Self-Supervised Pretext Tasks for Event Sequence Data from Detecting Misalignment

Yimu Wang *
University of Waterloo
yimu.wang@uwaterloo.ca

He Zhao
RBC Borealis
he.zhao@borealisai.com

Ruizhi Deng
RBC Borealis
ruizhi.deng@borealisai.com

Frederick Tung
RBC Borealis
frederick.tung@borealisai.com

Greg Mori
RBC Borealis
greg.mori@borealisai.com

Abstract

Pretext training followed by task-specific fine-tuning has been a successful approach in vision and language domains. This paper proposes a self-supervised pretext training framework tailored to event sequence data. We introduce novel auxiliary tasks (pretext tasks) that encourage the network to learn the coupling relationships between event times and types – a previously untapped source of self-supervision without labels. These pretext tasks unlock foundational representations that are generalizable across different downstream tasks, including next-event prediction for temporal point process models, event sequence classification, and missing event interpolation. Experiments on popular public benchmarks demonstrate the potential of the proposed method across different tasks and data domains.

1 Introduction

Self-supervised learning (SSL) has emerged as a transformational machine learning paradigm for foundation models in vision (He et al., 2022), language (Brown et al., 2020), and time series (Yue et al., 2022). The core to its success is designing effective auxiliary (pretext) tasks, *e.g.,* reconstruction of masked information (He et al., 2022; Devlin et al., 2018), which help extract general representations from unlabelled data.

Event sequences are prevalent in many domains, including commerce, science, and healthcare, with event data generated by human activities and natural phenomena such as online purchases, banking transactions, earthquakes, disease outbreaks, and hospital patients' medical observations. When event data are plentiful, large-scale pretext training can enable the learning of general representations without labels toward the building of foundational capabilities. For example, pretext training on banking transactions could help produce representations that are useful for detecting financial crime, for which labels are scarce. Nevertheless, self-supervised pretext tasks for event sequences remain underexplored.

In this work, we present novel pretext training tasks for event sequences inspired by the coupling relationship between event times and event types, and observations on alignment between different pieces of event sequences. For example, in Figure 1, a user's lunch break is usually aligned with noon

*Work done during internship at RBC Borealis.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

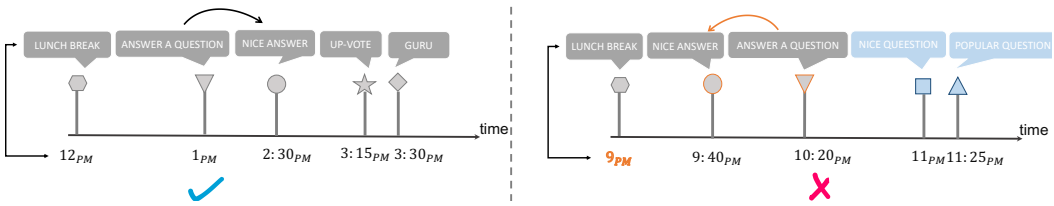


Figure 1: The figure shows an example of a valid event sequence of a person's daily events and online activities on the left side compared against a misaligned sequence on the right side. In the misaligned sequence, the activity of a lunch break is misaligned with its time stamp of 9 pm. The person could not receive a `NICE ANSWER` badge for his answer in an online forum before answering the question. The orange arrow highlights the causal relation between the two events. It is also unlikely for this internet forum user to receive a `NICE QUESTION` badge for answering a question.

times instead of bedtime. The user must answer a question in an internet forum before getting a badge of `NICE ANSWER`. Receiving a `NICE ANSWER` badge before answering a question would break our assumptions on the order of these two events even though they are closely associated. The user would then receive a prestigious badge of `GURU` as the answer receives more up-votes but awarding the user with a `POPULAR QUESTION` badge for his answer would create incoherence with the previous events.

To capture these alignment relationships from event sequences in a fully self-supervised manner, we devise three approaches to create disordered or mixed-up sequences automatically during training. The model is trained to detect these misaligned sequences with a binary classification task. This pretext training unlocks representations that are generalizable across different downstream tasks, including next-event prediction for temporal point process models, event sequence classification, and missing event interpolation. In addition, we verify that the proposed pretext tasks are complementary to masked autoencoding and contrastive learning pretext tasks that are popularly adopted in computer vision and natural language processing.

2 Approach

We explore the effectiveness of verifying alignment (*i.e.,* detecting disordered or mixed-up event sequences) as a way to perform self-supervised model pre-training without downstream labels. Once the general representations are learned, task-specific fine-tuning can be performed to adapt the model to downstream tasks of interest such as next-event prediction, sequence classification, and missing event interpolation.

2.1 Preliminaries

Definition of event sequences. Let $X = \{(t_1, m_1), ..., (t_N, m_N)\}$, $0 < t_1 < ... < t_N$ denote an event sequence, where N is the total number of events, $\{t_i\}$ are event arrival times and $\{m_i\} \in \{1, ..., K\}$ are categorical event types.

2.2 Alignment Verification as Auxiliary Task

We formulate the alignment verification pretext task as a binary classification problem, aiming to tell apart correctly aligned (original) event sequences from misaligned (disordered or mixed-up) sequences. To generate misaligned event sequences for pre-training, we take three approaches:

Misalignment 1: Shuffle. Randomly shuffling the event types both disrupts the consistency between event types and event arrival times and breaks their correct order. We keep the event arrival times intact so that they still follow the same distribution as the original sequences and the training focuses on learning better contextual representations of event types.

Misalignment 2: Swap. Similarly, we can create a misalignment by mixing event types of a given sequence with event times of another sequence. This can be done by randomly picking a different event sequence in the same batch and swapping their type and time dimensions[2].

[2]In implementation, we pad all sequences to the same length.

Algorithm 1 Methods for generating misaligned sequences in alignment verification

Misalignment 1: **Shuffle** (event sequence X):

1. Get the shuffled indices by $\tilde{ind} \leftarrow \texttt{shuffle}([N])$;
2. Return the event sequence shuffled along the type dimension $\tilde{X} = \{(t_1, m_{\tilde{ind}_1}), \ldots, (t_N, m_{\tilde{ind}_N})\}$.

Misalignment 2: Swap (event sequences X_a, X_b):
Return the swapped event sequences $\tilde{X}_a = \{(t_{[a,1]}, m_{[b,1]}), \ldots, (t_{[a,N]}, m_{[b,N]})\}$ and $\tilde{X}_b = \{(t_{[b,1]}, m_{[a,1]}), \ldots, (t_{[b,N]}, m_{[a,N]})\}$.

Misalignment 3: Crossover (event sequences X_a, X_b):
Assuming even lengths N_a, N_b for ease of presentation
Return the combined event sequences
$\tilde{X}_a = \{(t_{[a,1]}, m_{[a,1]}), \ldots, (t_{[a,N_a/2]}, m_{[a,N_a/2]}), (t_{[a,N_a/2]} + t_{[b,N_b/2+1]} - t_{[b,N_b/2]}, m_{[b,N_b/2+1]}), \ldots, (t_{[a,N_a/2]} + t_{[b,N_b]} - t_{[b,N_b/2]}, m_{[b,N_b]})\}$ and
$\tilde{X}_b = \{(t_{[b,1]}, m_{[b,1]}), \ldots, (t_{[b,N_b/2]}, m_{[b,N_b/2]}), (t_{[b,N_b/2]} + t_{[a,N_a/2+1]} - t_{[a,N_a/2]}, m_{[a,N_a/2+1]}), \ldots, (t_{[b,N_b/2]} + t_{[a,N_a]} - t_{[a,N_a/2]}, m_{[a,N_a]})\}$.

Misalignment 3: Crossover. The previous two approaches focus on creating misalignment on one feature dimension, *e.g.*, the event time. Here, we go one step further and adopt the crossover operator from genetic algorithms (Reeves, 2010) to generate misaligned data with incoherence in the first and second halves, by combining halves of two randomly selected sequences on both event type and time. We ensure monotonicity in arrival time values by crossing over the time intervals from the other sequence instead of the absolute arrival times t.

The three misalignment generation methods are summarized in Algorithm 1, and we apply them collectively during pre-training: given an input event sequence, we use each method to generate one misaligned sequence. We assign binary labels (i.e., **1 / 0**) to the original and misaligned sequences, and train a binary classifier on the aggregated representation. We average all the BCE losses as the final alignment loss. To represent the entire sequence with one vector, we append a special [EOS] token to the end of event sequences and use the output embeddings corresponding to [EOS] as the aggregated representation. Details are presented in Appendix A.1.

3 Experiments

3.1 Downstream tasks, datasets and evaluation protocol

To show the generality of our pretext tasks, we fine-tune the pre-trained model on three representative downstream tasks as described below.

The **temporal point process** (TPP) task models the probability distribution of event sequence data though a (conditional) intensity function, $\lambda(t)$, which gives the probability of the next event occurring during the interval $[t, t + \delta t)$, conditioned on the history H_t. The output of the network is a set of parameters to the intensity function. Following Mei et al. (2021), we model the intensity function as a softplus function, $1/\beta * \log(1 + \exp(\beta * x))$, where $x = \alpha * t + w * H_t + b$. The learnable parameters are $\{a, w, b\}$, where (a, w) fuse the time interval and the history vector information, and b is a base (or bias) term. We model a separate intensity function, $\lambda_k(t)$, for each event category k, and take their sum as the overall intensity $\lambda(t) = \sum_{k=1}^{K} \lambda_k(t)$. During fine-tuning, we minimize the negative log-likelihood.

Event sequence classification aims at predicting a sequence-level label. During fine-tuning, we minimize the binary cross-entropy loss (BCE).

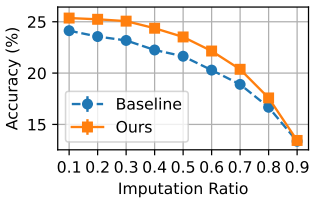
Missing events imputation aims at reconstructing missing events, and simulating partial observability in a noisy environment. During fine-tuning, we minimize the cross-entropy loss (CE) and root-mean-square error (RMSE) for the missing values of event type and time.

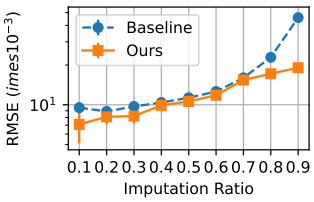
Benchmark datasets & evaluation protocols. We evaluate our proposed method on four real-world datasets: StackOverflow (Leskovec and Krevl, 2014), Mooc (Kumar et al., 2019), Reddit (Kumar et al., 2018), and MIMIC-II (Lee et al., 2011). Details are presented in Appendix A.3. For TPP, we report the negative log-likelihood (NLL) and root mean squared error (RMSE) for event time

Table 1: Results of TPP (NLL, RMSE, and accuracy) on Stack Overflow, MIMIC-II, Mooc, and Reddit datasets.

Method	Stack Overflow			MIMIC-II		
	↓NLL	↓RMSE	↑Accuracy	↓NLL	↓RMSE	↑Accuracy
ANHP Mei et al. (2021)	$2.16_{0.02}$	$\mathbf{1.19}_{0.01}$	$47.42_{0.00}$	$1.85_{0.06}$	$\mathbf{1.06}_{0.01}$	$\mathbf{85.64}_{0.00}$
OURS	$\mathbf{1.86}_{0.05}$	$1.36_{0.02}$	$\mathbf{47.96}_{0.01}$	$\mathbf{1.80}_{0.14}$	$1.19_{0.17}$	$84.84_{0.00}$

Method	MOOC			Reddit		
	↓NLL	↓RMSE	↑Accuracy	↓NLL	↓RMSE	↑Accuracy
ANHP Mei et al. (2021)	$-2.78_{0.02}$	$0.20_{0.01}$	$21.66_{0.01}$	$0.05_{0.03}$	$0.19_{0.00}$	$\mathbf{61.82}_{0.00}$
OURS	$\mathbf{-3.84}_{0.07}$	$\mathbf{0.19}_{0.00}$	$\mathbf{31.19}_{0.02}$	$\mathbf{-0.15}_{0.03}$	$0.19_{0.00}$	$58.19_{0.00}$





	Baseline	OURS
AUC	0.736	**0.748**

Figure 2 & Table 2: Left: Accuracy (higher is better) and RMSE (lower is better) of missing events imputation on the Mooc dataset. Baseline refers to OURS without pretext training. **Right**: AUC of sequence-level classification on the Mooc dataset. Best in **Bold**.

prediction, and accuracy for event type prediction. For binary classification, we report the area under the curve (AUC). For missing events imputation, we report accuracy and RMSE.

3.2 Empirical evaluation

Temporal point process. The experimental results on four real-world benchmark datasets are shown in Table 1. ANHP (Mei et al., 2021) shares the same backbone as ours and does not use pretext training. As the main performance metric for temporal point processes is NLL (TPP models produce probability distributions), our method shows consistent improvements on four different datasets. Meanwhile, our model achieves competitive scores in RMSE and accuracy (type prediction). The results demonstrate the effectiveness of pretext training in predicting future events on real-world data.

To give more insights, we further experiment on 1) combining our approach with other SSL tasks, and 2) few-shot pre-training/fine-tuning. See more results in Appendix A.4.

Sequence-level classification. The results of sequence-level classification are presented in Table 2. Each sequence in Mooc is associated with a course withdrawal meta-label (binary) and the overall label distribution is approximately balanced (57% positive vs. 43% negative). We use the same transformer architecture trained from scratch without pretext training as our baseline. Pretext training improves the AUC from 0.736 to 0.748. The result suggests that our alignment pre-training generalizes beyond temporal point processes and can also benefit sequence-level classification tasks.

Imputation. The results of missing events imputation are shown in Figure 2 for the Mooc dataset. Our model is fine-tuned on the imputation task using a 50% missing ratio and then tested on missing ratios from 10% to 90%; the baseline model is trained from scratch on the imputation task using a 50% missing ratio and then tested on missing ratios from 10% to 90%. We observe that pretext training improves imputation performance at all tested missing ratios. Imputation performance decreases as the missing ratio increases, reflecting the increasing difficulty of the task given less information.

4 Conclusion

Self-supervised learning enables the pre-training of generalizable representations, obviating the need to train models from scratch for different downstream tasks. In this paper, we presented alignment-based self-supervised pretext tasks for discrete event sequences. In vision and language, a common paradigm is to pre-train on large data and then specialize to smaller domains, for example via fine-tuning or prompt-tuning. Achieving this paradigm for event sequence data will require complementary progress in making large-scale data available for pre-training foundation models.

References

Amos, I., Berant, J., and Gupta, A. (2024). Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. In *International Conference on Learning Representations*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Chowdhury, R. R., Li, J., Zhang, X., Hong, D., Gupta, R. K., and Shang, J. (2023). Primenet: Pre-training for irregular multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7184–7192.

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

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 *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1555–1564.

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

Huang, P.-Y., Sharma, V., Xu, H., Ryali, C., Li, Y., Li, S.-W., Ghosh, G., Malik, J., Feichtenhofer, C., et al. (2024). Mavil: Masked audio-video learners. *Advances in Neural Information Processing Systems*, 36.

Kumar, S., Hamilton, W. L., Leskovec, J., and Jurafsky, D. (2018). Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 933–943. International World Wide Web Conferences Steering Committee.

Kumar, S., Zhang, X., and Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278. ACM.

Lee, J., Scott, D. J., Villarroel, M., Clifford, G. D., Saeed, M., and Mark, R. G. (2011). Open-access MIMIC-II database for intensive care research. In *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2011, Boston, MA, USA, August 30 - Sept. 3, 2011*, pages 8315–8318.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Mei, H., Yang, C., and Eisner, J. (2021). Transformer embeddings of irregularly spaced events and their participants. In *International conference on learning representations*.

Reeves, C. R. (2010). Genetic algorithms. *Handbook of metaheuristics*, pages 109–139.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

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

A Appendix

A.1 Pretext training details

The three misalignment generation methods are summarized in Algorithm 1 and visualized in Figure 3. In our implementation, we adopt the transformer architecture in Mei et al. (2021) for representation learning. Next, we briefly describe how event information is encoded under this setting.

Event time embeddings. Following recent advances, we use positional encodings (Vaswani et al., 2017; Mei et al., 2021) to transform each event arrival time into a vector, $\mathbf{e}^t \in \mathcal{R}^{D_{time}}$.

Event type embeddings. We simply encode categorical event types into a high dimensional vector, $\mathbf{e}^m \in \mathcal{R}^{D_{type}}$, with a learnable embedding layer, similar to Du et al. (2016).

The input representation of an event is the concatenation of $\mathbf{e} = [\mathbf{e}^m, \mathbf{e}^t]$.

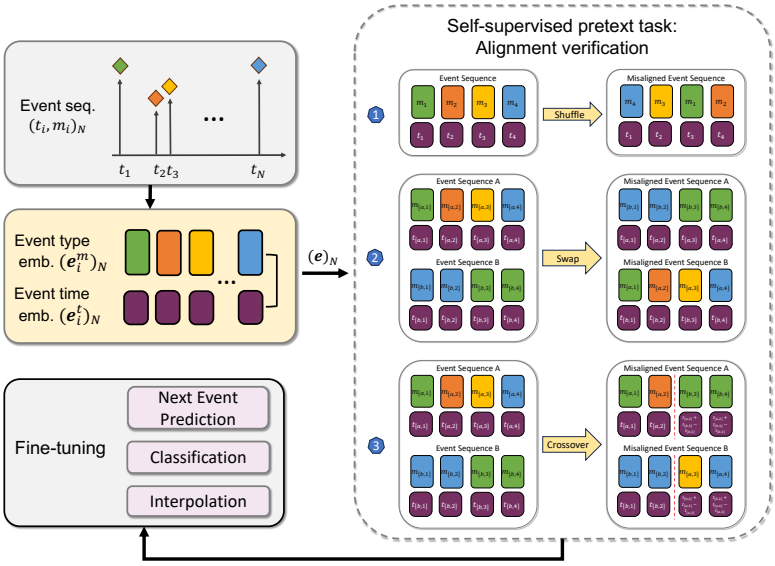


Figure 3: Overview of the pretext training and fine-tuning framework for event sequences. In the first stage, event sequence embeddings are input to self-supervised pretext tasks involving detecting misaligned (disordered or mixed-up) event sequences, to learn general representations without labels. The second stage performs task-specific fine-tuning to support downstream tasks such as next-event prediction, sequence classification, and missing event interpolation.

Aggregated embeddings for pretext. To represent the entire sequence with one vector, we append a special [EOS] token to the end of event sequences and use the output embeddings corresponding to [EOS], *i.e.*, $\mathbf{z}$, as the aggregated representation. The representation is input to an MLP classifier, which is trained using a binary cross-entropy loss to predict whether the sequence is an original or misaligned sequence.

A.2 Implementation details

We provide the details and hyper-parameters of the model architecture and training setting below. With the hyper-parameter below, each experiment would take an average of two days on a single Nvidia 1080ti GPU. **Architecture details.** We adopt the transformer encoder in Mei et al. (2021) as our backbone for representation learning. This module is configured with three encoder layers and each layer is multi-head attention implemented with four heads and feature dimensions as 64 ($D_{time} = D_{type} = 32$).

Pre-training details. During model pre-training, we directly use the downstream task data to incorporate data-driven priors, *c.f.,* Amos et al. (2024), and only optimize for the auxiliary task (alignment verification) as described in Section. 2.2. The learning rate is fixed at 1e-4, the number of epochs is 10, and the batch size is set to 4. Once pre-training is complete, the backbone acts as the

representation extractor and initial point for fine-tuning. At this stage, task-specific heads are added for each task.

Fine-tuning details. For every downstream task, we fine-tune the model and task head for 300 epochs with a batch size of 4 and a learning rate of 1e-4. The task heads are simple MLPs. For TPP, the task head yields parameterized intensity functions in Section 3.1. For event sequence classification, the head is applied on the embedding of [EOS] to predict the class probabilities. For imputation, the two MLP heads predict time (scalar regression) and type (classification).

Evaluation protocol. To evaluate the downstream fine-tuning, we baseline against the same architecture as ours trained from scratch for the downstream task (without pretext training). For TPP, we report the negative log-likelihood (NLL) and root mean squared error (RMSE) for event time prediction, and accuracy for event type prediction. For binary classification, we report the area under the curve (AUC). For missing events imputation, we report accuracy and RMSE.

A.3 Datasets

StackOverflow (SO, Link) (Leskovec and Krevl, 2014). It includes sequences of user awards within two years. StackOverflow is a question-answering website where users are awarded based on their proposed questions and their answers to questions proposed by others. This dataset contains a total of 6,633 sequences. There are 22 types of events: Nice Question, Good Answer, Guru, Popular Question, Famous Question, Nice Answer, Good Question, Caucus, Notable Question, Necromancer, Promoter, Yearling, Revival, Enlightened, Great Answer, Populist, Great Question, Constituent, Announcer, Stellar Question, Booster and Publicist. The award time records when a user receives an award. With this dataset, we can learn which type of awards will be given to a user and when.

Mooc (Kumar et al., 2019) (Link). It contains the interaction of students with an online course system. An interaction is an event and can be of various types (97 unique types), e.g. watching a video, solving a quiz etc.

Reddit (Kumar et al., 2018) (Link). On this social network website, users submit posts to subreddits. In the dataset, the most active subreddits are selected, and posts from the most active users on those subreddits are recorded. Each sequence corresponds to a list of submissions a user makes. The data contains 984 unique subreddits that we use as classes in mark prediction.

MIMIC-II (Lee et al., 2011) (Link). The Multiparameter Intelligent Monitoring in Intensive Care (MIMIC-II) dataset is developed based on an electric medical record system. The dataset contains a total of 650 sequences, each of which corresponds to an anonymous patient's clinical visits in a seven-year period. Each clinical event records the diagnosis result and the timestamp of that visit. The number of unique diagnosis results is 75. According to the clinical history, a temporal point process is supposed to capture the dynamics of when a patient will visit doctors and the diagnosis result.

A.4 Additional experiments

Compatibility with standard pretext tasks. The next question investigated is how the proposed pretext tasks are complementary to masked autoencoding (MAE) and contrastive learning (CL), which are popular self-supervised learning strategies in computer vision and natural language processing. Masked autoencoding (He et al., 2022; Devlin et al., 2018) randomly removes part of the data (*e.g.*, pixels, words, or time series values) and trains the model to fill in the masked part through a high-level understanding of the neighboring context. Contrastive learning aims to induce a representation space in which different views of the same sequence are close together and different sequences are far apart. To study these strategies in the context of event sequence data, we adopt MAE and CL related techniques defined in recent work on sequential data self-supervised learning Chowdhury et al. (2023); Huang et al. (2024). We pre-train our backbone model using all three tasks together and evaluate on the temporal point process task. Implementation details can be found in Appendix A.5.

Figure 4 shows the study results. Across all four datasets, the top performing model on distribution modelling (NLL) is the combined model. The result suggests that alignment verification is complementary to MAE and CL, and can teach the model to understand event sequence data in a manifold different from previous pretext algorithms. The combined model also improves in accuracy while results are mixed for RMSE.

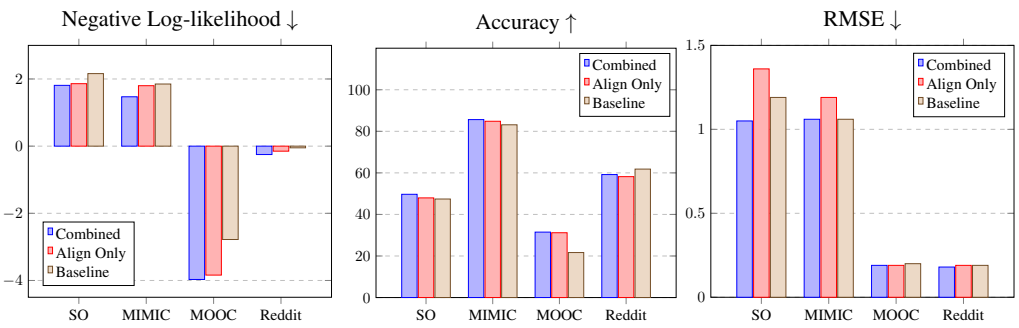


Figure 4: Experimental study combining the proposed alignment verification pretext task with masked reconstruction and contrastive learning on the temporal point process task. "Combined" refers to using masked reconstruction, contrastive learning, and alignment verification together, "Align Only" to alignment verification only and "Baseline" to the same backbone model without pretext training. Our proposed alignment pretext task can be further improved when pre-trained in a combined fashion.

Table 3: Few-shot performance by varying the portion of pretext training or fine-tuning data. We use a subset containing 25%, 50%, or 75% of the training or pretext training dataset and report fine-tuned TPP performance.

Methods	pre-train data %	fine-tune data %	MOOC			Reddit		
			NLL ↓	RMSE ↓	Acc ↑	NLL ↓	RMSE ↓	Acc ↑
Baseline	-	100%	$-2.78_{0.02}$	$0.20_{0.01}$	$21.66_{0.01}$	$0.05_{0.03}$	$0.19_{0.00}$	$61.82_{0.00}$
OURS	100%	25%	$-2.93_{0.38}$	$0.25_{0.04}$	$23.68_{0.02}$	$1.10_{0.26}$	$0.19_{0.01}$	$53.25_{0.03}$
	100%	50%	$-3.48_{0.16}$	$0.22_{0.02}$	$27.18_{0.01}$	$0.14_{0.17}$	$0.21_{0.01}$	$57.79_{0.01}$
	100%	75%	$-3.65_{0.12}$	$0.21_{0.02}$	$28.21_{0.01}$	$-0.07_{0.10}$	$0.19_{0.00}$	$58.75_{0.00}$
	25%	100%	$-3.42_{0.15}$	$0.19_{0.01}$	$25.48_{0.01}$	$-0.13_{0.02}$	$0.19_{0.01}$	$58.60_{0.00}$
	50%	100%	$-3.70_{0.15}$	$0.19_{0.00}$	$28.95_{0.02}$	$-0.15_{0.03}$	$0.19_{0.00}$	$58.12_{0.00}$
	75%	100%	$-3.83_{0.06}$	$0.19_{0.01}$	$29.79_{0.01}$	$-0.15_{0.02}$	$0.20_{0.00}$	$57.99_{0.01}$
	100%	100%	$-3.84_{0.07}$	$0.19_{0.00}$	$31.19_{0.02}$	$-0.15_{0.03}$	$0.19_{0.00}$	$58.19_{0.00}$

Few-shot settings. It is often necessary to learn generalizable representations when limited training data are available. Here, we examine the ability of our pretext training method to generalize well in few-shot settings by employing a subset of the dataset during the pretext training and fine-tuning stages respectively. Specifically, we utilize 25%, 50%, and 75% of the entire training dataset and subsequently evaluate the model's performance on the complete test dataset.

The experiment results are presented in Table 3. We find that fine-tuning our self-supervised representations on a fraction of the training dataset can, in some cases, already outperform the baseline network: our method using the full training data for pre-training and 25% of the training data for fine-tuning outperforms the baseline trained on the full training data across all metrics on MOOC. The result shows the few-shot learning potential of our method, making it valuable in scenarios with limited training resources. When we vary the percentage of training data available for pre-training, we find that the effectiveness of the self-supervised representations improves as more data is made available for pre-training.

A.5 Masked autoencoding and contrastive learning details

In Figure 4, we demonstrate how the proposed pretext tasks are complementary to masked autoencoding and contrastive learning, which are popular self-supervised learning strategies in computer vision and natural language processing. Masked autoencoding (He et al., 2022; Devlin et al., 2018) randomly removes part of the data (*e.g.*, pixels, words, or time series values) and train the model to fill in the masked part through a high-level understanding of the neighboring context. Contrastive learning aims to induce a representation space in which different views of the same sequence are close together and different sequences are far apart. The implementation details of masked autoencoding and contrastive learning for event sequence data are presented below.

Masked autoencoding details. We adopt a density-preserving masking strategy (Chowdhury et al., 2023) motivated by the non-uniform nature of event arrival times. This approach randomly samples intervals with constant time duration and masks all the events in the intervals. The underlying hypothesis is that time intervals with dense events contain more contextual information than sparse intervals, making the reconstruction of events easier in dense intervals. Therefore, an ideal masking strategy would preserve the original density of event arrival times, *i.e.*, masking more events when event arrivals become more frequent. Following common practice (Devlin et al., 2018; He et al., 2022), the masked events are replaced by a learnable [MASK] token. We feed the sequence consisting of (i) visible (non-masked) event embeddings and (ii) mask tokens to our encoder, and extract embeddings corresponding to the masked out locations. The embeddings are decoded to reconstruct the time and type embeddings of the masked events and we use mean squared error (MSE) to supervise the training of this pretext task. Denote the set of masked event indices by M. The masked reconstruction loss is

$$\mathcal{L}_{rec} = \frac{1}{|M|} \sum_{i \in M} \|\mathbf{e}_i^t - \tilde{\mathbf{e}}_i^t\|_2^2 + \|\mathbf{e}_i^m - \tilde{\mathbf{e}}_i^m\|_2^2, \tag{1}$$

where $\tilde{\mathbf{e}}_i^t$ and $\tilde{\mathbf{e}}_i^m$ denote the reconstructed event time and type embeddings of (t_i, m_i) respectively and $|M|$ is the number of masked events.

Contrastive learning details. For contrastive learning, we consider three augmentation techniques to construct multiple views: subsequence sampling, random masking, and adding noise.

View 1: Subsequence. We randomly extract subsequences from the original data and treat them as novel views, which is a strategy that has been found to be effective in time series (Yue et al., 2022; Chowdhury et al., 2023). This method is conceptually intuitive: Subsequences of event data can be seen as the counterparts of patches or croppings in vision research, representing a local division of original data. We extract subsequences by taking events in a continuous time period in the original sequences.

View 2: Masked events. We mask events to create novel views. We assume that masked data should contain similar semantic information as the original data, and contrastive learning based on masked data can further enhance the separation between similar and non-similar sequences.

View 3: Noisy data. We add multi-scale Gaussian noise to the event sequence embeddings and take them as novel views. Compared to *View 1* and *2*, noise injection is a type of augmentation closer to real-world data corruption. We take the following steps to inject noise into data: (i) Uniformly sample the scale of noise, $\sigma \in [0, 1]$, and (ii) sample Gaussian noise from $\mathcal{N}(0, \sigma)$. The sampled noise is added to the embeddings.

We use all three augmentation methods for each data in the mini-batch and adapt the normalized temperature-scaled cross-entropy loss (NT-Xent) (Chen et al., 2020) to fit our setting. Given a mini batch of training data of size B, we use S_i to denote the set consisting of the embedding of the ith sample and the embeddings of all its three augmented views, and $S = \bigcup_{i=1}^{N} S_i$ to denote the union of all S_is. Denote $P_{S_i}^2$ as the set of all combinations of two different embeddings from S_i with permutation (positive pairs), which is $\{(z, z') | z, z' \in S_i \text{ and } z \neq z'\}$. Given $k(= 3)$ augmented views, the total number of positive pairs in $P_{S_i}^2$ is $k \cdot (k + 1)$. We define the contrastive loss for the ith sample l_i as

$$l_i = -\frac{1}{|P_{S_i}^2|} \sum_{(z,z') \in P_{S_i}^2} \log \frac{\exp(z \cdot z'/\eta)}{\sum_{z'' \in S - \{z\}} \exp(z \cdot z''/\eta)}. \tag{2}$$

The overall contrastive loss $\mathcal{L}_{cl}$ is calculated as $\mathcal{L}_{cl} = \frac{1}{B} \sum_{i=1}^{B} l_i$.

When we combine the standard MAE and CL methods with our proposed alignment verification pretext task, the overall loss objective is

$$\mathcal{L} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{cl} + \lambda_3 \mathcal{L}_{align}, \tag{3}$$

where $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in our experiments.

A.6 Fine-grained analysis of masked autoencoding and contrastive learning.

To evaluate the effectiveness of the individual pretext tasks, i.e., masked reconstruction, contrastive learning and alignment verification, we conduct systematic ablations as shown in Table 4. The

Table 4: Systematic study of the three pretext tasks on temporal point process prediction using {Stack Overflow, MOOC and Reddit} datasets. "Rec", "Cont", and "Align" refer to masked reconstruction, contrastive learning, and alignment verification, respectively.

Methods	Rec	Cont	Align	NLL ↓	RMSE ↓	Acc ↑
Baseline				2.16	1.19	47.42
OURS	✓			1.86	1.13	49.68
		✓		1.84	1.16	49.80
			✓	1.86	1.36	47.96
	✓	✓		1.83	1.13	49.28
	✓		✓	1.84	1.23	**49.81**
		✓	✓	1.82	1.07	49.75
	✓	✓	✓	**1.81**	**1.05**	49.69

Methods	Rec	Cont	Align	MOOC			Reddit		
				NLL ↓	RMSE ↓	Acc ↑	NLL ↓	RMSE ↓	Acc ↑
Baseline	-	-	-	−2.78	0.20	21.66	0.05	0.19	61.82
OURS	✓			−3.75	0.19	29.96	−0.11	0.18	58.25
		✓		−3.54	0.18	29.47	−0.21	0.19	59.65
			✓	−3.84	0.19	31.19	−0.15	0.19	58.20
	✓	✓		−3.85	0.18	30.51	−0.25	0.19	59.91
		✓	✓	−3.94	0.20	30.22	−0.26	0.19	59.37
	✓		✓	−3.83	0.20	29.32	−0.23	0.19	58.82
	✓	✓	✓	−3.97	0.19	31.49	−0.25	0.18	59.19

baseline, which does not employ any pretext training, serves as a reference point for comparison, achieving an NLL of 2.16, RMSE of 1.19 and accuracy of 47.42%. When applying each pretext task individually, consistent improvements are observed on the main NLL metric. Applying pairs of pretext tasks results in better NLL than applying the pretext tasks individually (e.g., using both contrastive and alignment verficiation produces a lower NLL than applying only contrastive learning or only alignment verification). The best pairing combines contrastive learning and alignment verification. Finally, the overall best results are achieved when all three pretext tasks are employed together. The results of this ablation study indicates taht the three proposed pretext tasks are complementary, and that an integrated strategy combining all three pretext tasks provides the most effective pre-training.