Learning Temporal Point Processes for Efficient Retrieval of Continuous Time Event Sequences

Vinayak Gupta*, Srikanta Bedathur* and Abir De†

* IIT Delhi, † IIT Bombay
{vinayak.gupta, srikanta}@cse.iitd.ac.in, abir@cse.iitb.ac.in

Abstract

Recent developments in predictive modeling using marked temporal point processes (MTPP) have enabled an accurate characterization of several real-world applications involving continuous-time event sequences (CTESs). However, the retrieval problem of such sequences remains largely unaddressed in literature. To tackle this, we propose NEUROSEQRET which learns to retrieve and rank a relevant set of continuous-time event sequences for a given query sequence, from a large corpus of sequences. More specifically, NEUROSEQRET first applies a trainable unwarping function on the query sequence, which makes it comparable with corpus sequences, especially when a relevant query-corpus pair has individually different attributes. Next, it feeds the unwarped query sequence and the corpus sequence into MTPP guided neural relevance models. We develop two variants of the relevance model which offer a tradeoff between accuracy and efficiency. We also propose an optimization framework to learn binary sequence embeddings from the relevance scores, suitable for the locality-sensitive hashing leading to a significant speedup in returning top-K results for a given query sequence. Our experiments with several datasets show the significant accuracy boost of NEUROSEQRET beyond several baselines, as well as the efficacy of our hashing mechanism.

1 Introduction

The recent developments in marked temporal point processes (MTPP) has dramatically improved the predictive analytics in several real world applications— from information diffusion in social networks to healthcare— by characterizing them with continuous-time event sequences (CTESs) (Tabibian et al. 2019; Gupta et al. 2021a; Samanta et al. 2017; De et al. 2018; Valera et al. 2014; Rizoiu et al. 2017; Wang et al. 2017; Daley and Vere-Jones 2007; Guo et al. 2018; Du et al. 2015; Tabibian et al. 2019; Kumar et al. 2019; De et al. 2016; Zhang et al. 2021; Du et al. 2016; Farajtabar et al. 2017; Jing and Smola 2017; Saha et al. 2018; Gupta and Bedathur 2021; Likhyani et al. 2020). In this context, given a query sequence, retrieval of *relevant* CTESs from a corpus of sequences is a challenging problem having a wide variety of search-based applications. For example, in audio or music retrieval, one may like to search sequences having different audio or music signatures; in social network, retrieval of trajectories of information diffusion, relevant to a given trajectory can assist in viral marketing, fake news detection, *etc*. Despite having a rich literature on searching similar time-series (Blondel et al. 2021; Gogolou et al. 2020; Alaee et al. 2020; Yoon et al. 2019; Cai et al. 2019; Shen et al. 2018; Cuturi and Blondel 2017; Paparrizos and Gravano 2015), the problem of designing retrieval models specifically for CTES has largely been unaddressed in the past. Moreover, as shown in our experiments, the existing search methods for time sequences are largely ineffective for a CTES retrieval task, since the underlying characterization of the sequences vary across these two domains.

1.1 Present work

In this paper, we first introduce NEUROSEQRET, a family of supervised retrieval models for continuous-time event sequences and then develop a trainable locality sensitive hashing (LSH) based method for efficient retrieval over very large datasets. Specifically, our contributions are as follows:

Query unwarping. The notion of relevance between two sequences varies across applications. A relevant sequence pair can share very different individual attributes, which can mislead the retrieval model if the sequences are compared as-it-is. In other words, an observed sequence may be a warped transformation of a hidden sequence (Xu et al. 2018; Gervini and Gasser 2004). To tackle this problem, NEUROSEQRET first applies a trainable unwarping function on the query sequence before the computation of a relevance score. Such an unwarping function is a monotone transformation, which ensures that the chronological order of events across the observed and the unwarped sequences remains the same (Xu et al. 2018).

Neural relevance scoring model. In principle, the relevance score between two sequences depends on their latent similarity. We measure such similarity by comparing the generative distribution between the query-corpus sequence pairs. In detail, we feed the unwarped query sequence and the corpus sequence into a neural MTPP based relevance scoring model, which computes the relevance score using a Fisher kernel (Jaakkola et al. 1999) between the corpus and the unwarped query sequences. Such a kernel offers two key benefits over other distribution similarity measures, *e.g.*, KL divergence or Wasserstein distance: (i) it computes a natural similarity score between query-corpus sequence pairs in terms of

the underlying generative distributions; and, (ii) it computes a dot product between the gradients of log-likelihoods of the sequence pairs, which makes it compatible with locality-sensitive hashing for certain design choices and facilitates efficient retrieval. In this context, we provide two MTPP models, leading to two variants of NEUROSEQRET, which allows a nice tradeoff between accuracy and efficiency.

SELFATTN-NEUROSEQRET: Here, we use transformer Hawkes process (Zuo et al. 2020) which computes the likelihood of corpus sequences independently of the query sequence. Such a design admits precomputable corpus likelihoods, which in turn allows for prior indexing of the corpus sequences before observing the unseen queries. This setup enables us to apply LSH for efficient retrieval.

CROSSATTN-NEUROSEQRET: Here, we propose a novel cross attention based neural MTPP model to compute the sequence likelihoods. Such a cross-attention mechanism renders the likelihood of corpus sequence dependent on the query sequence, making it a more powerful retrieval model. While CROSSATTN-NEUROSEQRET is not directly compatible with such a hashing based retrieval, it can be employed in a telescopic manner— where a smaller set of relevant candidate set is first retrieved using LSH applied on top of SELFATTN-NEUROSEQRET, and then reranked using CROSSATTN-NEUROSEQRET.

Having computed the relevance scores, we learn the unwarping function and the MTPP model by minimizing a pairwise ranking loss, based on the ground truth relevance labels.

Scalable retrieval. Next, we use the predictions made by SELFATTN-NEUROSEQRET to develop a novel hashing method that enables efficient sequence retrieval. More specifically, we propose an optimization framework that compresses the learned sequence embeddings into binary hash vectors, while simultaneously limiting the loss due to compression. Then, we use locality-sensitive hashing (Gionis et al. 1999) to bucketize the sequences into hash buckets, so that sequences with similar hash representations share the same bucket. Finally, given a query sequence, we consider computing relevance scores only with the sequences within its bucket. Such a hashing mechanism combined with high-quality sequence embeddings achieves fast sequence retrieval with no significant loss in performance.

Finally, our experiments with real-world datasets from different domains show that both variants of NEUROSEQRET outperform several baselines including the methods for continuous-time series retrieval. Moreover, we observe that our hashing method applied on SELFATTN-NEUROSEQRET can make a trade-off between the retrieval accuracy and efficiency more effectively than baselines based on random hyperplanes as well as exhaustive enumeration.

2 Preliminaries

In this section, we first sketch an outline of marked temporal point processes (MTPP) and then setup our problem of retrieving a set of continuous time event sequences relevant to a given query sequence.

2.1 Overview of marked temporal point processes

Continuous time event sequences (CTESs). Marked temporal point processes (MTPP) are stochastic processes which capture the generative mechanism of a sequence of discrete events localized in continuous time. Here, an event e is realized using a tuple (t, x), where $t \in \mathbb{R}_+$ and $x \in \mathcal{X}$ are the arrival time and the mark of the event e. Then, we use $\mathcal{H}(t)$ to denote a continuous time event sequence (CTES) where each event has arrived until and excluding time t, *i.e.*, $\mathcal{H}(t) := \{e_i = (t_i, x_i) \,|\, t_{i-1} < t_i < t\}$. Moreover we use $\mathcal{T}(t)$ and $\mathcal{M}(t)$ to denote the sequence of arrival times $\{t_i \,|\, e_i \in \mathcal{H}(t)\}$ and the marks $\{x_i \,|\, e_i \in \mathcal{H}(t)\}$. Finally, we denote the counting process $N(t)$ as counts of the number of events happened until and excluding time t, encapsulating the generative mechanism of the arrival times.

Generative model for CTES. The underlying MTPP model consists of two components— (i) the dynamics of the arrival times and (ii) the dynamics of the distribution of marks. Most existing works (Du et al. 2016; Zhang et al. 2020; Mei et al. 2019; Mei and Eisner 2017; Shelton et al. 2018; Zuo et al. 2020) model the first component using an intensity function which explicitly models the likelihood of an event in the infinitesimal time window $[t, t + dt)$, *i.e.*, $\lambda(t) = \Pr(dN(t) = 1 | \mathcal{H}(t))$. In contrast, we use an intensity free approach following the proposal by Shchur et al. (2020), where we explicitly model the distribution of the arrival time t of the next event e. Specifically, we denote the density ρ of the arrival time and the distribution m of the mark of the next event as follows:

$$\rho(t)dt = \Pr(e \text{ in } [t, t + dt) \,|\, \mathcal{H}(t)), \tag{1}$$

$$m(x) = \Pr(x \,|\, \mathcal{H}(t)) \tag{2}$$

As discussed by Shchur et al. (2020), such an intensity free MTPP model enjoys several benefits over the intensity based counterparts, in terms of facilitating efficient training, scalable prediction, computation of expected arrival times, etc. Given a sequence of observed events $\mathcal{H}(T)$ collected during the time interval $(0, T]$, the likelihood function is given by:

$$p(\mathcal{H}(T)) = \textstyle\prod_{e_i=(t_i,x_i)\in\mathcal{H}(T)} \rho(t_i) \times m(x_i) \tag{3}$$

2.2 Problem setup

Next, we setup our problem of retrieving a ranked list of sequence from a corpus of continuous time event sequences (CTESs) which are relevant to a given query CTES.

Query and corpus sequences, relevance labels. We operate on a large corpus of sequences $\{\mathcal{H}_c(T_c) \,|\, c \in \mathcal{C}\}$, where $\mathcal{H}_c(T_c) = \{(t_i^{(c)}, x_i^{(c)}) \,|\, t_i^{(c)} < T_c\}$. We are given a set of query sequences $\{\mathcal{H}_q(T_q) \,|\, q \in \mathcal{Q}\}$ with $\mathcal{H}_q(T_q) = \{(t_i^{(q)}, x_i^{(q)}) \,|\, t_i^{(q)} < T_q\}$, as well as a query-specific relevance label for the set of corpus sequences. That is, for a given query sequence $\mathcal{H}_q$, we have: $y(\mathcal{H}_q, \mathcal{H}_c) = +1$ if $\mathcal{H}_c$ is marked as relevant to $\mathcal{H}_q$ and $y(\mathcal{H}_q, \mathcal{H}_c) = -1$ otherwise.

We define $\mathcal{C}_{q+} = \{c \in \mathcal{C} \,|\, y(\mathcal{H}_q, \mathcal{H}_c) = +1\}$, and, $\mathcal{C}_{q-} = \{c \in \mathcal{C} \,|\, y(\mathcal{H}_q, \mathcal{H}_c) = -1\}$, with $\mathcal{C} = \mathcal{C}_{q+} \cup \mathcal{C}_{q-}$. Finally, we denote $T = \max\{T_q, T_c \,|\, q \in \mathcal{Q}, c \in \mathcal{C}\}$ as the maximum time of the data collection.

Our goal. We aim to design an efficient CTES retrieval system, which would return a list of sequences from a known corpus of sequences, relevant to a given query sequence $\mathcal{H}_q$. Therefore, we can view a sequence retrieval task as an instance of ranking problem. Similar to other information retrieval algorithms, a CTES retrieval algorithm first computes the estimated relevance $s(\mathcal{H}_q, \mathcal{H}_c)$ of the corpus sequence $\mathcal{H}_c$ for a given query sequence $\mathcal{H}_q$ and then provides a ranking of $\mathcal{C}$ in the decreasing order of their scores.

3 NEUROSEQRET model

In this section, we describe NEUROSEQRET family of MTPP-based models that we propose for the retrieval of continuous time event sequences (CTES). We begin with an outline of its two key components.

3.1 Components of NEUROSEQRET

NEUROSEQRET models the relevance scoring function between query and corpus sequence pairs. However the relevance of a corpus sequence to the query is latent and varies widely across applications. To accurately characterize this relevance measure, NEUROSEQRET works in two steps. First, it unwarps the query sequences to make them compatible for comparing with the corpus sequences. Then, it computes the pairwise relevance score between the query and corpus sequences using neural MTPP models.

Unwarping query sequences. Direct comparison between a query and a corpus sequence can provide misleading outcomes, since they also contain their own individual idiosyncratic factors in addition to sharing some common attributes. In fact, a corpus sequence can be highly relevant to the query, despite greatly varying in timescale, initial time, etc. In other words, it may have been generated by applying a warping transformation on a latent sequence (Xu et al. 2018; Gervini and Gasser 2004). Thus, a direct comparison between a relevant sequence pair may give poor relevance score.

To address this challenge, we first apply a trainable unwarping function (Xu et al. 2018) $U(\cdot)$ on the arrival times of a query sequence, which enhances its compatibility for comparing it with the corpus sequences. More specifically, we define $U(\mathcal{H}_q) := \{(U(t_i^{(q)}), x_i^{(q)})\}$. In general, U satisfies two properties (Xu et al. 2018; Gervini and Gasser 2004): *unbiasedness*, *i.e.*, having a small value of $\|U(t)] - t\|$ and *monotonicity*, *i.e.*, $dU(t)/dt \geq 0$. These properties ensure that the chronological order of the events across both the warped observed sequence and the unwarped sequence remains same. Such a sequence transformation learns to capture the similarity between two sequences, even if it is not apparent due to different individual factors, as we shall later in our experiments (Figure 1).

Computation of relevance scores. Given a query sequence $\mathcal{H}_q$ and a corpus sequence $\mathcal{H}_c$, we compute the relevance score $s(\mathcal{H}_q, \mathcal{H}_c)$ using two similarity scores, *e.g.*, (i) a *model independent* sequence similarity score and (ii) a *model based* sequence similarity score.

—*Model independent similarity score:* Computation of model independent similarity score between two sequences is widely studied in literature (Xiao et al. 2017; Mueen and Keogh 2016; Su et al. 2020; Abanda et al. 2019; Müller 2007). They are computed using different distance measures between two sequences, *e.g.*, DTW, Wasserstein distance, etc. and therefore, can be immediately derived from data without using the underlying MTPP model. In this work, we compute the model independent similarity score, $\textsc{Sim}_U(\mathcal{H}_q, \mathcal{H}_c)$, between $\mathcal{H}_q$ and $\mathcal{H}_c$ as follows:

$$\textsc{Sim}_U(\mathcal{H}_q, \mathcal{H}_c) = -\Delta_t(U(\mathcal{H}_q), \mathcal{H}_c) - \Delta_x(\mathcal{H}_q, \mathcal{H}_c) \quad (4)$$

where, Δ_t and Δ_x are defined as:

$$\Delta_t(U(\mathcal{H}_q), \mathcal{H}_c) = \sum_{i=0}^{H_{\min}} \left| U(t_i^{(q)}) - t_i^{(c)} \right| + \sum_{\substack{t_i \in \mathcal{H}_c \cup \mathcal{H}_q \\ i > |H_{\min}|}} (T - t_i),$$

$$\Delta_x(\mathcal{H}_q, \mathcal{H}_c) = \sum_{i=0}^{H_{\min}} \mathbb{I}[x_i^{(q)} \neq x_i^{(c)}] + \big| |\mathcal{H}_c| - |\mathcal{H}_q| \big|.$$

Here, $H_{\min} = \min\{|\mathcal{H}_q|, |\mathcal{H}_c|\}$, $T = \max\{T_q, T_c\}$ where the events of $\mathcal{H}_q$ and $\mathcal{H}_c$ are gathered until time T_q and T_c respectively; $\Delta_t(U(\mathcal{H}_q), \mathcal{H}_c)$ is the Wasserstein distance between the unwarped arrival time sequence $U(\mathcal{H}_q)$ and the corpus sequence (Xiao et al. 2017) and, $\Delta_x(\mathcal{H}_q, \mathcal{H}_c)$ measures the matching error for the marks, wherein the last term penalizes the marks of last $|\mathcal{H}_c| - |\mathcal{H}_q|$ events of $|\mathcal{H}_c|$.

—*Model based similarity score using Fisher kernel:* We hypothesize that the relevance score $s(\mathcal{H}_q, \mathcal{H}_c)$ also depends on a latent similarity which may not be immediately evident from the observed query and corpus sequences even after unwarping. Such a similarity can be measured by comparing the generative distributions of the query-corpus sequence pairs. To this end, we first develop an MTPP based generative model $p_\theta(\mathcal{H})$ parameterized by θ and then compute a similarity score using the Fisher similarity kernel between the unwarped query and corpus sequence pairs $(U(\mathcal{H}_q), \mathcal{H}_c)$ (Jaakkola et al. 1999). Specifically, we compute the relevance score between the unwarped query sequence $U(\mathcal{H}_q)$ and the corpus sequence $\mathcal{H}_c$ as follows:

$$\kappa_{p_\theta}(\mathcal{H}_q, \mathcal{H}_c) = \boldsymbol{v}_{p_\theta}(U(\mathcal{H}_q))^\top \boldsymbol{v}_{p_\theta}(\mathcal{H}_c) \quad (5)$$

where θ is the set of trainable parameters; $\boldsymbol{v}_{p_\theta}(\cdot)$ is given by

$$\boldsymbol{v}_p(\mathcal{H}) = \boldsymbol{I}_\theta^{-1/2} \nabla_\theta \log p_\theta(\mathcal{H}) / ||\boldsymbol{I}_\theta^{-1/2} \nabla_\theta \log p_\theta(\mathcal{H})||_2,$$

$\boldsymbol{I}_\theta$ is the Fisher information matrix (Jaakkola et al. 1999), *i.e.*, $\boldsymbol{I}_\theta = \mathbb{E}_{\mathcal{H} \sim p_\theta(\bullet)} \left[\nabla_\theta \log p_\theta(\mathcal{H}) \nabla_\theta \log p_\theta(\mathcal{H})^\top \right]$. We would like to highlight that $\kappa_{p_\theta}(\mathcal{H}_q, \mathcal{H}_c)$ in Eq. (5) is a normalized version of Fisher kernel since $||\boldsymbol{v}_{p_\theta}(\cdot)|| = 1$. Thus, $\kappa_{p_\theta}(\mathcal{H}_q, \mathcal{H}_c)$ measures the cosine similarity between $\boldsymbol{v}_{p_\theta}(U(\mathcal{H}_q))$ and $\boldsymbol{v}_{p_\theta}(\mathcal{H}_c)$.

Note that, KL divergence or Wasserstein distance could also serve our purpose of computing the latent similarity between the generative distributions. However, we choose Fisher similarity kernel because of two reasons: (i) it is known to be a natural similarity measure which allows us to use the underlying generative model in a discriminative learning task (Jaakkola et al. 1999; Sewell 2011); and, (ii) unlike KL divergence or other distribution (dis)similarities, it computes the cosine similarity between $\boldsymbol{v}_{p_\theta}(U(\mathcal{H}_q))$ and $\boldsymbol{v}_{p_\theta}(\mathcal{H}_c)$, which makes it compatible with locality sensitive hashing (Charikar 2002).

Finally, we compute the relevance score as

$$s_{p,U}\left(\mathcal{H}_q, \mathcal{H}_c\right) = \kappa_p\left(\mathcal{H}_q, \mathcal{H}_c\right) + \gamma \textsc{Sim}_U\left(\mathcal{H}_q, \mathcal{H}_c\right) \quad (6)$$

where γ is a hyperparameter.

3.2 Neural parameterization of NEUROSEQRET

Here, we first present the neural architecture of the unwarping function and then describe the MTPP models used to compute the model based similarity score in Eq. (5). As we describe later, we use two MTPP models with different levels of modeling sophistication, *viz.*, SELFATTN-NEUROSEQRET and CROSSATTN-NEUROSEQRET. In SELFATTN-NEUROSEQRET, the likelihood of a corpus sequence is computed independently of the query sequence using a self attention based MTPP model, *e.g.*, Transformer Hawkes Process (Zuo et al. 2020). As a result, we can employ a locality sensitive hashing based efficient retrieval based SELFATTN-NEUROSEQRET. In CROSSATTN-NEUROSEQRET, on the other hand, we propose a more expressive and novel cross attention MTPP model, where the likelihood of a corpus sequence is dependent on the query sequence. Thus, our models can effectively tradeoff between accuracy and efficiency.

Neural architecture of $U(\cdot)$. As discussed in Section 3.1, the unwarping function $U(\cdot)$ should be unbiased and monotonic. To this end, we model $U(\cdot) \approx U_\phi(\cdot)$ using a nonlinear monotone function which is computed using an unconstrained monotone neural network (UMNN) (Wehenkel and Louppe 2019), *i.e.*,

$$U_\phi(t) = \int_0^t u_\phi(\tau) d\tau + \eta, \quad (7)$$

where ϕ is the parameter of the underlying neural network $u_\phi(\cdot)$, $\eta \in \mathcal{N}(0, \sigma)$ and $u_\phi : \mathbb{R} \to \mathbb{R}_+$ is a non-negative nonlinear function. Since the underlying monotonicity can be achieved only by enforcing non-negativity of the integrand u_ϕ, UMNN admits an unconstrained, highly expressive parameterization of monotonic functions. Therefore, any complex unwarping function $U_\phi(\cdot)$ can be captured using Eq. (7), by integrating a suitable neural model augmented with ReLU$(\cdot)$ in the final layer. In other words, if u_ϕ is a universal approximator for positive function, then U_ϕ can capture any differentiable unwarping function. We impose an additional regularizer $\frac{1}{\sigma^2}\int_0^T \left\|u_\phi(t) - 1\right\|^2 dt$ on our training loss which ensures that $\|U(t) - t\|$ remains small.

Neural architecture of MTPP model $p_\theta(\cdot)$. We provide two variants of $p_\theta(\cdot)$, which leads to two retrieval models, *viz.*, SELFATTN-NEUROSEQRET and CROSSATTN-NEUROSEQRET.

SELFATTN-NEUROSEQRET: Here, we use Transformer Hawkes process (Zuo et al. 2020) which applies a self attention mechanism to model the underlying generative process. In this model, the gradient of corpus sequences $\boldsymbol{v}_\theta(\mathcal{H}_c) = \nabla_\theta \log p_\theta(\mathcal{H}_c)$ are computed independently of the query sequence $\mathcal{H}_q$. Once we train the retrieval model, $\boldsymbol{v}_\theta(\mathcal{H}_c)$ can be pre-computed and bucketized before observing the test query. Such a model, together with the Fisher kernel based cosine similarity scoring model, allows us to apply locality sensitive hashing for efficient retrieval.

CROSSATTN-NEUROSEQRET: The above self attention based mechanism models a query agnostic likelihood of the corpus sequences. Next, we introduce a *cross attention* based MTPP model which explicitly takes into account of the underlying query sequence while modeling the likelihood of the corpus sequence. Specifically, we measure the latent relevance score between $\mathcal{H}_q$ and $\mathcal{H}_c$ via a query-induced MTPP model built using the cross attention between the generative process of both the sequences. Given a query sequence $\mathcal{H}_q$ and the first r events of the corpus sequence $\mathcal{H}_c$, we parameterize the generative model for $(r+1)$-th event, *i.e.*, $p(e_{r+1}^{(c)} \mid \mathcal{H}(t_r))$ as $p_\theta(\cdot)$, where $p_\theta(e_{r+1}^{(c)}) = \rho_\theta(t_{r+1}^{(c)})\, m_\theta(x_{r+1}^{(c)})$, where ρ and m are the density and distribution functions for the arrival time and the mark respectively, as described in Eq. (1).

—*Input Layer:* For each event $e_i^{(q)}$ in the query sequence $\mathcal{H}_q$ and each event $e_j^{(c)}$ in the first r events in the corpus sequence $\mathcal{H}_c$, the input layer computes the initial embeddings $\boldsymbol{y}_i^{(q)}$ and $\boldsymbol{y}_j^{(c)}$ as follows:

$$\begin{aligned}
\boldsymbol{y}_i^{(q)} &= \boldsymbol{w}_{y,x} x_i^{(q)} + \boldsymbol{w}_{y,t} U(t_i^{(q)}) \\
&\quad + \boldsymbol{w}_{y,\Delta t}\left(U(t_i^{(q)}) - U(t_{i-1}^{(q)})\right) + \boldsymbol{b}_y, \forall i \in [|\mathcal{H}_q|] \\
\boldsymbol{y}_j^{(c)} &= \boldsymbol{w}_{y,x} x_j^{(c)} + \boldsymbol{w}_{y,t} t_j^{(c)} \\
&\quad + \boldsymbol{w}_{y,\Delta t}\left(t_j^{(c)} - t_{j-1}^{(c)}\right) + \boldsymbol{b}_y, \forall j \in [|\mathcal{H}_c(t_r)| - 1]
\end{aligned}$$

where $\boldsymbol{w}_{\bullet,\bullet}$ and $\boldsymbol{b}_y$ are trainable parameters.

—*Attention layer:* The second layer models the interaction between all the query events and *the past corpus events*, *i.e.*, $\mathcal{H}_q$ and $\mathcal{H}_c(t_r)$ using an attention mechanism. Specifically, following the existing attention models (Vaswani et al. 2017; Kang and McAuley 2018; Li et al. 2020) it first adds a trainable position embedding $\boldsymbol{p}$ with $\boldsymbol{y}$— the output from the previous layer. More specifically, we have the updates: $\boldsymbol{y}_i^{(q)} \leftarrow \boldsymbol{y}_i^{(q)} + \boldsymbol{p}_i$ and $\boldsymbol{y}_j^{(c)} \leftarrow \boldsymbol{y}_j^{(c)} + \boldsymbol{p}_j$. where, $\boldsymbol{p}_\bullet \in \mathbb{R}^D$. Next, we apply two linear transformations on the vectors $[\boldsymbol{y}_i^{(q)}]_{i \in [|\mathcal{H}_q|]}$ and one linear transformation on $[\boldsymbol{y}_j^{(c)}]_{j \in [r]}$, *i.e.*, $\boldsymbol{s}_j = \boldsymbol{W}^S \boldsymbol{y}_j^{(c)}, \boldsymbol{k}_i = \boldsymbol{W}^K \boldsymbol{y}_i^{(q)}, \boldsymbol{v}_i = \boldsymbol{W}^V \boldsymbol{y}_i^{(q)}$. The state-of-the-art works on attention models (Vaswani et al. 2017; Zuo et al. 2020; Kang and McAuley 2018; Li et al. 2020) often refer $\boldsymbol{s}_\bullet$, $\boldsymbol{k}_\bullet$ and $\boldsymbol{v}_\bullet$ as `query`, `key` and `value` vectors respectively. Similarly, we call the trainable weights $\boldsymbol{W}^S, \boldsymbol{W}^K$ and $\boldsymbol{W}^V$ as the `Query`, `Key` and `Value` matrices, respectively. Finally, we use the standard attention recipe (Vaswani et al. 2017) to compute the final embedding vector $\boldsymbol{h}_j^{(c,q)}$ for the event $e_j^{(c)}$, induced by query $\mathcal{H}_q$. Such a recipe adds the values weighted by the outputs of a softmax function induced by the `query` and `key`, *i.e.*,

$$\boldsymbol{h}_j^{(c,q)} = \sum_{i \in [|\mathcal{H}_q|]} \frac{\exp\left(\boldsymbol{s}_j^\top \boldsymbol{k}_i / \sqrt{D}\right)}{\sum_{i' \in [|\mathcal{H}_q|]} \exp\left(\boldsymbol{s}_j^\top \boldsymbol{k}_{i'} / \sqrt{D}\right)} \boldsymbol{v}_i, \quad (8)$$

—*Output layer:* Given the vectors $\boldsymbol{h}_j^{(c,q)}$ provided by the attention mechanism (8), we first apply a feed-forward neural

network on them to compute $\overline{\boldsymbol{h}}_r^{(c,q)}$ as follows:

$$\overline{\boldsymbol{h}}_r^{(c,q)} = \sum_{j=1}^{r} \left[\boldsymbol{w}_{\overline{h}} \odot \text{ReLU}(\boldsymbol{h}_j^{(c,q)} \odot \boldsymbol{w}_{h,f} + \boldsymbol{b}_{h,o}) + \boldsymbol{b}_{\overline{h}} \right],$$

where $\boldsymbol{w}_{\bullet,\bullet}$, $\boldsymbol{w}_{\bullet}$ and $\boldsymbol{b}_v$. Finally, we use these vectors to compute the probability density of the arrival time of the next event $e_{r+1}^{(c)}$, *i.e.*, $\rho_\theta(t_{r+1})$ and the mark distribution $m_\theta(x_{r+1})$. In particular, we realize $\rho_\theta(t_{r+1})$ using a log normal distribution of inter-arrival times, *i.e.*,

$$t_{r+1}^{(c)} - t_r^{(c)} \sim \text{LogNormal}\left(\mu_e\left(\overline{\boldsymbol{h}}_r^{(c,q)}\right), \sigma_e^2\left(\overline{\boldsymbol{h}}_r^{(c,q)}\right)\right),$$

where, $\left[\mu_e\left(\overline{\boldsymbol{h}}_r^{(c,q)}\right), \sigma_e\left(\overline{\boldsymbol{h}}_r^{(c,q)}\right)\right] = \boldsymbol{W}_{t,q}\overline{\boldsymbol{h}}_r^{(c,q)} + \boldsymbol{b}_{t,q}$. Similarly, we model the mark distribution as,

$$m_\theta(x_{r+1}) = \frac{\exp\left(\boldsymbol{w}_{x,m}^\top \overline{\boldsymbol{h}}^{(c,q)} + b_{x,m}\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\boldsymbol{w}_{x',m}^\top \overline{\boldsymbol{h}}^{(c,q)} + b_{x',m}\right)}, \quad (9)$$

where $\boldsymbol{W}_{\bullet,\bullet}$ are the trainable parameters. Therefore the set of trainable parameters for the underlying MTPP models is $\theta = \{\boldsymbol{W}^\bullet, \boldsymbol{W}_{\bullet,\bullet}, \boldsymbol{w}_\bullet, \boldsymbol{w}_{\bullet,\bullet}, \boldsymbol{b}_\bullet, \boldsymbol{b}_{\bullet,\bullet}\}$.

3.3 Parameter estimation

Given the query sequences $\{\mathcal{H}_q\}$, the corpus sequences $\{\mathcal{H}_c\}$ along with their relevance labels $\{y(\mathcal{H}_q, \mathcal{H}_c)\}$, we seek to find θ and ϕ which ensure that:

$$s_{p_\theta, U_\phi}(\mathcal{H}_q, \mathcal{H}_{c_+}) \gg s_{p_\theta, U_\phi}(\mathcal{H}_q, \mathcal{H}_{c_-}) \, \forall \, c_\pm \in \mathcal{C}_{q\pm}. \quad (10)$$

To this aim, we minimize the following pairwise ranking loss (Joachims 2002) to estimate the parameters θ, ϕ:

$$\min_{\theta,\phi} \sum_{q \in \mathcal{Q}} \sum_{\substack{c_+ \in \mathcal{C}_{q+}, \\ c- \in \mathcal{C}_{q-}}} \left[s_{p_\theta, U_\phi}(\mathcal{H}_q, \mathcal{H}_{c_-}) - s_{p_\theta, U_\phi}(\mathcal{H}_q, \mathcal{H}_{c_+}) + \delta \right]_+,$$

where, δ is a tunable margin.

4 Scalable retrieval with hashing

Once we learn the model parameters θ and ϕ, we can rank the set of corpus sequences $\mathcal{H}_c$ in the decreasing order of $s_{p_\theta, U_\phi}(\mathcal{H}_{q'}, \mathcal{H}_c)$ for a new query $\mathcal{H}_{q'}$ and return top$-K$ sequences. Such an approach requires $|\mathcal{C}|$ comparisons per each test query, which can take a huge amount of time for many real-life applications where $|\mathcal{C}|$ is high. However, for most practical query sequences, the number of relevant sequences is a very small fraction of the entire corpus of sequences. Therefore, the number of comparisons between query-corpus sequence pairs can be reduced without significantly impacting the retrieval quality by selecting a small number of candidate corpus sequences that are more likely to be relevant to a query sequence.

4.1 Trainable hashing for retrieval

Computation of a trainable hash code. We first apply a trainable nonlinear transformation Λ_ψ with parameter ψ on the gradients $\boldsymbol{v}^c = \boldsymbol{v}_{p_\theta}(\mathcal{H}_c)$ and then learn the binary hash vectors $\boldsymbol{\zeta}^c = \text{sign}\left(\Lambda_\psi\left(\boldsymbol{v}^c\right)\right)$ by solving the following optimization, where we use $\tanh\left(\Lambda_\psi(\cdot)\right)$ as a smooth approximation of $\text{sign}\left(\Lambda_\psi(\cdot)\right)$.

Algorithm 1: Efficient retrieval with hashing

Require: Trained corpus embeddings $\{\boldsymbol{v}^c = \boldsymbol{v}_{p_\theta}(\mathcal{H}_c)\}$ using SelfAttn-NeuroSeqRet; new query sequences $\{\mathcal{H}_{q'}\}$, K: # of corpus sequences to return; trained models for SelfAttn-NeuroSeqRet and CrossAttn-NeuroSeqRet.

1: **Output:** $\{L_{q'}\}$: top-K relevant sequences from $\{\mathcal{H}_c\}$.
2: $\psi \leftarrow$ TrainHashNet $(\Lambda_\psi, [\boldsymbol{v}^c]_{c \in \mathcal{C}})$
3: InitHashBuckets$(\cdot)$
4: **for** $c \in \mathcal{C}$ **do**
5: $\quad \boldsymbol{\zeta}^c \leftarrow$ ComputeHashCode $(\boldsymbol{v}^c; \Lambda_\psi)$
6: $\quad \mathcal{B} \leftarrow$ AssignBucket$(\boldsymbol{\zeta}^c)$
7: **end for**
8: **for** each new query $\mathcal{H}_{q'}$ **do**
9: $\quad \boldsymbol{v}^{q'} \leftarrow$ SelfAttn-NeuroSeqRet$(\mathcal{H}_{q'})$
10: $\quad \boldsymbol{\zeta}^{q'} \leftarrow$ ComputeHashCode$(\boldsymbol{v}^{q'}; \Lambda_\psi)$
11: $\quad \mathcal{B} \leftarrow$ AssignBucket$(\boldsymbol{\zeta}^{q'})$
12: $\quad$ **for** $c \in \mathcal{B}$ **do**
13: $\quad\quad \boldsymbol{v}_{\text{cross}}^{q'}, \boldsymbol{v}_{\text{cross}}^{c} \leftarrow$ CrossAttn-NeuroSeqRet$(\mathcal{H}_{q'}, \mathcal{H}_c)$
14: $\quad\quad s_{p_\theta, U_\phi}(\mathcal{H}_{q'}, \mathcal{H}_c) \leftarrow$ Score$(\boldsymbol{v}_{\text{cross}}^{q'}, \boldsymbol{v}_{\text{cross}}^{c}, \mathcal{H}_q, \mathcal{H}_c)$
15: $\quad$ **end for**
16: $\quad L_{q'} \leftarrow$ Rank$\left(\left\{s_{p_\theta, U_\phi}(\mathcal{H}_{q'}, \mathcal{H}_c)\right\}, K\right)$
17: **end for**
18: **Return** $\{L_{q'}\}$

$$\min_{\psi} \frac{\eta_1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \left| \mathbf{1}^\top \tanh\left(\Lambda_\psi\left(\boldsymbol{v}^c\right)\right) \right| + \frac{\eta_2}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \left\| \left| \tanh\left(\Lambda_\psi\left(\boldsymbol{v}^c\right)\right) \right| - \mathbf{1} \right\|_1 + \frac{2\eta_3}{\binom{D}{2}} \cdot \left| \sum_{\substack{c \in \mathcal{C} \\ i \neq j \in [D]}} \tanh\left(\Lambda_\psi\left(\boldsymbol{v}^c\right)[i]\right) \cdot \tanh\left(\Lambda_\psi\left(\boldsymbol{v}^c\right)[j]\right) \right| \quad (11)$$

Here $\sum_{i=1}^{3} \eta_i = 1$. Moreover, different terms in Eq. (11) allow the hash codes $\boldsymbol{\zeta}^c$ to have a set of four desired properties: (i) the first term ensures that the numbers of $+1$ and -1 are evenly distributed in the hash vectors $\boldsymbol{\zeta}^c = \tanh\left(\Lambda_\psi\left(\boldsymbol{v}^c\right)\right)$; (ii) the second term encourages the entries of $\boldsymbol{\zeta}^c$ become as close to ± 1 as possible so that $\tanh(\cdot)$ gives an accurate approximation of $\text{sign}(\cdot)$; (iii) the third term ensures that the entries of the hash codes $\boldsymbol{\zeta}^c$ contain independent information and therefore they have no redundant entries. Trainable hashing has been used in other domains of information retrieval including graph hashing (Liu et al. 2014; Qin et al. 2020; Roy et al. 2020), document retrieval (Zhang et al. 2010; Salakhutdinov and Hinton 2009; Zamani Dadaneh et al. 2020). However, to the best of our knowledge, such an approach has never been proposed for continuous time sequence retrieval.

Outline of our retrieval method. We summarize our retrieval procedure in Algorithm 1. We are given gradient vectors $\boldsymbol{v}^c = \boldsymbol{v}_{p_\theta}(\mathcal{H}_c)$ obtained by training SelfAttn-NeuroSeqRet. Next, we train an additional neural network Λ_ψ parameterized by ψ (TrainHashNet(), line 2), which is used to learn a binary hash vector $\boldsymbol{\zeta}^c$ for each sequence $\mathcal{H}_c$. Then these hash codes are used to arrange corpus sequences in different hash buckets (for-loop in lines 4–7) using the algorithm proposed by Gionis et al. (1999), so that two sequences $\mathcal{H}_c, \mathcal{H}_{c'}$ lying in the same hash bucket have very high value of cosine similarity $\cos(\boldsymbol{v}^c, \boldsymbol{v}^{c'})$ (bucketization details are in Appendix B). Finally, once a new query $\mathcal{H}_{q'}$

comes, we first compute $\boldsymbol{v}^{q'}$ using the trained SELFATTN-NEUROSEQRET model and then compute the binary hash codes $\boldsymbol{\zeta}^{q'}$ using the trained hash network Λ_ψ (lines 9–10). Next, we assign an appropriate bucket $\mathcal{B}$ to it (line 11) and finally compare it with only the corpus sequences in the same bucket, *i.e.*, $\mathcal{H}_c \in \mathcal{B}$ (lines 12–15) using our model.

Recall that we must use SELFATTN-NEUROSEQRET to compute gradient vectors for subsequent hash code generation (in lines 9–10). However, at the last stage for final score computation and ranking, we can use any variant of NEUROSEQRET (in line 13), preferably CROSSATTN-NEUROSEQRET, since the corpus sequences have already been indexed by our LSH method.

5 Experiments

In this section, we provide a comprehensive evaluation of NEUROSEQRET and our hashing method. Appendix D (Gupta et al. 2021b) contains additional experiments. Our implementation and datasets are available at `https://github.com/data-iitd/neuroseqret/`.

5.1 Experimental Setup

Datasets. We evaluate the retrieval performance of NEUROSEQRET and other methods across large-scale real-world datasets with up to 60 million events. The statistics of all datasets are given in Appendix (Gupta et al. 2021b). Across all datasets, $|\mathcal{H}_q| = 5K$ and $|\mathcal{H}_c| = 200K$.

(1) **Audio:** The dataset contains audio files for spoken commands to a smart-light system and the demographics(age, nationality) of the speaker. Here, a query corpus sequence pair is relevant if they are from an audio file with a common speaker.

(2) **Sports:** The dataset contains actions (*e.g.*run, pass, shoot) taken while playing different sports. We consider the time of action and action class as time and mark of sequence respectively. Here, a query corpus sequence pair is relevant if they are from a common sport.

(3) **Celebrity:** In this dataset, we consider the series of frames extracted from youtube videos of multiple celebrities as event sequences where event-time denotes the video-time and the mark is decided upon the coordinates of the frame where the celebrity is located. Here, a query corpus sequence pair is relevant if they are from a video file having a common celebrity.

(4) **Electricity:** This dataset contains the power-consumption records of different devices across smart-homes in the UK. We consider the records for each device as a sequence with event mark as the *normalized* change in the power consumed by the device and the time of recording as event time. Here, a query corpus sequence pair is relevant if they are from a similar appliance.

(5) **Health:** The dataset contains ECG records for patients suffering from heart-related problems. Since the length of the ECG record for a single patient can be up to 10 million, we generate smaller individual sequences of length 10,000 and consider each such sequence as an independent sequence. The marks and times of events in a sequence are determined using a similar procedure as in Electricity. A query corpus sequence pair is relevant if they are from a common patient.

For Health, Celebrity and Electricity, we lack the true ground-truth labeling of relevance between sequences. Therefore, we adopt a heuristic in which, given a dataset $\mathcal{D}$, from each sequence $\texttt{seq}_q \in \mathcal{D}$ with $q \in [|\mathcal{D}|]$, we first sample a set of sub-sequences $\mathcal{U}_q = \{\mathcal{H} \subset \texttt{seq}_q\}$ with $|\mathcal{U}_q| \sim \text{Unif}\,[200, 300]$. For each such collection $\mathcal{U}_q$, we draw exactly one query $\mathcal{H}_q$ uniformly at random from $\mathcal{U}_q$, *i.e.*, $\mathcal{H}_q \sim \mathcal{U}_q$. Then, we define $\mathcal{C} = \cup_{q \in [|\mathcal{D}|]} \mathcal{U}_q \backslash \mathcal{H}_q$, $\mathcal{C}_{q+} = \mathcal{U}_q \backslash \mathcal{H}_q$ and $\mathcal{C}_{q-} = \cup_{c \neq q} (\mathcal{U}_c \backslash \mathcal{H}_c)$.

Baselines. We consider three continuous time-series retrieval models: (i) MASS (Mueen et al. 2017), (ii) UDTW (Rakthanmanon et al. 2012) and (iii) Sharp (Blondel et al. 2021); and, three MTPP models (iv) RMTPP (Du et al. 2016), (v) SAHP (Zhang et al. 2020), and (vi) THP (Zuo et al. 2020). For sequence retrieval with MTPP models, we first train them across all the sequences using maximum likelihood estimation. Then, given a test query $\mathcal{H}_{q'}$, this MTPP method ranks the corpus sequences $\{\mathcal{H}_c\}$ in decreasing order of their cosine similarity $\text{CosSim}(\texttt{emb}^{(q')}, \texttt{emb}^{(c)})$, where $\texttt{emb}^{(\bullet)}$ is the corresponding sequence embedding provided by the underlying MTPP model. In addition, we build supervised ranking models over these approaches, *viz.,* `Rank`-RMTPP, `Rank`-SAHP and `Rank`-THP corresponding to RMTPP, SAHP, and THP. Specifically, `Rank`-MTPP formulates a ranking loss on the query-corpus pairs based on the cosine similarity scores along with the likelihood function to get the final training objective. Therefore, the vanilla MTPP models are used as unsupervised models and the corresponding `Rank`-MTPP models work as supervised models. Appendix C (Gupta et al. 2021b) contains the implementation details.

Evaluation protocol. We partition the set of queries $\mathcal{Q}$ into 50% training, 10% validation and rest as test sets. First, we train a retrieval model using the set of training queries. Then, for each test query q', we use the trained model to obtain a top-K ranked list from the corpus sequences. Next, we compute the average precision (AP) and discounted cumulative gain (DCG) of each top-K list, based on the ground truth. Finally, we compute the mean average precision (MAP) and NDCG@K, by averaging AP and DCG values across all test queries. We set $K = 10$.

5.2 Results on retrieval accuracy

Comparison with baselines. First, we compare our model against the baselines in terms of MAP and NDCG. Table 1 summarizes the results, which shows that (i) both CROSSATTN-NEUROSEQRET and SELFATTN-NEUROSEQRET outperform all the baselines by a substantial margin; (ii) CROSSATTN-NEUROSEQRET outperforms SELFATTN-NEUROSEQRET, since the former has a higher expressive power; (iii) the variants of baseline MTPP models trained for sequence retrieval, *i.e.*, `Rank`-RMTPP, `Rank`-SAHP, and `Rank`-THP outperform the vanilla MTPP models; (iv) the performances of vanilla MTPPs and the time series retrieval models (MASS, UDTW and Sharp) are comparable.

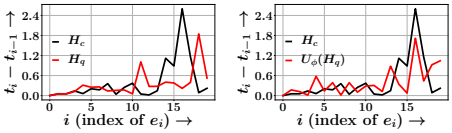
Ablation study. Next, we compare the retrieval performance across four model variants: (i) our model with only model-independent score *i.e.*, $s_{p_\theta, U_\phi}(\mathcal{H}_q, \mathcal{H}_c) = -\Delta_x(\mathcal{H}_q, \mathcal{H}_c) -$

	Mean Average Precision (MAP) in %					NDCG@10 in %				
	Audio	Celebrity	Electricity	Health	Sports	Audio	Celebrity	Electricity	Health	Sports
MASS (Mueen et al. 2017)	51.1±0.0	58.2±0.0	19.3±0.0	26.4±0.0	54.7±0.0	20.7±0.0	38.7±0.0	9.1±0.0	13.6±0.0	22.3±0.0
UDTW (Rakthanmanon et al. 2012)	50.7±0.0	58.7±0.0	20.3±0.0	28.1±0.0	54.5±0.0	21.3±0.0	39.6±0.0	9.7±0.0	14.7±0.0	22.9±0.0
Sharp (Blondel et al. 2021)	52.4±0.2	59.8±0.5	22.8±0.2	28.6±0.2	56.8±0.3	21.9±0.2	40.6±0.5	11.7±0.1	16.8±0.1	23.7±0.2
RMTPP (Du et al. 2016)	48.9±2.3	57.6±1.8	18.7±0.8	24.8±1.2	50.3±2.5	20.1±1.9	39.4±2.1	8.3±0.8	12.3±0.5	19.1±1.8
Rank-RMTPP	52.6±2.0	60.3±1.7	23.4±0.7	29.3±0.6	55.8±2.1	22.4±1.3	41.2±1.3	11.4±0.4	15.5±0.5	23.9±1.4
SAHP (Zhang et al. 2020)	49.4±3.2	57.2±2.9	19.0±1.8	26.0±2.1	53.9±3.6	20.4±2.3	39.0±3.1	8.7±1.2	13.2±1.4	22.6±2.5
Rank-SAHP	52.9±1.8	61.8±2.3	26.5±1.2	31.6±1.1	55.1±2.3	23.3±1.4	42.1±1.7	13.3±0.7	17.5±0.9	25.4±1.8
THP (Zuo et al. 2020)	51.8±2.3	60.3±1.9	21.3±0.9	27.9±0.9	54.2±2.1	22.1±1.1	40.3±1.2	10.4±0.6	14.4±0.3	22.9±1.1
Rank-THP	54.3±1.7	63.1±2.1	29.4±0.9	33.6±1.3	56.3±1.9	25.4±0.9	44.2±1.0	15.3±0.4	19.7±0.4	26.5±0.9
SELFATTN-NEUROSEQRET	55.8±1.8	64.4±1.9	30.7±0.7	35.9±0.9	57.6±1.9	25.9±1.1	45.8±1.0	16.5±0.5	20.4±0.4	27.8±1.1
CROSSATTN-NEUROSEQRET	**56.2±2.1**†	**65.1±1.9**†	**32.4±0.8**†	**37.4±0.9**†	**58.7±2.1**	**28.3±1.1**†	**46.9±1.2**†	**18.1±0.7**†	**22.0±0.4**†	**27.9±1.2**

Table 1: Retrieval accuracy in terms of mean average precision (MAP) and NDCG@10 (both in %) of all the methods across five datasets on the test set. Numbers with bold font (underline) indicate best (second best) performer. **Boxed** numbers indicate best performing state-of-the-art baseline. Results marked † are statistically significant (two-sided Fisher's test with $p \leq 0.05$) over the best performing state-of-the-art baseline (Rank-THP or Sharp). The standard deviation for MASS and UDTW are zero, since they are deterministic retrieval algorithms.

Variant	Audio	Celebrity	Health
(i) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = -\Delta_x(\mathcal{H}_q,\mathcal{H}_c) - \Delta_t(U_\phi(\mathcal{H}_q),\mathcal{H}_c)$	36.1±0.0	43.7±0.0	18.9±0.0
(ii) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c)$	53.9±1.9	62.5±1.3	33.6±0.7
(iii) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c) - \gamma\Delta_x(\mathcal{H}_q,\mathcal{H}_c)$	54.6±1.9	63.1±1.4	33.7±0.7
(iv) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c) - \gamma\Delta_t(U_\phi(\mathcal{H}_q),\mathcal{H}_c)$	55.7±2.0	63.7±1.8	35.9±0.8
(v) CROSSATTN-NEUROSEQRET Without $U_\phi(\cdot)$	55.2±2.2	62.9±2.0	34.3±0.9
(vi) CROSSATTN-NEUROSEQRET	56.2±2.1	65.1±1.9	37.4±0.9

Table 2: Ablation study.



(a) $\mathcal{H}_q, \mathcal{H}_c$ (b) $U_\phi(\mathcal{H}_q), \mathcal{H}_c$

Figure 1: Effect of unwarping on a *relevant* query-corpus pair in Audio. $U_\phi(\cdot)$ learns to transform $\mathcal{H}_q$ in order to capture a high value of its latent similarity with $\mathcal{H}_c$.

$\Delta_t(U_\phi(\mathcal{H}_q),\mathcal{H}_c)$; (ii) our model with only model-dependent score, *i.e.*, $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c)$; (iii) our model without any model independent time similarity *i.e.*, $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c) - \gamma\Delta_x(\mathcal{H}_q,\mathcal{H}_c)$; (iv) our model without any model independent mark similarity *i.e.*, $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c) - \gamma\Delta_t(U_\phi(\mathcal{H}_q),\mathcal{H}_c)$; (v) our model without unwarping function $U_\phi(\cdot)$; and (vi) the complete design of our model. In all cases, we used CROSSATTN-NEUROSEQRET.

Table 2 shows that the complete design of our model (variant (vi)) achieves the best performance. We further note that removing κ_{p_θ} from the score (variant (i)) leads to significantly poor performance. Interestingly, our model without any mark based similarity (variant (iv)) leads to better performance than the model without time similarity (variant (iii))—this could be attributed to the larger variance in query-corpus time distribution than the distribution of marks. Finally, we observe that the performance deteriorates if we do not use an unwarping function $U_\phi(\cdot)$ (variant (v)). Figure 1 illustrates the effect of $U_\phi(\cdot)$. It shows that $U_\phi(\cdot)$ is able to learn suitable transformation of the query sequence, which encapsulates the high value of latent similarity with the corpus sequence.

5.3 Results on retrieval efficiency

We compare our efficient sequence retrieval method given in Algorithm 1 against random hyperplane (RH) method (Appendix B in (Gupta et al. 2021b)) and three variants of our proposed training problem in Eq. (11). (i) Our(η_2,η_3) which sets $\eta_1 = 0$ and thus does not enforce even distribution of ± 1 in $\boldsymbol{\zeta}^c$; (ii) Our(η_1,η_3) which sets $\eta_2 = 0$ and thus tanh does not accurately approximate sign; (iii) Our(η_1,η_2) which sets $\eta_3 = 0$ and thus does not enforce $\boldsymbol{\zeta}^c$ to be compact and free of redundancy. Our(η_1,η_2,η_3) is the complete design which includes all trainable components. Figure 2 summarizes the results.

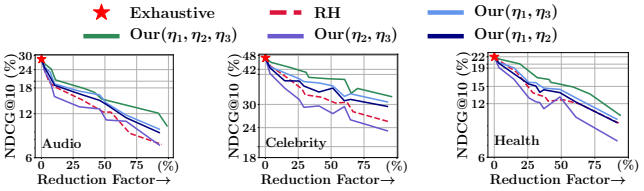


Figure 2: Tradeoff between NDCG@10 vs. Reduction factor, *i.e.*, % reduction in number of comparisons between query-corpus pairs w.r.t. the exhaustive comparisons for different hashing methods. The point marked as ⋆ indicates the case with exhaustive comparisons on the set of corpus sequences.

Comparison with random hyperplane. Figure 2 shows that our method (Our(η_1,η_2,η_3)) demonstrates better Pareto efficiency than RH. This is because RH generates hash code in a data oblivious manner whereas our method learns the hash code on top of the trained embeddings.

Ablation study on different components of Eq. (11). Figure 2 summarizes the results, which shows that (i) the first three variants are outperformed by Our(η_1,η_2,η_3); (ii) the first term having $\eta_1 \neq 0$, which enforces an even distribution of ± 1, is the most crucial component for the loss function—as the removal of this term causes significant deterioration of the performance.

6 Conclusions

In this paper, we proposed a novel supervised continuous time event sequence retrieval system called NEUROSEQRET using neural MTPP models. To achieve efficient retrieval over very large corpus of sequences, we also propose a trainable hash-coding of corpus sequences which can be used to narrow down the number of sequences to be considered for similarity score computation. Our experiments with real world datasets from a diverse range of domains show that both our retrieval model and hashing methods are more effective than several baselines.

References

Abanda, A.; Mori, U.; and Lozano, J. A. 2019. A review on distance based time series classification. In *DMKD*.

Alaee, S.; Kamgar, K.; and Keogh, E. 2020. Matrix Profile XXII: Exact Discovery of Time Series Motifs under DTW. In *ICDM*.

Blondel, M.; Mensch, A.; and Vert, J.-P. 2021. Differentiable Divergences Between Time Series. In *AISTATS*.

Cai, X.; Xu, T.; Yi, J.; Huang, J.; and Rajasekaran, S. 2019. DTWNet: a dynamic time warping network. In *NeurIPS*.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*.

Cuturi, M.; and Blondel, M. 2017. Soft-dtw: a differentiable loss function for time-series. In *ICML*.

Daley, D. J.; and Vere-Jones, D. 2007. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media.

De, A.; Bhattacharya, S.; and Ganguly, N. 2018. Demarcating endogenous and exogenous opinion diffusion process on social networks. In *Proceedings of the 2018 World Wide Web Conference*, 549–558.

De, A.; Valera, I.; Ganguly, N.; Bhattacharya, S.; and Gomez-Rodriguez, M. 2016. Learning and Forecasting Opinion Dynamics in Social Networks. In *NeurIPS*.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*.

Du, N.; Farajtabar, M.; Ahmed, A.; Smola, A. J.; and Song, L. 2015. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *KDD*.

Farajtabar, M.; Yang, J.; Ye, X.; Xu, H.; Trivedi, R.; Khalil, E.; Li, S.; Song, L.; and Zha, H. 2017. Fake news mitigation via point process based intervention. In *ICML*.

Gervini, D.; and Gasser, T. 2004. Self-modelling warping functions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4): 959–971.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity Search in High Dimensions via Hashing. In *VLDB*.

Gogolou, A.; Tsandilas, T.; Echihabi, K.; Bezerianos, A.; and Palpanas, T. 2020. Data series progressive similarity search with probabilistic quality guarantees. In *SIGMOD*.

Guo, R.; Li, J.; and Liu, H. 2018. INITIATOR: Noise-contrastive Estimation for Marked Temporal Point Process. In *IJCAI*.

Gupta, V.; and Bedathur, S. 2021. Region Invariant Normalizing Flows for Mobility Transfer. In *CIKM*.

Gupta, V.; Bedathur, S.; Bhattacharya, S.; and De, A. 2021a. Learning Temporal Point Processes with Intermittent Observations. In *International Conference on Artificial Intelligence and Statistics*, 3790–3798. PMLR.

Gupta, V.; Bedathur, S.; and De, A. 2021b. Supplementary Material for Learning Temporal Point Processes for Efficient Retrieval of Continuous Time Event Sequences. https://github.com/data-iitd/neuroseqret/.

Jaakkola, T. S.; Haussler, D.; et al. 1999. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, 487–493.

Jing, H.; and Smola, A. J. 2017. Neural survival recommender. In *WSDM*.

Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. 133–142. ACM.

Kang, W.-C.; and McAuley, J. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*.

Li, J.; Wang, Y.; and McAuley, J. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*.

Likhyani, A.; Gupta, V.; Srijith, P.; Deepak, P.; and Bedathur, S. 2020. Modeling Implicit Communities from Geo-tagged Event Traces using Spatio-Temporal Point Processes. In *WISE*.

Liu, W.; Mu, C.; Kumar, S.; and Chang, S.-F. 2014. Discrete graph hashing.

Mei, H.; and Eisner, J. M. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*.

Mei, H.; Qin, G.; and Eisner, J. 2019. Imputing Missing Events in Continuous-Time Event Streams. In *ICML*.

Mueen, A.; and Keogh, E. 2016. Extracting optimal performance from dynamic time warping. In *KDD*.

Mueen, A.; Zhu, Y.; Yeh, M.; Kamgar, K.; Viswanathan, K.; Gupta, C.; and Keogh, E. 2017. The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance. Http://www.cs.unm.edu/ mueen/FastestSimilaritySearch.html.

Müller, M. 2007. Dynamic time warping. *Information retrieval for music and motion*, 69–84.

Paparrizos, J.; and Gravano, L. 2015. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*.

Qin, Z.; Bai, Y.; and Sun, Y. 2020. GHashing: Semantic Graph Hashing for Approximate Similarity Search in Graph Databases. In *KDD*.

Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; and Keogh, E. 2012. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In *KDD*.

Rizoiu, M.-A.; Xie, L.; Sanner, S.; Cebrian, M.; Yu, H.; and Van Hentenryck, P. 2017. Expecting to be hip: Hawkes intensity processes for social media popularity. In *WWW*.

Roy, I.; De, A.; and Chakrabarti, S. 2020. Adversarial Permutation Guided Node Representations for Link Prediction. *arXiv preprint arXiv:2012.08974*.

Saha, A.; Samanta, B.; Ganguly, N.; and De, A. 2018. Crpp: Competing recurrent point process for modeling visibility dynamics in information diffusion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 537–546.

Salakhutdinov, R.; and Hinton, G. 2009. Semantic hashing. In *International Journal of Approximate Reasoning*.

Samanta, B.; De, A.; Chakraborty, A.; and Ganguly, N. 2017. LMPP: a large margin point process combining reinforcement and competition for modeling hashtag popularity. In *IJCAI*.

Sewell, M. 2011. The fisher kernel: a brief review. *RN*, 11(06): 06.

Shchur, O.; Biloš, M.; and Günnemann, S. 2020. Intensity-Free Learning of Temporal Point Processes. In *ICLR*.

Shelton, C. R.; Qin, Z.; and Shetty, C. 2018. Hawkes Process Inference with Missing Data. In *AAAI*.

Shen, Y.; Chen, Y.; Keogh, E.; and Jin, H. 2018. Accelerating time series searching with large uniform scaling. In *SDM*.

Su, H.; Liu, S.; Zheng, B.; Zhou, X.; and Zheng, K. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB Journal*.

Tabibian, B.; Upadhyay, U.; De, A.; Zarezade, A.; Schölkopf, B.; and Gomez-Rodriguez, M. 2019. Enhancing human learning via spaced repetition optimization. *PNAS*.

Valera, I.; Gomez-Rodriguez, M.; and Gummadi, K. 2014. Modeling Diffusion of Competing Products and Conventions in Social Media. *arXiv preprint arXiv:1406.0516*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All You Need. In *NeurIPS*.

Wang, P.; Fu, Y.; Liu, G.; Hu, W.; and Aggarwal, C. 2017. Human Mobility Synchronization and Trip Purpose Detection with Mixture of Hawkes Processes. In *KDD*.

Wehenkel, A.; and Louppe, G. 2019. Unconstrained monotonic neural networks. In *NeurIPS*.

Xiao, S.; Farajtabar, M.; Ye, X.; Yan, J.; Song, L.; and Zha, H. 2017. Wasserstein Learning of Deep Generative Point Process Models. In *NeurIPS*.

Xu, H.; Carin, L.; and Zha, H. 2018. Learning registered point processes from idiosyncratic observations. In *ICML*.

Yoon, J.; Jarrett, D.; and van der Schaar, M. 2019. Time-series generative adversarial networks. In *NeurIPS*.

Zamani Dadaneh, S.; Boluki, S.; Yin, M.; Zhou, M.; and Qian, X. 2020. Pairwise Supervised Hashing with Bernoulli Variational Auto-Encoder and Self-Control Gradient Estimator. In *UAI*.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010. Self-taught hashing for fast similarity search. In *SIGIR*.

Zhang, P.; Iyer, R.; Tendulkar, A.; Aggarwal, G.; and De, A. 2021. Learning to Select Exogenous Events for Marked Temporal Point Process. *Advances in Neural Information Processing Systems*, 34.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020. Self-attentive Hawkes processes. *ICML*.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer Hawkes Process. In *ICML*.

Appendix
(Learning Temporal Point Processes for Efficient Retrieval of Continuous Time Event Sequences)

A Background: Marked Temporal Point Processes

Marked Temporal point processes (MTPP) are probabilistic generative models for continuous-time event sequences. MTPP can be represented as a probability distribution over sequences of variable lengths belonging to a time interval $[0, T]$. Therefore, they can be realized as an event sequence $\mathcal{S}_k = \{(t_1, m_1), \cdots (t_N, m_N)\}$, where N is the number of events. Here, the times are ever-increasing *i.e.* $0 < t_1 < \cdots < t_N \leq T$ and $m_i \in \mathcal{C}$ is the corresponding mark with $\mathcal{C}$ as the set of all categorical marks. A MTPP can be characterized by its conditional intensity function, $\lambda^*(t)$. The $*$ denotes a dependence on the history. Given the conditional intensity function, we can obtain the conditional probability density function (PDF) as:

$$p^*(\Delta_{t,i}) = \lambda^*(t_{i-1} + \Delta_{t,i}) \exp\big(-\int_0^{\Delta_{t,i}} \lambda^*(t_{i-1} + r)dr\big) \tag{12}$$

where $\Delta_{t,i}$ denotes the inter-event time interval *i.e.*, $t_i - t_{i-1}$.

In recent years, neural enhancements to MTPP models have significantly enhanced the predictive power of these models. Specifically, they combine the continuous-time approach from the point process with deep learning approaches and thus can better capture complex relationships between events. The most popular approaches (Du et al. 2016; Mei and Eisner 2017; Shchur et al. 2020; Zhang et al. 2020; Zuo et al. 2020) use different methods to model the time- and mark-distribution via neural networks. Specifically, (Du et al. 2016) embeds the event history to a vector representation via a recurrent encoder that updates its state after parsing each event in a sequence; (Mei and Eisner 2017) modified the LSTM architecture to employ a continuous-time state evolution; (Shchur et al. 2020) replaced the intensity function with a mixture of *Log-Normal* flows for closed-form sampling; (Zhang et al. 2020) utilized the transformer architecture(Vaswani et al. 2017) to capture the long-term dependencies between events in the history embedding and (Zuo et al. 2020) used the transformer architecture for sequence embedding but extended it to graph settings as well. However, these models were designed to capture the generative distribution of future events in sequences, rather than the relevance between sequences. Thus, these models cannot be extended to the problem of sequence retrieval.

B Hashing

B.1 Random hyperplane based hashing method

Since the relevance between the query and corpus sequence pairs $(\mathcal{H}_q, \mathcal{H}_c)$ is measured using the cosine similarity between the gradient vectors, *i.e.*, $\kappa_{p_\theta}(\mathcal{H}_q, \mathcal{H}_c)$, one can use random hyperplane based locality sensitive hashing method for hashing the underlying gradient vectors $\boldsymbol{v}_{p_\theta}(\mathcal{H}_c)$ (Charikar 2002). Towards this goal, after training SELFATTN-NEUROSEQRET we generate R unit random vectors $\boldsymbol{u}_r \in \mathbb{R}^D$ from i.i.d. Normal distributions and then compute a binary hash code $\boldsymbol{\zeta}^c = [\text{sign}(\boldsymbol{u}_1^\top \boldsymbol{v}_{p_\theta}(\mathcal{H}_c)), \ldots, \text{sign}(\boldsymbol{u}_R^\top \boldsymbol{v}_{p_\theta}(\mathcal{H}_c))]$ for each $c \in \mathcal{C}$. This leads to 2^R possible hash buckets $\{\mathcal{B}\}$, where each corpus sequence is assigned to one hash bucket using the algorithm proposed by Gionis et al. (1999).

When we encounter an unseen test query $\mathcal{H}_q$, we compute the corresponding hash code $\boldsymbol{\zeta}^q$, assign it to a bucket $\mathcal{B}$ and finally return *only those sequences* $\mathcal{H}_c$ which were assigned to this bucket $\mathcal{B}$. Thus, for each query, the number of comparisons is reduced from $|\mathcal{C}|$ to $|\mathcal{B}|$, *i.e.*, the number of corpus sequences in the bucket $\mathcal{B}$. Thus, if the corpus sequences are assigned uniformly across the different buckets, then the expected number of comparisons becomes $|\mathcal{C}|/2^R$, which provides a significant improvement for $R > 2$.

Limitations. In practice, binary hash codes are not trained from data and consequently, they are not optimized to be uniformly distributed across different hash buckets. Consequently, the assignment of corpus sequences across different buckets may be quite skewed, leading to inefficient sequence retrieval.

B.2 Details about our proposed hashing method

As suggested in (Gionis et al. 1999), we design multiple hash tables and assign a bucket to the hashcode of a sequence using only a set of bits selected randomly. More specifically, let the number of hash-tables be M. Given a query sequence, we calculate its hashcode using the procedure described in Algorithm 1, $\boldsymbol{\zeta}^{q'} = \text{sign}\left(\Lambda_\psi(\boldsymbol{v}^{q'})\right)$. The hash code is a R dimension vector with $\boldsymbol{\zeta}^{q'} \in \{-1, 1\}^R$ and from this vector, we consider L bits at random positions to determine the bucket to be assigned to the sequence. Here, $\boldsymbol{\zeta}^{q'}$ represent the numbers between $\{0, 2^L - 1\}$, *i.e.*, one of the 2^L different buckets in a hash table. Correspondingly, we assign $\boldsymbol{\zeta}^{q'}$ into a bucket. However, such a procedure is dependent on the specific set of bits –that were selected randomly– used for deciding the bucket-ID. Therefore, we use M hash-tables and repeat the procedure of sampling L bits and bucket assignment for each table. We follow a similar bucket assignment procedure for corpus sequences. As described in Algorithm 1, for an incoming query sequence in the test set, we use the above bucket assignment procedure and compute the relevance score for only the corpus sequences within the same buckets. For all our experiments we set H same as the hidden dimension D, $M = 10$, and $L = 12$.

Dataset	Audio	Celebrity	Electricity	Health	Sports
$\lvert\mathcal{C}_{q+}\rvert/\lvert\mathcal{C}\rvert$	0.25	0.23	0.20	0.28	0.30
Total Events	1M	50M	60M	60M	430k
# Marks	5	16	5	5	21

Table 3: Statistics of the search corpus for all datasets. $|\mathcal{C}_{q+}|/|\mathcal{C}|$ denotes the ratio of positive corpus sequences to the total sequences sampled for training. The ratio is kept same for all queries.

C Additional details about the experimental setup

In this appendix, we elaborate on the details of dataset characteristics, evaluation metrics, and hardware configuration.

C.1 Dataset Statistics

We evaluate the retrieval performance of NEUROSEQRET and other methods across large-scale real-world datasets with up to 60 million events. The statistics of all datasets are given in Table 3. Across all datasets, $|\mathcal{H}_q| = 5K$ and $|\mathcal{H}_c| = 200K$. We partition the set of queries into 50% training, 10% validation, and the rest as test sets. During training, we negatively sample 100 corpus sequences for each query.

(1) **Audio:** The dataset contains audio files for spoken commands to a smart-light system and the demographics(age, nationality) of the speaker. Here, a query corpus sequence pair is relevant if they are from an audio file with a common speaker.

(2) **Sports:** The dataset contains actions (*e.g.*run, pass, shoot) taken while playing different sports. We consider the time of action and action class as time and mark of sequence respectively. Here, a query corpus sequence pair is relevant if they are from a common sport.

(3) **Celebrity:** In this dataset, we consider the series of frames extracted from youtube videos of multiple celebrities as event sequences where event-time denotes the video-time and the mark is decided upon the coordinates of the frame where the celebrity is located. Here, a query corpus sequence pair is relevant if they are from a video file having a common celebrity.

(4) **Electricity:** This dataset contains the power-consumption records of different devices across smart-homes in the UK. We consider the records for each device as a sequence with event mark as the *normalized* change in the power consumed by the device and the time of recording as event time. Here, a query corpus sequence pair is relevant if they are from a similar appliance.

(5) **Health:** The dataset contains ECG records for patients suffering from heart-related problems. Since the length of the ECG record for a single patient can be up to 10 million, we generate smaller individual sequences of length 10,000 and consider each such sequence as an independent sequence. The marks and times of events in a sequence are determined using a similar procedure as in Electricity. Here, a query corpus sequence pair is relevant if they are from a common patient.

For Health, Celebrity and Electricity, we lack the true ground-truth labeling of relevance between sequences. Therefore, we adopt a heuristic in which, given a dataset $\mathcal{D}$, from each sequence $\texttt{seq}_q \in \mathcal{D}$ with $q \in [|\mathcal{D}|]$, we first sample a set of sub-sequences $\mathcal{U}_q = \left\{\mathcal{H} \subset \texttt{seq}_q\right\}$ with $|\mathcal{U}_q| \sim \text{Unif}\,[200, 300]$. For each such collection $\mathcal{U}_q$, we draw exactly one query $\mathcal{H}_q$ uniformly at random from $\mathcal{U}_q$, *i.e.*, $\mathcal{H}_q \sim \mathcal{U}_q$. Then, we define $\mathcal{C} = \cup_{q\in[|\mathcal{D}|]}\mathcal{U}_q\backslash\mathcal{H}_q$, $\mathcal{C}_{q+} = \mathcal{U}_q\backslash\mathcal{H}_q$ and $\mathcal{C}_{q-} = \cup_{c\neq q}\big(\mathcal{U}_c\backslash\mathcal{H}_c\big)$.

C.2 System Configuration

All our models were implemented using Pytorch v1.6.0 [1]. We conducted all our experiments on a server running Ubuntu 16.04, CPU: Intel(R) Xeon(R) Gold 6248 2.50GHz, RAM: 377GB, and GPU: NVIDIA Tesla V100.

C.3 Hyperparameters setup

We set the hyper-parameters values of NEUROSEQRET as follows: (i) contribution of model-independent similarity score in Eq. (6), $\gamma = 0.1$; (ii) margin parameters for parameter estimation, $\delta \in \{0.1, 0.5, 1\}$ and weight for constraint violations, $\lambda \in \{0.1, 0.5, 1\}$; (iii) weight parameters for hashing objective (11) $\eta_1, \eta_2, \eta_3 \in \{0.1, 0.2, 0.25\}$ and correspondingly $\eta_4 \in \{0.25, 0.4, 0.7\}$.

Parameters	Datasets				
	Audio	Celebrity	Electricity	Health	Sports
γ	0.1	0.1	0.5	0.1	0.1
δ	0.5	0.5	0.1	0.1	0.5
$\{\eta_1, \eta_2, \eta_3\}$	{0.4, 0.3, 0.3}	{0.4, 0.3, 0.3}	{0.4, 0.3, 0.3}	{0.5, 0.25, 0.25}	{0.5, 0.25, 0.25}
Batch-size $\mathcal{B}$	32	32	32	16	16
D	64	64	48	32	32

Table 4: Hyper-parameter values used for different datasets. The values are determined by fine-tuning the performance on the validation set.

[1] https://pytorch.org/

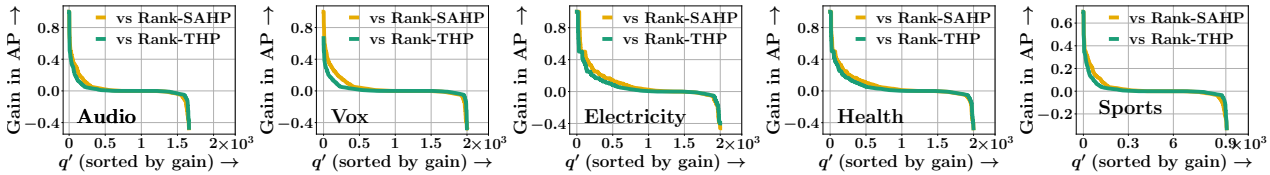


Figure 3: Query-wise performance comparison between NEUROSEQRET and best baseline methods – `Rank`-THP, `Rank`-SAHP. Queries are sorted by the decreasing gain in AP.

Moreover, the values of training specific parameter values are: (i) batch-size, $\mathcal{B}$ is selected from $\{16, 32\}$, *i.e.*for each batch we select $\mathcal{B}$ query sequences and all corresponding corpus sequences; (ii) hidden-layer dimension for cross-attention model, $D \in \{32, 48, 64\}$; (iii) number of attention blocks $N_b = 2$; (iv) number of attention heads $N_h = 1$ and (v) UMNN network as a 2 layer feed-forward network with dimension $\{128, 128\}$. We also add a dropout after each attention layer with probability $p = 0.2$ and an l_2 regularizer with over the trainable parameters with the coefficient set to 0.001. All our parameters are learned using the Adam optimizer. We summarize the details of hyperparameters across different datasets in Table 4.

C.4 Evaluation metrics

We evaluate NEUROSEQRET and the baselines using mean average precision (MAP), NDCG@k, and mean reciprocal rank (MRR). We calculate these metrics as follows:

$$\text{MAP} = \frac{1}{|\mathcal{H}_{q'}|} \sum_{q' \in \mathcal{H}_{q'}} \text{AP}_{q'}, \quad \text{NDCG@k} = \frac{\text{DCG}_k}{\text{IDCG}_k}, \quad \text{MRR} = \frac{1}{|\mathcal{H}_{q'}|} \sum_{q' \in \mathcal{H}_{q'}} \frac{1}{r_{q'}}, \tag{13}$$

where $\text{AP}_{q'}, \text{DCG}_k, \text{IDCG}_k$, and $r_{q'}$ denote the average precision, discounted cumulative gain at top-k position, ideal discounted cumulative gain (at top-k), and the topmost rank of a related corpus sequence respectively. For all our evaluations, we follow a standard evaluation protocol (Kang and McAuley 2018; Li et al. 2020) for our model and all baselines wherein for each query sequence in the test set, we rank all relevant corpus sequence and 1000 randomly sampled non-relevant sequences. All confidence intervals and standard deviations are calculated after 5 independent runs. For all metrics – MAP, NDCG, and MRR, we report results in terms of percentages with respect to maximum possible value *i.e.*1.

C.5 Baseline Implementations

For all the baselines, we use the official python implementations released by the authors of MASS [2], UDTW [3], Sharp [4], RMTPP [5], SAHP [6], and THP [7] and we thank them for making their codes public. For MASS and UDTW, we report the results using the default parameter values. For Sharp, we tune the hyper-parameter '*gamma*' (for more details see (Blondel et al. 2021)) based on the validation set. In RMTPP, we set the BPTT length to 50, the RNN hidden layer size to 64, and the event embedding size 16. These are the parameter values recommended by the authors. For SAHP and THP, we set the dimension to 128 and the number of heads to 2. The values for all other transformer parameters are similar to the one we used for the attention-part in NEUROSEQRET.

D Additional experiments with real data

D.1 Analysis of Retrieval Accuracy

In addition to results in Table 1, we evaluate the performance of NEUROSEQRET and all baselines through mean reciprocal rank (MRR) and NDCG@20, given in Table 5 and Table 6 respectively. These results show that NEUROSEQRET outperforms all other baseline models across different evaluation metrics. Moreover, in contrast to Table 1, we note that NEUROSEQRET outperforms `Rank`-THP in both MRR and NDCG@20 metrics.

D.2 Analysis at a query level

Next, we compare the performance between NEUROSEQRET and other state-of-the-art methods, at a query level. Specifically, for each query $\mathcal{H}_q$ we compute the advantage of using NEUROSEQRET in terms of gain in average precision, i.e., AP(NEUROSEQRET) − AP(baseline) for two most competitive baselines – `Rank`-SAHP and `Rank`-THP. We summarize the results in Figure 3, which show that for at least 70% of the queries, NEUROSEQRET outperforms or fares competitively with these baselines, across all datasets.

[2] https://www.cs.unm.edu/∼mueen/MASS.py
[3] https://github.com/klon/ucrdtw
[4] https://github.com/google-research/soft-dtw-divergences
[5] https://github.com/Networks-Learning/tf_rmtpp
[6] https://github.com/QiangAIResearcher/sahp_repo
[7] https://github.com/SimiaoZuo/Transformer-Hawkes-Process

Dataset	Mean Reciprocal Rank (MRR)				
	Audio	Celebrity	Electricity	Health	Sports
MASS (Mueen et al. 2017)	57.3±0.0	63.7±0.0	17.6±0.0	27.2±0.0	61.6±0.0
UDTW (Rakthanmanon et al. 2012)	58.5±0.0	64.8±0.0	18.7±0.0	29.2±0.0	61.2±0.0
Sharp (Blondel et al. 2021)	58.7±1.7	65.4±2.6	19.8±0.6	30.4±0.7	61.1±2.3
RMTPP (Du et al. 2016)	54.2±3.9	64.5±4.6	15.8±1.2	25.2±1.8	56.2±4.8
Rank-RMTPP	60.8±3.7	65.6±4.0	23.3±1.5	30.7±1.7	62.7±3.9
SAHP (Zhang et al. 2020)	56.9±4.3	64.2±4.8	17.2±1.3	26.7±2.1	59.4±4.9
Rank-SAHP	60.3±2.5	66.1±2.9	25.9±1.2	32.6±1.4	62.1±2.9
THP (Zuo et al. 2020)	58.6±2.6	65.0±2.9	20.1±1.1	28.2±1.3	60.9±3.3
Rank-THP	62.2±2.8	68.9±3.0	31.4±1.4	36.2±1.7	63.4±2.9
SELFATTN-NEUROSEQRET	63.6±2.7	69.3±3.1	33.4±1.6	37.9±1.7	64.3±3.1
CROSSATTN-NEUROSEQRET	**64.5±2.9**	**70.1±3.3**	**35.2±1.7**	**40.3±1.9**	**66.7±3.1**

Table 5: Retrieval quality in terms of mean reciprocal rank(MRR in %) of all the methods across five datasets on the test set. Numbers with bold font (underline) indicate best (second best) performer. Boxed numbers indicate best performing state-of-the-art baseline.

Dataset	NDCG@20				
	Audio	Celebrity	Electricity	Health	Sports
MASS (Mueen et al. 2017)	17.5±0.0	31.4±0.0	8.1±0.0	13.5±0.0	16.3±0.0
UDTW (Rakthanmanon et al. 2012)	17.9±0.0	32.5±0.0	8.8±0.0	14.4±0.0	16.0±0.0
Sharp (Blondel et al. 2021)	18.2±0.5	33.6±0.7	11.9±0.3	15.9±0.4	17.2±0.7
RMTPP (Du et al. 2016)	16.0±1.1	32.2±1.6	7.1±0.5	12.1±0.7	17.1±1.1
Rank-RMTPP	20.2±1.0	33.4±1.4	10.5±0.5	15.4±0.6	21.8±1.2
SAHP (Zhang et al. 2020)	19.8±1.4	31.7±2.1	7.8±0.5	13.1±0.9	19.2±1.5
Rank-SAHP	21.5±0.9	34.1±1.0	12.3±0.4	17.4±0.6	22.9±1.0
THP (Zuo et al. 2020)	19.7±0.8	32.9±1.0	9.5±0.4	14.4±0.5	20.8±0.9
Rank-THP	21.8±0.9	37.7±1.2	14.4±0.6	19.3±0.6	23.3±1.1
SELFATTN-NEUROSEQRET	22.9±1.3	40.3±1.5	16.3±0.7	20.9±1.0	23.8±1.6
CROSSATTN-NEUROSEQRET	**24.2±1.5**	**42.0±1.8**	**17.6±0.8**	**22.3±1.0**	**25.7±1.7**

Table 6: Retrieval quality in terms of NDCG@20 (in %) of all the methods across five datasets on the test set. Numbers with bold font (underline) indicate best (second best) performer. Boxed numbers indicate best performing state-of-the-art baseline.

Variant	Electricity	Sports
(i) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = -\Delta_x(\mathcal{H}_q,\mathcal{H}_c) - \Delta_t(U_\phi(\mathcal{H}_q),\mathcal{H}_c)$	18.9±0.0	41.3±0.0
(ii) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c)$	30.6±0.9	56.3±2.1
(iii) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c) - \gamma\Delta_x(\mathcal{H}_q,\mathcal{H}_c)$	30.8±0.8	55.6±2.0
(iv) $s_{p_\theta,U_\phi}(\mathcal{H}_q,\mathcal{H}_c) = \kappa_{p_\theta}(\mathcal{H}_q,\mathcal{H}_c) - \gamma\Delta_t(U_\phi(\mathcal{H}_q),\mathcal{H}_c)$	31.3±0.8	58.1±2.0
(v) CROSSATTN-NEUROSEQRET Without $U_\phi(\cdot)$	29.7±1.3	56.2±2.3
(vi) CROSSATTN-NEUROSEQRET	32.4±0.8	58.7±2.1

Table 7: Ablation study of CROSSATTN-NEUROSEQRET and its variants in terms of MAP (in %).

Run Time	Audio	Celebrity	Electricity	Health	Sports
NEUROSEQRET	$\leq$ 3hr	$\leq$ 5hr	$\leq$ 6hr	$\leq$ 6hr	$\leq$ 2hr

Table 8: Training-times of NEUROSEQRET for all datasets.

D.3 Ablation Study

We also perform an ablation study for Electricity and Sports datasets in Table 7, which reveal similar insights as in Table 2,

D.4 Comparison with Random Hyperplane

We also perform an ablation study of our efficient retrieval method along with its comparison against RH for Electricity and Sports datasets. The results in Figure. 4, reveal similar insights as in Figure 2.

D.5 Runtime Analysis

Next, we calculate the run-time performance of NEUROSEQRET. With this experiment, our goal is to determine if the training times of NEUROSEQRET are suitable for designing solutions for real-world problems. From the results in Table 8, we note that even for datasets with up to 60 million events, the training times are well within the feasible range for practical deployment.

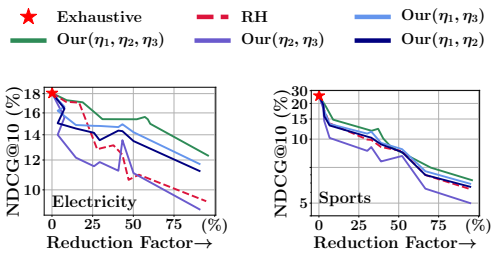


Figure 4: Tradeoff between NDCG@10 vs. Reduction factor, *i.e.*, % reduction in number of comparisons between query-corpus pairs w.r.t. the exhaustive comparisons for different hashing methods. The point marked as $\star$ indicates the case with exhaustive comparisons on the set of corpus sequences.

$\lvert\mathcal{H}_q\rvert = (10, 20)$	**Audio**	**Celebrity**	**Health**	**Electricity**	**Sports**
`Rank`-THP	18.97	22.06	9.48	12.27	26.58
CROSSATTN-NEUROSEQRET	21.30	25.77	15.83	10.79	27.63

Table 9: Retrieval quality in terms of mean average precision (MAP) for query sequence lengths sampled between 10 and 50.

$\lvert\mathcal{H}_q\rvert = (50, 100)$	**Audio**	**Celebrity**	**Health**	**Electricity**	**Sports**
`Rank`-THP	27.94	37.85	17.58	21.61	31.26
CROSSATTN-NEUROSEQRET	28.58	41.92	19.67	23.54	36.92

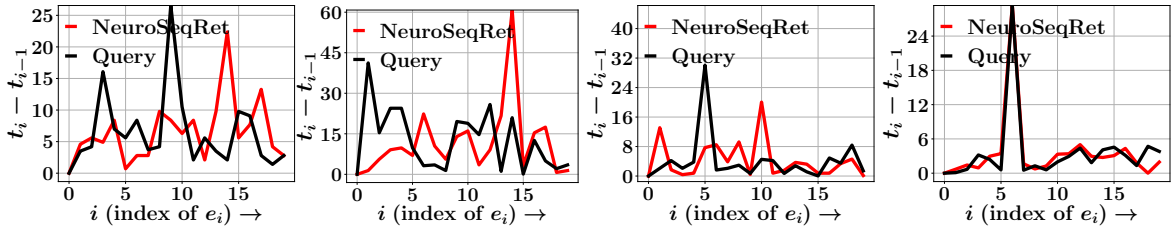
Table 10: Retrieval quality in terms of mean average precision (MAP) for query sequence lengths sampled between 50 and 100.

D.6 Query Length

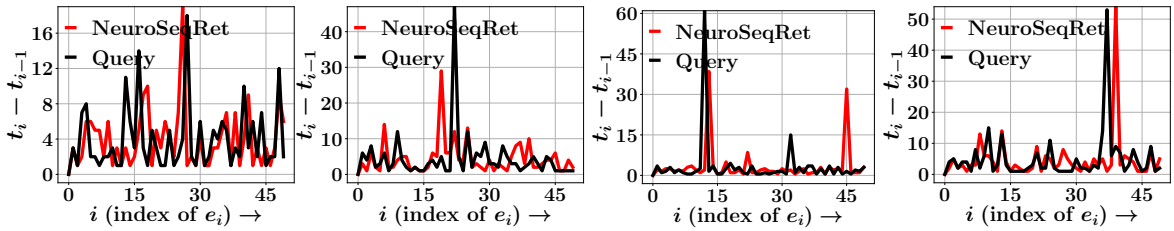
We perform an additional experiment of sequence retrieval with varying query lengths. Specifically, we sample queries of lengths $|\mathcal{H}_q| \sim Unif(10, 50)$ and $|\mathcal{H}_q| \sim Unif(50, 100)$ and report the sequence retrieval results in Table 9 and Table 10 respectively. The results show that the performance of all models deteriorates significantly as we reduce the length of query sequences. They also show that even with smaller query lengths, CROSSATTN-NEUROSEQRET significantly outperforms the other state-of-the-art baseline `Rank`-THP.

D.7 Qualitative Analysis

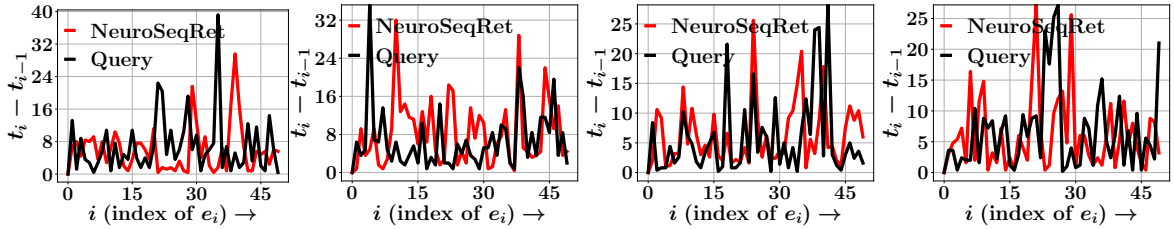
To get deeper insights into the working of our model, we perform a qualitative analysis between a query sequence from the dataset and the sequence retrieved by NEUROSEQRET. More specifically, we aim to understand the similar patterns between query and corpus sequences that NEUROSEQRET searches for in the corpus and plot the query sequence and the corresponding top-ranked relevant corpus sequence retrieved by NEUROSEQRET. The results across all datasets in Figure 5 show that the inter-arrival times of the CTES retrieved by NEUROSEQRET closely matches with the query inter-arrival times.



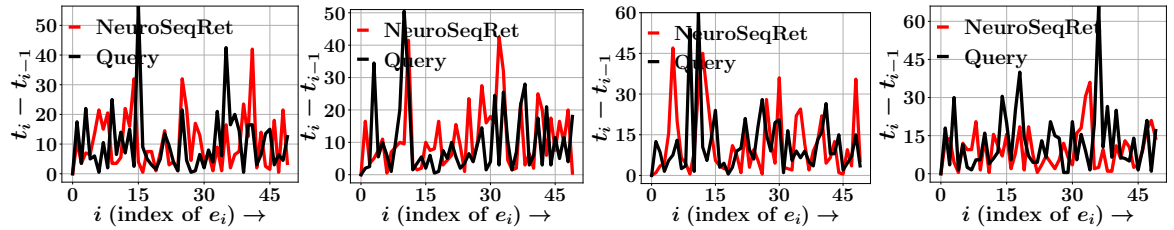
(a) Audio dataset



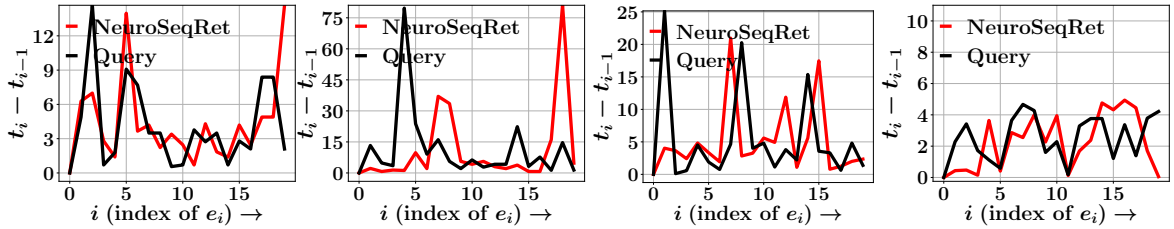
(b) Celebrity dataset



(c) Electricity dataset



(d) Health dataset



(e) Sports dataset

Figure 5: Qualitative examples of inter-event times of events in a query sequence and the top-search results by NEUROSEQRET for all datasets.