Deep Representation Learning for Prediction of Temporal Event Sets in the Continuous Time Domain

Parag Dutta PARAGDUTTA@IISC.AC.IN
Kawin Mayilvaghanan KAWINM@IISC.AC.IN
Pratyaksha Sinha PRATYAKSHA1@IISC.AC.IN
Ambedkar Dukkipati AMBEDKAR@IISC.AC.IN
Department of Computer Science and Automation
Indian Institute of Science (IISc), Bangalore, KA, IN - 560012

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Temporal Point Processes (TPP) play an important role in predicting or forecasting events. Although these problems have been studied extensively, predicting multiple simultaneously occurring events can be challenging. For instance, more often than not, a patient gets admitted to a hospital with multiple conditions at a time. Similarly people buy more than one stock and multiple news breaks out at the same time. Moreover, these events do not occur at discrete time intervals, and forecasting event sets in the continuous time domain remains an open problem. Naïve approaches for extending the existing TPP models for solving this problem lead to dealing with an exponentially large number of events or ignoring set dependencies among events. In this work, we propose a scalable and efficient approach based on TPPs to solve this problem. Our proposed approach incorporates contextual event embeddings, temporal information, and domain features to model the temporal event sets. We demonstrate the effectiveness of our approach through extensive experiments on multiple datasets, showing that our model outperforms existing methods in terms of prediction metrics and computational efficiency. To the best of our knowledge, this is the first work that solves the problem of predicting event set intensities in the continuous time domain by using TPPs.

Keywords: Temporal Point Processes, Self-supervised learning, Forecasting, Events

1. Introduction

In today's complex and dynamic world, the need for accurate and reliable predictions is greater than ever. By making event predictions, we can identify potential risks and opportunities and take appropriate action to prepare for or capitalize on them. Event prediction problems have been studied in machine learning literature extensively, where the approaches range from sequence modeling to temporal point processes. Almost every approach deals with the problem of predicting a single event based on historical data. On the other hand, many practical problems require forecasting multiple events (a set of events), which inevitably requires us to model the distribution of such event sets over time because a simple prediction of whether an event set will occur is insufficient. In medical diagnosis, for instance, it is critical to know whether a particular condition is present and when it is likely to occur next so that preventive measures are taken accordingly (refer to Figure 1). Another example is trying to predict when and what set of items will a person check-out on an e-commerce website. Prior knowledge of the same can help reduce the shipment charges if the items are placed at convenient locations beforehand. Solution to event set prediction problem can provide valuable

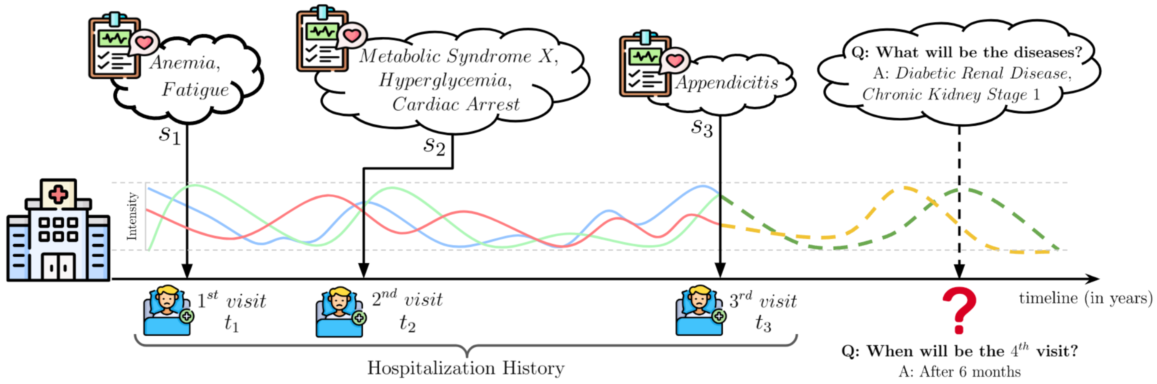

Figure 1: A typical temporal event set data sequence $\mathcal{S}$. Temporal Event set Modeling aims to predict both the event sets and the time of its occurrence given the corresponding history in the continuous time domain. For instance, as shown in the figure, given hospitalization history, we predict when and with what diseases/conditions the patient might be hospitalized in the future.

insights into the underlying patterns in the data and can be useful for identifying trends and making long-term predictions about the future.

Although multi-variate temporal event modeling has been explored before, these methods are rendered ineffective for modeling temporal event sets (Liniger, 2009; Mei and Eisner, 2017; Zuo et al., 2020). One may try to modify existing approaches for predicting temporal event sets. Considering all combinations of events as unique events can be one way to model the problem and still use the existing temporal event modeling approaches. However, the number of events increases exponentially and is impractical. An alternate approach is to decompose the event set into multiple singleton elements, assign the event set timestamp to each event of the event set individually, and then model them as regular temporal events. This approach, although tractable, does not consider the relations and dependencies among the events in the event sets.

In this paper, we propose a new approach based on deep representation learning that can resolve all the above-mentioned problems. Our *contributions* are as follows:

1. We propose a Contextual Self-Supervised Contrastive Learning objective for training an *Event-Encoder*, which learns representations of events in event sets.
2. We propose TESET, a Temporal Event set modeling framework that uses event set embeddings and combines them sequentially using transformer-based models.
3. We utilize intensity and temporal prediction heads to predict the intensity distribution of the event set along with the time of occurrence.
4. In our approach, we also facilitate using domain-specific features for learning better representations.

2. Related Works

Classic temporal event modeling works include Gaussian Processes (Ebden, 2015) and Multi-variate Hawkes Processes (Liniger, 2009), as mentioned earlier. To deal with the parametric kernels of the

Hawkes process, Mei and Eisner (2017) proposed the Neural Hawkes Process, which can use the expressive power of LSTMs to learn the intensities. Transformer Hawkes Process (Zuo et al., 2020) is another work that tries to use the computational efficiency of Transformers and the self-attention mechanism to solve RNNs' inability to learn long-term dependencies.

While the work of Choi et al. (2015) tries to model the patient's EHR, it does not consider the patient having multiple codes in the same visit. For the same task, (Choi et al., 2017) uses a graph-based model to show improvements over simple RNN-based methods (Choi et al., 2016). The work of Shang et al. (2019) uses a variant of the Masked Language Modelling (MLM) objective as a pre-training task. Similarly, for recommendations, (Kang and McAuley, 2018) uses attention-based models, and (Sun et al., 2019) uses an MLM.

BEHRT (Li et al., 2020) depicts diagnoses as words, visits as sentences, and a patient's entire medical history as a document to use multi-head self-attention, positional encoding, and MLM for EHR. Med-Bert (Rasmy et al., 2021) further adds to this concept by using serialization embeddings (order of codes within each visit using prior knowledge) besides code embeddings and positional (visit) encodings. Bert4Rec (Sun et al., 2019) uses an MLM-like bidirectional pre-training for predicting user-item interactions. Transformer4Rec (de Souza Pereira Moreira et al., 2021) further uses session information to enhance the previous works.

Recent works in set modeling as Sets2Sets (Hu and He, 2019) propose an encoder-decoder framework to predict event sets at discrete time steps, where the event set representation is obtained by aggregating the corresponding event embeddings by average pooling. DSNTSP (Sun et al., 2020) uses a transformer framework to learn item and set representations and captures temporal dependencies separately.

However, it must be noted that all the aforementioned methods either lack the ability to encode sets or they are applicable only for a discrete-time setting. To the best of our knowledge, the work proposed in this paper is the first to models event sets in the continuous time domain and solve the forecasting problem using TTPs.

3. Proposed Approach

We propose a two-step representation learning approach in Section 3.2 and 3.3 for modeling the temporal event sets. The pre-trained representation model thus obtained after the two steps of training can then be fine-tuned for the required downstream tasks.

3.1. Notations and Preliminaries

Let $\mathcal{S}$ denote an input sequence. Each element $\mathbf{s}_k \in \mathcal{S}$ corresponds to an event set and is ordered chronologically, where $k \in [|\mathcal{S}|]$ ($|\mathbf{x}|$ counts the number of elements in the set $\mathbf{x}$ and $[n]$ represents the set $\{1, 2, ..., n\}$). $\mathbf{s}_k \subset \mathcal{I}$ is a set of events, where $\mathcal{I}$ is the set of all possible events. Every $\mathbf{s}_k$ has an associated timestamp and (optionally) a set of features that we denote by $\mathbf{t}_k$ and $\mathbf{f}_k$ respectively. The features can be both static or dynamic; however, the feature set needs to be consistent across all the events. For instance, the age and weight of a patient change across hospital visits, whereas gender can be assumed to remain the same. $\mathcal{T} \subset \mathcal{I}$ is the target set. The target set is different from the input set of events. For instance, in the case of hospital visits, the target set may consist of only diagnoses, whereas the set of all possible events can additionally contain procedures and treatments.

We use $\mathcal{M}$ to denote the model being trained for a given task. $\mathcal{M}$ has a module called an encoder, denoted by $\mathcal{M}_E$, for encoding the input sequences along with a given set of features corresponding

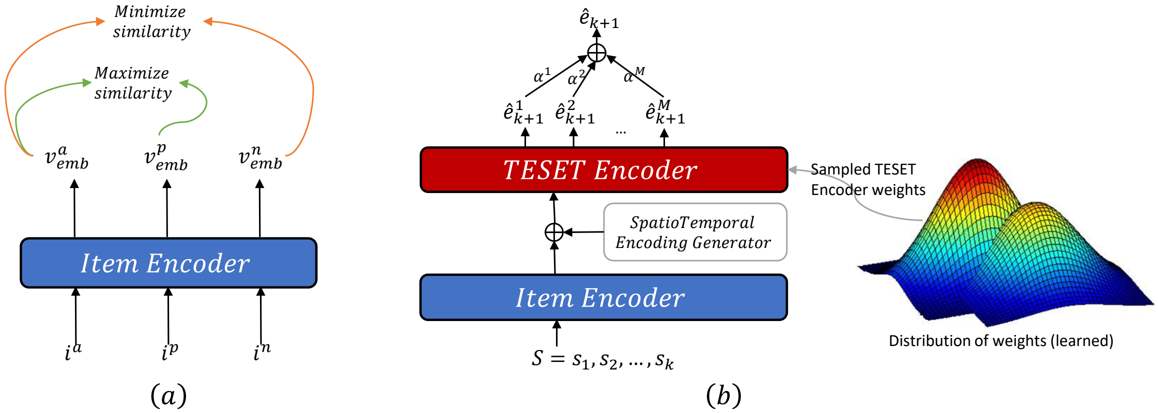

Figure 2: [Best viewed in color] Block diagram of our proposed approaches: (a) Learning event set representations, and (b) Inference procedure of our Bayesian Transformer based TESET model.

to each event in the sequence. We also use $\mathcal{A}_E$ to denote an auxiliary encoder model as described in Section 3.2. For instance, the auxiliary encoder $\mathcal{A}_E$ can be modeled using an affine layer, and $\mathcal{M}$ can be modeled using a transformer (Vaswani et al., 2017).

3.2. Learning Contextual Embeddings of Events

A measure of similarity among the vector representations of events $\mathbf{i} \in \mathcal{I}$ is required among the co-occurring events for learning meaningful contextual event embeddings. Consequently, in the first step of training, we use a self-supervised noise contrastive pre-training objective, similar to Noise Contrastive Estimation (Gutmann and Hyvärinen, 2010), for learning vector representations corresponding to every event in the set $\mathcal{I}$. The input to $\mathcal{A}_E$ is an event $\mathbf{i} \in \mathcal{I}$ and the output of $\mathcal{A}_E$ is $\mathbf{v}_{emb}$, which is a $\mathbf{d}_{emb}$–dimensional embedding vector.

To train this encoder network, we iterate over each $\mathbf{s}_k$ in $\mathcal{S}$, for all sequences in the dataset. At each iteration, we have the event set $\mathbf{s}_k$, which consists of a set of events. We sample two events $\mathbf{i}^a$ and $\mathbf{i}^p$ uniformly at random from $\mathbf{s}_k$, which becomes our anchor sample and positive sample respectively. We similarly sample our negative sample $\mathbf{i}^n$ uniformly at random from among the events that are not present in the set s_k. i.e.

$$\mathbf{i}^a \sim \mathbb{U}(\mathbf{s}_k); \mathbf{i}^p \sim \mathbb{U}(\mathbf{s}_k \backslash \{\mathbf{i}^a\}); \mathbf{i}^n \sim \mathbb{U}(\mathcal{I} \backslash \mathbf{s}_k) \tag{1}$$

where $\mathbb{U}(\cdot)$ denotes sampling uniformly at random from a given set of events. We then pass $\mathbf{i}^a$, $\mathbf{i}^p$, and $\mathbf{i}^n$ through $\mathcal{A}_E(\cdot)$ to obtain $\mathbf{v}^a_{emb}$, $\mathbf{v}^p_{emb}$, and $\mathbf{v}^n_{emb}$ respectively (as shown in Equation 3).

Then we calculate and maximize the following auxiliary contextual loss objective:

$$\mathcal{L}_{aux} = \log(\sigma(\mathbf{v}^a_{emb} \cdot \mathbf{v}^p_{emb})) + \log(1 - \sigma(\mathbf{v}^a_{emb} \cdot \mathbf{v}^n_{emb})) \tag{2}$$

where $\mathbf{a} \cdot \mathbf{b}$ represents the inner product among the vectors $\mathbf{a}$ and $\mathbf{b}$, and $\sigma(\mathbf{x}) = 1/1 + e^{-\mathbf{x}}$ represents the sigmoid function. Finally, the error is back-propagated through the auxiliary encoder model $\mathcal{A}_E$ and the corresponding parameters are updated using an appropriate optimizer.

Algorithm 1: Contextual self-supervised representation learning of events in event set

```
Data: $\mathcal{D}$;                                                          // Dataset
while not converged do
    $s_k \sim \mathcal{U}(\mathcal{D})$ ;                           // Sample event set from dataset
    $i^a \sim \mathcal{U}(s_k), i^p \sim \mathcal{U}(s_k \backslash \{i^a\}), i^n \sim \mathcal{U}(\mathcal{I} \backslash s_k)$;          // Sample events
    $v^a_{emb} = \mathcal{A}_E(i^a)$, where $i^a \sim \mathcal{U}(s_k)$ ;                      // Anchor
    $v^p_{emb} = \mathcal{A}_E(i^p)$, where $i^p \sim \mathcal{U}(s_k \backslash \{i^a\})$ ;              // Positive
    $v^n_{emb} = \mathcal{A}_E(i^n)$, where $i^n \sim \mathcal{U}(\mathcal{I} \backslash s_k)$ ;               // Negative
    Update $\mathcal{A}_E$ using $\nabla \mathcal{L}_{aux}(v^a_{emb}, v^p_{emb}, v^n_{emb})$ ;          // Backpropagate
end
Return: $\mathcal{A}_E$ ;      // Return the trained encoder (embedding generator)
```

3.3. Temporal Event set (TESET) Modeling

After the auxiliary encoder model $\mathcal{A}_E$ is trained, it can generate embeddings as follows:

$$\mathbf{v}_{emb} = \mathcal{A}_E(\mathbf{i}) \tag{3}$$

In the next step of training, we train the encoder module $\mathcal{M}_E$ in our model $\mathcal{M}$. For a given sequence $\mathcal{S}$, we assume the most recently occurred set of events to be $\mathbf{s}_k$. $\mathbf{s}_k$ also had associated $\mathbf{t}_k$ (a positive real value) as its corresponding timestamp, denoting when the event occurred in the timeline. Additionally $\mathbf{s}_k$ may also optionally contain an associated set of features $\mathbf{f}_k$.

All the previous set of events along with their corresponding timestamps and features until $\mathbf{s}_k$ is assumed to be history as follows:

$$\mathcal{H}_k = \{\langle \mathbf{s}_1, \mathbf{t}_1, \mathbf{f}_1 \rangle, \langle \mathbf{s}_2, \mathbf{t}_2, \mathbf{f}_2 \rangle, ..., \langle \mathbf{s}_{k-1}, \mathbf{t}_{k-1}, \mathbf{f}_{k-1} \rangle\} \tag{4}$$

We denote the target event set $\mathbf{e}_{k+1}$ as

$$\mathbf{e}_{k+1} = \mathbf{e}_{k+1} \cap \mathcal{T} \tag{5}$$

The objective in this step is to predict the tuple $\langle \mathbf{e}_{k+1}, \mathbf{t}_{k+1} \rangle$ given the tuple $\langle \mathbf{s}_k, \mathbf{t}_k, \mathbf{f}_k, \mathcal{H}_k \rangle$ as the input. In other words, the goal is to model the prediction of the set of next events along with the timestamp when it is supposed to occur given the most recent event, its timestamp, its associated features, and the entire history of events in that sequence of events.

Notice that the events $\mathbf{s}_k \in \mathcal{S}$ are sets of events. Hence, the events do not necessarily consist of only singleton elements and may contain two or more events. Consequently, we require $\mathcal{M}_E$ to be composed of a hierarchy of encoders: **(i)** Set Encoder that will combine the sets and give a single representation for all the events in the set, and **(ii)** Sequential Encoder that will take these combined representations as input and encode them temporally.

However, this approach possesses its own set of challenges as follows:

(a) Set encoding is a difficult problem since the set representations must satisfy properties such as permutation invariance and equivariance.

(b) During implementation, the event encoder either requires to be duplicated with one copy corresponding to every event set $s_1, ..., s_k$ or techniques such as gradient accumulation are needed while training. The duplication again requires high accelerator memory and efficient coding to utilize parallelization properly.

Algorithm 2: Representation learning of temporal event sets

```
Data: $\mathcal{D}$;                                                // Dataset
while not converged do
    $\mathcal{S} \sim \mathcal{U}(\mathcal{D})$ ;                   // Sample a sequence from dataset
    $k \sim \mathcal{U}([|S|])$ ;                                   // Sample an integer $1 \leq k \leq len(\mathcal{S})$
    $\langle \hat{\mathbf{e}}_{k+1}, \hat{\mathbf{t}}_{k+1} \rangle = \mathcal{M}_E(\langle s_k, t_k, f_k, \mathcal{H}_k \rangle)$ ;   // Forward pass
    Update $\mathcal{M}_E$ and $\mathcal{A}_E$ using $\nabla \mathcal{L}(\langle \hat{\mathbf{e}}_{k+1}, \hat{\mathbf{t}}_{k+1} \rangle, \langle \mathbf{e}_{k+1}, \mathbf{t}_{k+1} \rangle)$ ;   // Backpropagate
end
Return: $\mathcal{A}_E$ and $\mathcal{M}_E$ ;                       // Return the learned representation models
```

(c) The sequential nature of the event encoder prevents efficient parallelization, on top of it waiting for the event encoder to get the set representations.

As a solution to these problems, we propose a transformer-based architecture for training the model's encoder module $\mathcal{M}_E$. We stack all the events in the most recent occurred event $\mathbf{s}_k$ together with all the events $\mathbf{s}_1, ..., \mathbf{s}_{k-1}$ in history $\mathcal{H}_k$. Thus, assuming the events in each event set $\mathbf{s}_j$ are: $\mathbf{i}_j^1, \mathbf{i}_j^2, ..., \mathbf{i}_j^{|\mathbf{s}_j|}$, the current event set along with the history event sets in a given sequence $\mathcal{S}$ becomes

$$\mathcal{S}_k = \mathbf{i}_1^1, \mathbf{i}_1^2, ..., \mathbf{i}_1^{|\mathbf{s}_1|}, \mathbf{i}_2^1, \mathbf{i}_2^2, ..., \mathbf{i}_2^{|\mathbf{s}_2|}, ..., \mathbf{i}_k^1, \mathbf{i}_k^2, ..., \mathbf{i}_k^{|\mathbf{s}_k|} \tag{6}$$

In order to differentiate among the various event sets, we use the following techniques that are specifically applicable to a transformer-based architecture: (i) Special Tokens, and (ii) Custom SpatioTemporal Encodings containing both positional and temporal information.

Special Tokens: We use a special token, which is often referred to as the separator token (denoted by `[SEP]`) in the literature, after the event listed as $\mathbf{i}_j^{|\mathbf{s}_j|}$ for all $j \leq k$. This enables us to separate the event sets from each other whilst also providing us with a representation corresponding to each event set in the sequence $\mathcal{S}_k$. We use another classifier special token (denoted by `[CLS]`) at the very end of the sequence $\mathcal{S}_k$. This token helps us summarize the contents of the entire sequence, and its corresponding vector can be used for downstream tasks. We denote the resultant augmented sequence of events and tokens as $\mathcal{S}_k^*$. In the rest of this section, we will assume all the elements in the sequence $\mathcal{S}_k^*$ to be tokens to keep the discussion and notations uniform with the transformer literature. Hence the augmented sequence becomes

$$\mathcal{S}_k^* = \mathbf{i}_1^1, ..., \mathbf{i}_1^{|\mathbf{s}_1|}, \texttt{[SEP]}, \mathbf{i}_2^1, ..., \mathbf{i}_2^{|\mathbf{s}_1|}, \texttt{[SEP]}, ..., \texttt{[SEP]}, \mathbf{i}_k^1, ..., \mathbf{i}_k^{|\mathbf{s}_k|}, \texttt{[SEP]}, \texttt{[CLS]} \tag{7}$$

SpatioTemporal Embeddings: Next, we add custom Spatial and Temporal (SpatioTemporal) Encodings to all the events in the augmented sequence $\mathcal{S}_k^*$. The transformer framework assumes the entire sequence $\mathcal{S}_k$ as an atomic unit. It is therefore essential that we specify an encoding vector for each token in $\mathbf{s}_k$ such that it can not only enable the model to differentiate among various event sets in the sequence but also effectively accumulate contextual information in the respective output embeddings. Our requirement for encoding is different from positional encoding (Vaswani et al., 2017) due to the following reasons: **(i)** events $\mathbf{i}_j^1, \mathbf{i}_j^2, ..., \mathbf{i}_j^{|\mathbf{s}_j|}$ within a set $\mathbf{s}_j$ for all $j \leq k$ are unordered, unlike the ordered words in a textual sequence, and **(ii)** two consecutive timesteps are not uniformly separated in the timeline. For instance, the duration between the current and next visit of a patient might vary from as short as a month to as long as multiple years.

Consequently, we use the SpatioTemporal Encodings as described below, which can handle these non-uniform temporal differences whilst also retaining information about the co-occurrence of events in event sets.

$$\mathbf{v}^{pos}_{enc}(j,d) = \begin{cases} \sin\left(j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & \texttt{;if } j \texttt{ is even} \\ \cos\left(j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & \texttt{;otherwise} \end{cases}$$

$$\mathbf{v}^{temp}_{enc}(\mathbf{t}_j,d) = \begin{cases} \sin\left(\mathbf{t}_j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & \texttt{;if } \mathbf{t}_j \texttt{ is even} \\ \cos\left(\mathbf{t}_j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & \texttt{;otherwise} \end{cases}$$

$$\mathbf{v}_{enc}(j,\mathbf{t}_j,d) = \mathbf{v}^{pos}_{enc}(j,d) + \mathbf{v}^{temp}_{enc}(\mathbf{t}_j,d) \tag{8}$$

where $\mathbf{t}_j$ is the timestamp corresponding to the j^{th} event set $\mathbf{s}_j$ for $1 \leq j \leq k$.

Note that the initial value of $\mathbf{v}_{emb}$ is obtained by passing each event from event sets through $\mathcal{A}_E$. Then we add $\mathbf{v}_{enc}$ to $\mathbf{v}_{emb}$ before passing it on to the Transformer model. We get a $\mathbf{d}_{emb}$–dimensional embedding vector $\mathbf{v}^{\texttt{[CLS]}}_{emb}$ corresponding to the `[CLS]` token. We denote this output vector by $\mathbf{v}_{out}$. Refer to Figure 2 for a block diagram of our approach.

Additionally, we use the following two prediction heads for training the representation model $\mathcal{M}$: **(i)** Event set Prediction Head, denoted by $\mathcal{P}_E$, and **(ii)** Temporal Prediction Head, denoted by $\mathcal{P}_T$. $\mathcal{P}_E$ takes $\mathbf{v}_{out}$ as input and predicts M pairs of Gaussian distributional parameter vectors $\langle \mu^1_{\hat{\mathbf{e}}_{k+1}}, \sigma^1_{\hat{\mathbf{e}}_{k+1}} \rangle$, $\langle \mu^2_{\hat{\mathbf{e}}_{k+1}}, \sigma^2_{\hat{\mathbf{e}}_{k+1}} \rangle$, ..., $\langle \mu^M_{\hat{\mathbf{e}}_{k+1}}, \sigma^M_{\hat{\mathbf{e}}_{k+1}} \rangle$ along with M mixing coefficients $\alpha^1_{\hat{\mathbf{e}}_{k+1}}, \alpha^2_{\hat{\mathbf{e}}_{k+1}}, ..., \alpha^M_{\hat{\mathbf{e}}_{k+1}}$. The M event set prediction vectors $\hat{\mathbf{e}}^1_{k+1}, \hat{\mathbf{e}}^2_{k+1}, ..., \hat{\mathbf{e}}^M_{k+1}$ are sampled from the distribution parameters with the same number of dimensions as events in the target set $\mathcal{T}$ with each dimension modeling a Bernoulli distribution. After mixing the sampled vectors according to the mixing coefficients, we get

$$\hat{\mathbf{e}}_{k+1} = \alpha^1_{\hat{\mathbf{e}}_{k+1}} \cdot \hat{\mathbf{e}}^1_{k+1}, \alpha^2_{\hat{\mathbf{e}}_{k+1}} \cdot \hat{\mathbf{e}}^2_{k+1}, ..., \alpha^M_{\hat{\mathbf{e}}_{k+1}} \cdot \hat{\mathbf{e}}^M_{k+1} \tag{9}$$

Similarly, $\mathcal{P}_T$ takes v_{out} as input and outputs the M temporal Gaussian distributional parameter pairs $\langle \mu^1_{\hat{\mathbf{t}}_{k+1}}, \sigma^1_{\hat{\mathbf{t}}_{k+1}} \rangle$, $\langle \mu^2_{\hat{\mathbf{t}}_{k+1}}, \sigma^2_{\hat{\mathbf{t}}_{k+1}} \rangle$, ..., $\langle \mu^M_{\hat{\mathbf{t}}_{k+1}}, \sigma^M_{\hat{\mathbf{t}}_{k+1}} \rangle$ along with M temporal mixing coefficients $\alpha^1_{\hat{\mathbf{t}}_{k+1}}, \alpha^2_{\hat{\mathbf{t}}_{k+1}}, ..., \alpha^M_{\hat{\mathbf{t}}_{k+1}}$. We sample scalars $\hat{\mathbf{t}}^1_{k+1}, \hat{\mathbf{t}}^2_{k+1}, ..., \hat{\mathbf{t}}^M_{k+1}$ and similar to Equation 9, we obtain $\hat{\mathbf{t}}_{k+1}$ by mixing them according to the mixing coefficients. We use reparametrization (similar to Kingma and Welling (2014)) to enable backpropagation in our model.

Temporal Event set Modeling: In order to learn representations in the second step of pre-training, we propose the Temporal Event set Modeling objective as described below. Upon sampling the tuple $\langle \hat{\mathbf{e}}_{k+1}, \hat{\mathbf{t}}_{k+1} \rangle$ corresponding to an input $\langle \mathbf{s}_k, \mathbf{t}_k, \mathbf{f}_k, \mathcal{H}_k \rangle$, we use the following loss objectives as a part of our TESET Modeling:

(i) An element-wise binary cross-entropy loss on $\hat{\mathbf{e}}_{k+1}$ against $\mathbf{e}_{k+1}$:

$$\mathcal{L}^{BCE}_{Event} = \frac{1}{|\mathcal{T}|} \sum_{d \in [|\mathcal{T}|]} \mathbb{1}_{\{\mathcal{T}^{(d)} \in \mathbf{e}_{k+1}\}} \hat{\mathbf{e}}^{(d)}_{k+1} + \mathbb{1}_{\{\mathcal{T}^{(d)} \notin \mathbf{e}_{k+1}\}} (1 - \hat{\mathbf{e}}^{(d)}_{k+1}) \tag{10}$$

where $\mathbb{1}$ denotes the indicator function and $\mathbf{v}^{(d)}$ represents the d^{th} dimension of the vector $\mathbf{v}$.

(ii) Additionally, we use dice loss for handling the class imbalance problem in $s_{k+1} \cap \mathcal{T}$.

$$\mathcal{L}^{Dice}_{Event} = 1 - \frac{1}{|\mathcal{T}|} \sum_{d \in [|\mathcal{T}|]} \frac{2\, \hat{\mathbf{e}}^{(d)}_{k+1}\, \mathbf{e}^{(d)}_{k+1} + \epsilon}{\sum_{d' \in [|\mathcal{T}|]} \hat{\mathbf{e}}^{(d')}_{k+1} + \mathbf{e}^{(d')}_{k+1} + \epsilon} \tag{11}$$

Table 1: **Temporal Event set Modeling Results.** We compare our approaches to baselines. For DSC, the larger the better; for MAE, the smaller the better. We can see that even without Contextual embeddings, our methods outperform the baselines. Best results are in bold

Training method	Synthea		Instacart	
	Event set pred (DSC)	Time pred (MAE)	Event set pred (DSC)	Time pred (MAE)
Baselines:				
Neural Hawkes Process	0.08	2.50	0.29	0.24
Transformer Hawkes Process	0.18	2.41	0.32	0.24
Hierarchical Model	0.12	2.51	0.30	0.23
Ours:				
TESET	0.20	2.29	0.35	0.21
TESET + Contextual Embeddings	**0.30**	**2.17**	**0.42**	**0.18**

where ϵ is a small Laplace Smoothening constant.

(iii) Finally, for the timestamp prediction head, we use Huber loss to align $\hat{\mathbf{t}}_{k+1}$ and $\mathbf{t}_{k+1}$:

$$\mathcal{L}_{Temporal}^{Huber} = \begin{cases} \Delta^2/2 & \text{; if } \Delta < \delta \\ \delta(\Delta - \delta/2) & \text{; otherwise} \end{cases} \tag{12}$$

where Δ is the absolute value of $\hat{\mathbf{t}}_{k+1} - \mathbf{t}_{k+1}$ and δ is a positive constant.

We minimize a linear combination of the above loss objectives as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{Event}^{BCE} + \lambda_2 \mathcal{L}_{Event}^{Dice} + \lambda_3 \mathcal{L}_{Temporal}^{Huber} \tag{13}$$

where, $\lambda_1, \lambda_2, \lambda_3 > 0$. We calculate this loss and back-propagate the gradients through our encoder models $\mathcal{M}_E$ and $\mathcal{A}_E$ and update the model parameters accordingly.

Multiple Generations: We implement our Transformer model $\mathcal{M}$ using the Probabilistic Bayesian neural network framework. Essentially, every time we train the model we sample the weights and biases (for every layer) from a weight distribution, and then update the distribution through backpropagation. This enables us to sample the weights multiple times, thus providing us with an ensemble of N networks whose predictions we combine to get our final predicted outputs. This stabilizes the training, helps converge the loss objective faster, and the validation metrics improve noticeably during the initial stages of training (see Section 5 for more details).

4. Experiments[1]

4.1. Datasets

1. **Synthea:** (Walonoski et al., 2017) encompasses the comprehensive medical records of each patient generated synthetically. The medical history of each patient is represented as a

[1] Codes for our experiments are available at: https://github.com/paragduttaiisc/temporal_event_set_modeling

Table 2: **Fine-tuning results.** FT stands 'fine-tuned'. Our models consistently outperform the baselines in the setting of being fine-tuned for the downstream tasks. It should be noted that Fine tuning doesn't work on the baseline models, and they often perform worse. In each stratum, the best-performing models have been italicized. The best-performing models have been shown in bold.

FT?	Training method	Synthea		Instacart	
		Event set given time (DSC)	Time given event (MAE)	Event set given time (DSC)	Time given event (MAE)
Trained from scratch	Neural Hawkes Process	0.21	5.70	0.35	2.19
	Transformer Hawkes Process	0.20	4.52	0.34	2.15
	Hierarchical Model	0.19	5.29	0.34	2.20
	TESET (Ours)	*0.22*	*4.28*	*0.38*	*1.83*
Fine-tuned	Neural Hawkes Process	0.13	6.01	0.30	2.29
	Transformer Hawkes Process	0.19	4.60	0.33	2.24
	Hierarchical Model	0.18	5.87	0.35	2.31
	TESET (Ours)	***0.25***	***3.91***	***0.41***	***1.19***

sequential sequence of their hospital visits, along with the corresponding timestamps denoting the time of each visit. Each hospital visit comprises an event set, containing the diagnoses, treatments, and procedures administered during that particular visit, in addition to the patient's characteristics such as age, weight, and gender.

2. **Instacart:** (Instacart, 2017) is a comprehensive collection of customers' order histories. Each individual customer's order history is represented as a sequential arrangement of orders, wherein each order includes a set of items purchased by the respective customer and the corresponding timestamp indicating the time of the order.
3. **MIMIC-III:** (Johnson et al., 2018) provided includes the historical data of patients who have visited the Intensive Care Unit (ICU) at a hospital. By analyzing the Electronic Health Record (EHR) history of each patient, we extract the sequential information regarding the set of medical conditions diagnosed and the respective admission timestamps. For the purpose of the finetuning task, the medical codes used in the Synthea dataset are correspondingly mapped to the medical codes employed in the MIMIC-III dataset.

4.2. Baselines

We quantify the advantage of our proposed approach by comparing with the following competitive baselines[2]:

1. **Neural Hawkes Process (NHP):** The NHP (Mei and Eisner, 2017) employs a Recurrent Neural Network, specifically a continuous-LSTM, to parameterize the intensity function λ of the Hawkes process. The intensity function is K-dimensional, denoted as $\lambda_k(t) = f_k(w_k^T h(t);$

[2] Baselines 1 and 2 models are originally used for event prediction given a sequence of events and Baseline 3 is originally used in the discrete timestep set prediction. We extend them to predict sets in the continuous domain.

Table 3: **Transfer learning results**. We fine-tune the TEM model trained on the synthetic dataset using the MIMIC-III dataset. It is observable that syntheic to real transfer works better for our approach.

Model	Event set (DSC)	Time (MAE)	Event set given time (DSC)	Time given event (MAE)
TESET trained from scratch	0.49	0.67	0.47	0.70
TESET fine-tuned	*0.52*	*0.14*	*0.50*	*0.19*

where, $f_k(.)$ is the decay function δ that is chosen to be the softplus function, K is the number of events, and $h(t)$ is the hidden state of the LSTM.

2. **Transformer Hawkes Process (THP):** The THP (Zuo et al., 2020) utilizes a self-attention mechanism and temporal encoding to model the Hawkes process. This approach effectively captures long-term dependencies while maintaining computational efficiency, distinguishing it from the NHP (Neural Hawkes Process).
3. **Hierarchical Model (HM):** HM uses a hierarchical encoder where in the first step it encodes the sets and provides a representation for each set using a pooling function and in the second step it encodes the set representations temporally. We use a fully connected neural network to encode the sets and a Bi-LSTM model for encoding the set representations temporally. This is in similar lines to the Sets2Sets model (Hu and He, 2019).

4.3. Downstream tasks

We demonstrate the superiority of the representations learned by our TESET model by fine-tuning on the following downstream tasks:

1. **Event set prediction given time:** In this downstream task, the idea is to model $\mathbf{s}_{k+1}$ given the tuple $(\mathbf{s}_k, \mathbf{t}_k, \mathbf{f}_k, \mathcal{H}_k, \mathbf{t}_{k+1})$. In other words, we would like to predict the event set that might occur in the future given the future timestamp in addition to the tuple of the most recent event, its timestamp, its associated features, and the entire history of events-sets in that sequence as mentioned in Section 3. Note that the future timestamp when we want to predict the most probable event set lies in the continuous domain.
2. **Temporal prediction given an event:** Conversely, in this downstream task, the idea is to model $\mathbf{t}_{k+1}$ given the tuple $(\mathbf{s}_k, \mathbf{t}_k, \mathbf{f}_k, \mathcal{H}_k, \mathbf{i})$, where $\mathbf{i} \in \mathcal{T}$. In other words, we would like to predict the most probable time when a particular event might occur in the future given the event from the target

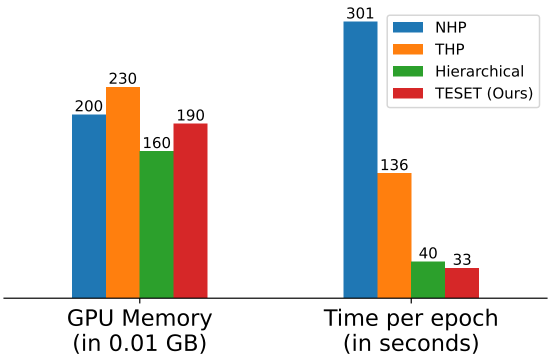

Figure 3: [Best viewed in color] Resource usage and training time comparison during TEM training on Synthea dataset. The TESET model is the fastest although it has similar computational requirements.

Table 4: **SpatioTemporal Encodings**: Need for custom encoding during TEM is evident from the considerable advantage we observe. Models were trained on the Instacart dataset.

Transformer Encoding	Event set pred. (DSC)	Time pred. (MAE)
Positional Enc (Vaswani et al., 2017)	0.35	0.22
SpatioTemporal Enc (Ours)	**0.42**	**0.18**

set in addition to the tuple of a most recent event, its timestamp, its associated features, and the entire history of events in that sequence of events.

4.4. Ablation Studies

We formulate our ablation experiments in the form of the following interesting research questions:

RQ-1: Are the representations learned by TEM useful for related tasks?

RQ-2: What is the role of incorporating additional features into our framework?

RQ-3: How effective is the Contextual Event Representation Learning step?

RQ-4: How likely are our approaches to adapt and generalize with respect to a domain shift?

RQ-5: Can the advantage of using SpatioTemporal encodings over conventional positional encodings be quantified?

RQ-6: What is the training time saved by considering the set of temporal data points rather than each event individually?

RQ-7: Is the Bayesian Transformer with the distributional heads even required?

RQ-8: Do the predicted event intensities for a given history sequence correspond to something meaningful?

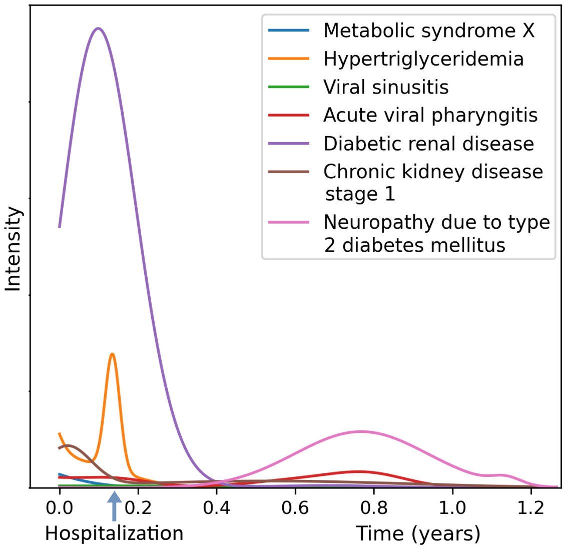

Figure 4: [Best viewed in color] Our model's predicted intensity plot for an elderly female patient who had a history of diabetes. The peak of two high-intensity disease curves (Diabetic Renal Disease and Hypertriglyceridemia) coincides with the date of actual hospitalization with those conditions. Also, it is predicted that Neuropathy might be a problem in the future, which is a well-known condition for people suffering from Type II Diabetes.

5. Results

Table 1 compares the performance of our proposed models with baselines under similar settings on two different tasks and two datasets:

(i) It is evident that our TESET model outperforms existing baselines in both event set and temporal prediction metrics. We achieve 0.12 and 0.10 DSC improvement (absolute metrics) in Synthea and Instamart datasets respectively for the event set prediction sub-task.

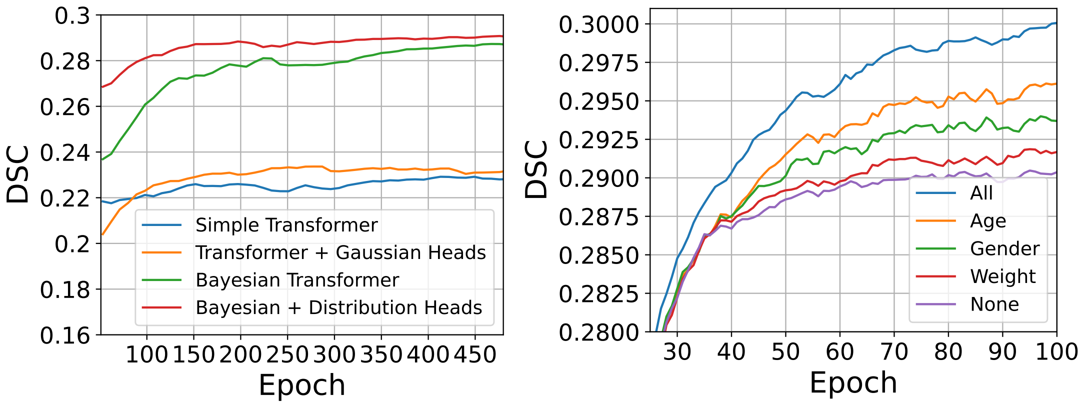

Figure 5: [Best viewed in color] The plot on the **left** compares the test set dice scores for the TESET variants. Bayesian Transformer extends Simple Transformer with the Probabilistic Bayesian NN framework, while the Transformer with Gaussian Heads predict single Gaussian distribution at the final layer. It is clearly observable that Bayesian Transformer with Distributional Heads (Ours) is more stable and performs well right from the start. The plot on the **right** plots the test set dice scores of the TESET model with a combination of various features. It can be noticed that using all the features has considerable advantage.

We also achieve 0.34 and 0.05 absolute improvement in MAE in the same datasets for the time prediction sub-task.

(ii) We can quantify **RQ-3** by looking at the difference of metrics with and without using contextual embeddings. It can be noticed that using contextual embedding is clearly advantageous.

Figure 6 additionally shows a magnified t-SNE plot of the Contextual vectors in the representation space to demonstrate the clustering of similar items.

Table 2 compares the fine-tuning results for the event set given time and time given event downstream task:

(i) It can be noticed from the table that our methods outperform the baselines when fine-tuned instead of being trained from scratch.

(ii) It is again evident that our model (TESET) learns representations during TEM that can be used for downstream tasks, thus answering **RQ-1**. On the other hand, the baseline approaches learn representations that are not generalizable for downstream tasks, and hence they perform better (compared to themselves) when trained from scratch.

Figure 5 answers **RQ-2**. We can attribute the consistently improved performance of the model throughout the TEM training to the use of domain-specific features such as age, weight and gender. When trained with all of the features together, the model achieves the best performance.

Table 3 presents the domain generalization capabilities of the representations learned by our TEM model, answering **RQ-4**. We can see that even though our TEM model was trained on the Synthea dataset, it generalizes quite effectively
to the real-world dataset MIMIC-III.

Table 4 answers **RQ-5** by showing that our SpatioTemporal Encodings definitely score higher metrics in both event set prediction and time prediction during TEM when compared to vanilla Positional Embeddings.

Table 5: **Time and Computational Complexity.** An analysis of the computational and time complexity for each layer of the baseline methods and our method. The notations are as follows: T indicates the Sequence Length, μ_E indicates the Average Event-Set Length (average number of items in the event-sets), and d indicates the Embedding (hidden) dimension.

	NHP	THP	HM	TESET (Ours)
Computational Complexity	$\mathcal{O}(T \cdot \mu_E \cdot d^2)$	$\mathcal{O}(T^2 \cdot \mu_E^2 \cdot d)$	$\mathcal{O}(T \cdot d)$	$\mathcal{O}(T^2 \cdot d)$
Time Complexity	$\mathcal{O}(T \cdot \mu_E)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

We answer **RQ-6** by observing Figure 3. It can be observed that the NHP and THP are $10\times$ and $4\times$ slower compared to our TESET model. Even the Hierarchial approach is $1.3\times$ slower. We additionally present asymptotic computation and time complexity analysis in Table 5.

From Figure 5, we can compare the training plots for the following models (i) simple transformer, (ii) transformer with distributional heads, (iii) Bayesian transformer with distributional heads. The considerable advantage of the model (iii) is clearly visible from the plots, thus answering **RQ-7**.

Finally, from Figure 4, we can see that the predicted intensities of various correlated diseases are shown to be high in the future. The peak of the curves coincides with the next hospitalization date in the dataset. Thus, not only is **RQ-8** answered by meaningful predictions, but also the hospitalization can be prevented with precautionary checkup before the predicted date of hospitalization.

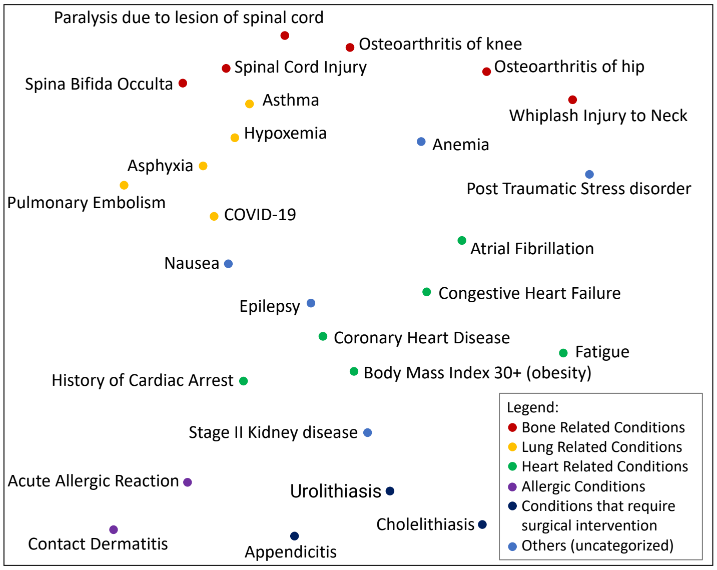

Figure 6: [Best viewed in color] 2D t-SNE embeddings of the representations learned after first step of our approach in Synthea Dataset. It can be observed that clusters are formed in the embedding space.

6. Limitations and Future Works

Our method is limited to representing the relationship among items (such as diseases and treatments) as distances amongst embeddings in the representation space. Consequently, our method can only capture pairwise relationships between items but not more complex relationships such as transitive or hierarchical. One natural extension of our work would be to infuse external knowledge from a knowledge graph. From the entities and their relationships, it might be possible to capture more complex relationships among items and additional information such as their attributes and side effects.

Another limitation of our method is that it only predicts the items in the set themselves, and not how the set should be used in a decision-making context. For example, if we are predicting diseases and treatments, our method cannot tell us which treatment is best for a particular patient. Thus, another ambitious future direction for our work would be to extend our work for decision making, for instance by learning decision-making strategies that tell us which treatment to give to a patient at each time step, based on the patient's current state and the history of treatments that they have received. This is often called the dynamic treatment regime, and extending our work in this domain would make it more useful in real-world applications.

7. Conclusion

In this paper, we propose a method for modeling the temporal event set distribution. We additionally learn self-supervised contextual event embeddings and incorporate temporal and domain-specific features into the framework to generate better representations. We also provide a Transformer based approach along with SpatioTemporal Encodings to model the same. We empirically demonstrate the validity of our methods along with the necessity of the various components of our proposed methods through appropriate experiments.

Acknowledgement

The authors would like to thank the SERB, Department of Science and Technology, Government of India, for the generous funding towards this work through the IMPRINT Project: IMP/2019/000383.

References

Edward Choi, Nan Du, Robert Chen, Le Song, and Jimeng Sun. Constructing disease network and temporal progression model via context-sensitive hawkes process. In *2015 IEEE International Conference on Data Mining*, pages 721–726. IEEE, 2015.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*, pages 301–318. PMLR, 2016.

Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. Gram: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 787–795, 2017.

Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Fifteenth ACM Conference on Recommender Systems*, pages 143–153, 2021.

Mark Ebden. Gaussian processes: A quick introduction. *CoRR: Statistics Theory*, 2015.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Haoji Hu and Xiangnan He. Sets2sets: Learning from sequential sets with neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1491–1499, New York, NY, USA, 2019. Association for Computing Machinery.

Instacart. Instacart market basket analysis, 2017. Dataset hosted on Kaggle since 2017-05-16.

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-iii clinical database demo. *Scientific Data*, 2018.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. Behrt: transformer for electronic health records. *Scientific reports*, 10(1):1–12, 2020.

Thomas Josef Liniger. *Multivariate hawkes processes*. PhD thesis, ETH Zurich, 2009.

Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. Med-bert: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *NPJ digital medicine*, 4(1):1–13, 2021.

Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. Pre-training of graph augmented transformers for medication recommendation. *CoRR arXiv:1906.00346*, 2019.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

Leilei Sun, Yansong Bai, Bowen Du, Chuanren Liu, Hui Xiong, and Weifeng Lv. Dual sequential network for temporal sets prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1439–1448, New York, NY, USA, 2020. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthetic patient generation. *Journal of the American Medical Informatics Association*, 2017.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes process. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11692–11702. PMLR, 13–18 Jul 2020.