.632	1.758	**0.876**	0.916
NMF10	1.218	1.194	1.096	1.422	1.094	1.182
NMF10+A	1.056	1.069	0.895	1.177	0.894	0.929
NMF50	1.136	1.128	1.336	1.112	1.092	1.155
NMF50+A	1.076	1.061	1.468	1.181	0.913	0.912

Results of regressor performance depending on latent representation building technique are presented in table VIII for experiments without using text and in table VII for experiments with text. Additionally we compare performance of the regression without using logarithm as a link function, corresponding results are presented in tables V and VI.

Example of forecast obtained with regression model is presented in Fig. 6.

3) Performance analysis: As we can see, in classification task LSTM network leaves other models far behind. Introducing auxiliary features (+A, +AR) increases forecast performance for the classification task by up to 40%, depending on model. It means that autoregressive component is quite important for accurate forecasts. Usage of text descriptions allows us to achieve better result (+1.5%), but its applicability strongly depends on model type and latent representation.

Regression task results are a bit similar, but best-performing model is RandomForest Regressor instead of LSTM network. Again, auxiliary features heavily boosted preciseness of forecast (up to 30% depending on the model). In contrast with the classification result, text usage decreases forecast quality by 1.5-2%, which means that text information

TABLE VI. Regression: not using log-transform, using text (MASE)

	LSTM L=10	LSTM L=30	LinReg L=10	LinReg L=30	RFR L=10	RFR L=30
AE10	1.104	1.151	1.292	2.151	1.295	1.323
AE10+A	1.084	1.058	1.025	1.924	0.898	0.925
AE10+R	1.148	1.133	1.312	1.920	1.287	1.306
AE10+AR	1.047	1.067	1.029	1.745	0.896	0.929
AE50	1.148	1.120	3.492	1.862	1.300	1.307
AE50+A	1.046	1.068	3.082	1.810	0.899	0.918
AE50+R	1.124	1.136	2.840	1.967	1.305	1.307
AE50+AR	1.057	1.051	2.152	2.126	0.895	0.919
NMF10	1.171	1.068	1.241	2.921	1.171	1.187
NMF10+A	1.077	1.059	0.973	2.280	**0.894**	0.933
NMF50	1.168	1.108	3.534	1.212	1.123	1.148
NMF50+A	1.072	1.062	3.094	1.205	0.902	0.910

TABLE VII. Regression: using log-transform, not using text (MASE)

	LSTM L=10	LSTM L=30	LinReg L=10	LinReg L=30	RFR L=10	RFR L=30
AE10	1.128	1.100	1.017	1.020	1.139	1.141
AE10+A	1.054	1.014	1.028	1.729	0.848	0.875
AE50	1.059	1.035	1.127	3.075	1.073	1.105
AE50+A	1.017	1.012	1.171	6.938	0.836	0.886
AE10+R	1.105	1.068	1.046	1.087	1.135	1.151
AE10+R+A	1.031	1.022	1.006	1.869	**0.829**	0.872
AE50+R	1.052	0.979	1.141	5.021	1.107	1.116
AE50+R+A	1.072	1.024	8.402	9.042	0.834	0.872
NMF10	1.055	1.088	1.108	1.263	1.010	1.047
NMF10+A	1.025	1.032	1.008	1.582	0.862	0.898
NMF50	1.006	1.003	5.073	1.060	1.026	1.063
NMF50+A	1.070	1.015	9.072	1.354	0.856	0.916

TABLE VIII. Regression: using log-transform, using text (MASE)

	LSTM L=10	LSTM L=30	LinReg L=10	LinReg L=30	RFR L=10	RFR L=30
AE10	1.057	1.051	1.216	5.461	1.149	1.172
AE10+A	1.044	1.021	1.071	2.431	0.850	0.875
AE50	1.069	1.030	1.429	1.835	1.098	1.101
AE50+A	1.047	1.024	1.283	2.097	0.859	0.869
AE10+R	1.087	1.050	1.260	2.983	1.156	1.173
AE10+R+A	1.041	1.030	1.082	4.846	**0.843**	0.868
AE50+R	1.058	1.048	3.185	2.193	1.083	1.106
AE50+R+A	1.035	1.029	9.062	1.089	**0.843**	0.891
NMF10	1.085	1.121	1.283	2.061	1.015	1.075
NMF10+A	1.036	1.006	1.098	6.303	0.844	0.927
NMF50	1.114	1.175	5.817	1.112	1.056	1.060
NMF50+A	1.047	1.028	3.428	3.692	0.861	0.914

useful in some tasks and useless or even harmful in the others. Results with using logarithm as a link function are presented in tables V, VIII. Usage of this link function improves our forecast performance by 5-6%.

V. Conclusion and future work

In this paper, we proposed a general approach for event forecasting task, which is flexible enough to be adjusted with different forecasting models and different latent representation building techniques. Firstly, the latent representation for complex event features is built. Secondly, forecasting is performed for either classification or regression task. Since the approach is general, it is possible to choose desired methods on both steps from a variety of applicable methods. Finally, we present comprehensive comparison results in Tables III – VI. Additionally, we propose method to extract keywords for latent features using partly available event text descriptions. This addition is valuable if latent features have to be interpretive.

As a future research direction, we believe that joint training of autoencoder for latent representation and LSTM for forecasting can improve performance of this approach by enforcing autoencoder to extract latent features in a way that they are useful for further forecasting.

References

[1] M. Yasuko and S. Yasushi and F. Christos and I. Tomoharu and Y. Masatoshi, "Fast mining and forecasting of complex time-stamped events", *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2012.

[2] N. Tax and I. Verenich and M. La Rosa and M. Dumas, "Predictive business process monitoring with LSTM neural networks", *Lecture Notes in Computer Science*, 2017.

[3] N. Laptev and J. Yosinski and L. Erran and S. Smyl, "Time-series extreme event forecasting with neural networks at Uber", *Time-series Workshop*, 2017.

[4] S. Du and T. Li and X. Gong and Z. Yu and S. Horng, "A hybrid method for traffic flow forecasting using multimodal deep learning", *CoRR*, 2018.

[5] G. Lai and W. Chang and Y. Yang and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks", *CoRR*, 2017.

[6] G. Manish and G. Jing and Z. ChengXiang and H. Jiawei, "Predicting future popularity trend of events in microblogging platforms", *Proc. of the 75th Annual Meeting of the American Society for Information Science and Technology (ASIS & T)*, vol.49, Jan. 2012, pp. 1-10.

[7] J. Beunza and E. Puertas and E. Garca-Ovejero and G. Villalba and E. Condes and G. Koleva and et al., "Comparison of machine learning algorithms for clinical event prediction (risk of coronary heart disease)", *Journal of Biomedical Informatics*, vol. 97, 2019.

[8] D. Lipika and M. Hardik and V. Ishan, "Predictive analytics with structured and unstructured data - a deep learning based approach", *IEEE Intelligent Informatics Bulletin*, vol. 18, Dec. 2017.

[9] Extreme rare event classification: a straight forward solution for a real world dataset, Web: https://towardsdatascience.com/extreme-rare-event-classification-a-straight-forward-solution-58a20ef56ef5.

[10] Extreme rare event classification using autoencoders in Keras, Web: https://towardsdatascience.com/extreme-rare-event-classification-using-autoencoders-in-keras-a565b386f098

[11] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization", *Proc. of the 13th International Conference on Neural Information Processing Systems*, vol. 13, 2000, pp. 535541.

[12] S. Mishra and U. Sarkar and S. Taraphder and S. Datta and D. Swain and R. Saikhom et al., "Principal component analysis", *International Journal of Livestock Research*, Jan. 2017.

[13] W. Wang and Y. Huang and Y. Wang and L. Wang, "Generalized autoencoder: a neural network framework for dimensionality reduction", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2014, pp. 496-503.

[14] N. Johnson and S. Kotz and N. Balakrishnan, "Lognormal Distributions", *Continuous univariate distributions*, vol. 1, 1994.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol.9, Dec. 1997.

[16] R. Hyndman, "Another Look at Forecast Accuracy Metrics for Intermittent Demand", *The International Journal of Applied Forecasting*, vol. 4, 2006, pp. 43-46.

[17] A. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", *Pattern Recogn.*, vol. 30, July 1997, pp. 1145-1159.

[18] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, Oct. 2001, pp. 5-32.