PyTorch-Lifestream: Learning Embeddings on Discrete Event Sequences

Artem Sakhno[1] , **Ivan Kireev**[1] , **Dmitrii Babaev**[2] , **Maxim Savchenko**[1] , **Gleb Gusev**[1] and **Andrey Savchenko**[1,3]
[1]Sber AI Lab, Moscow, Russia
[2]SaluteDevices, Moscow, Russia
[3]ISP RAS Research Center for Trusted Artificial Intelligence, Moscow, Russia
sakhno.ad18@physics.msu.ru

Abstract

The domain of event sequences is widely applied in various industrial tasks in banking, healthcare, etc., where temporal tabular data processing is required. This paper introduces PyTorch-Lifestream, the first open-source library specially designed to handle event sequences. It supports scenarios with multimodal data and offers a variety of techniques for learning embeddings of event sequences and end-to-end model training. Furthermore, PyTorch-Lifestream efficiently implements state-of-the-art methods for event sequence analysis and adapts approaches from similar domains, thus enhancing the versatility and performance of sequence-based models for a wide range of applications, including financial risk scoring, campaigning, user ID matching, churn prediction, fraud detection, medical diagnostics, and recommender systems.

1 Introduction

In a broad range of real-world applications, including finance, healthcare, risk assessment, e-commerce, and telecommunications, information is processed in the form of tabular time series [Padhi *et al.*, 2021], i.e., event sequences [Babaev *et al.*, 2022], where each event is characterized by its timestamp and attributes. Such events include financial transactions (with attributes like MCC code, amount, etc.), medical visits (attributes: diagnosis, ICD code, analysis results), customer interactions, and online actions like clicks or purchases.

There exist many frameworks for working with such data, such as recommender systems [Zhao *et al.*, 2022; Vasilev *et al.*, 2024] and temporal point processes [Bacry *et al.*, 2017; Karpukhin *et al.*, 2024]. Both are designed to predict the next event type, user action, or item interaction. However, many applications of sequential data go beyond next-event prediction. Some tasks focus on individual events, such as fraud detection in transactions. In contrast, others require modeling entire event sequences, such as classifying full event streams, refining customer segmentation, or scoring user behavior. Frameworks designed exclusively for next-event prediction are often insufficient for these broader challenges, limiting their applicability in domains that require a more comprehensive approach to event sequence modeling.

Frameworks for other similar domains, multivariate time series [Tavenard *et al.*, 2020; Alexandrov *et al.*, 2020] or text processing [Wolf, 2019], though provide exceptional performance in sequence modeling, cannot be directly applied to event sequence processing. Indeed, they rely on regularly spaced data, where observations or text tokens occur at fixed intervals. In contrast, event sequences are inherently irregular, with varying time between events. Moreover, events in sequence may contain a mix of numerical and categorical features, making their processing more complex.

Large organizations typically have vast amounts of event data, enabling unsupervised/self-supervised learning (SSL) techniques to learn meaningful embeddings. Since only a small fraction is labeled, the typical pipeline involves training embeddings of event sequences, which are then utilized for downstream classification and regression tasks [Bazarova *et al.*, 2025]. However, frameworks specifically designed for this purpose, such as SimCLR [Chen *et al.*, 2020], often focus on narrow methodological aspects and do not provide a general framework for learning event representations.

As a result, practitioners working with event sequences often rely on general-purpose libraries like pandas or PySpark for feature engineering and preprocessing, followed by training models using deep learning frameworks such as PyTorch or TensorFlow. However, this approach requires significant adaptation, as these tools offer highly general-purpose functionality, whereas handling event sequences necessitates specialized solutions. This increases development time and model complexity and results in inefficiencies during both model creation and inference processes.

To bridge this gap, we introduce PyTorch-Lifestream (PTLS)[1], an open-source library[2] explicitly designed for learning embeddings of event sequences. Unlike existing solutions that primarily focus on explicit predictive tasks, our library is optimized for learning informative sequence embeddings that can be used across various downstream applications, such as customer segmentation, anomaly detection, and forecasting. The library supports multimodal event sequences, including transactions, clickstreams, and geolocation data [Mollaev *et al.*, 2024], offering a unified and effi-

[1]Lifestream is a concept for processing sequential events that represent a subject's "stream of life"

[2]https://github.com/pytorch-lifestream/pytorch-lifestream

cient approach to representation learning.

A key advantage of PyTorch-Lifestream is its flexible and modular design, allowing users to adapt the framework to diverse tasks and model architectures swiftly. The library offers a configurable pipeline that supports various sequence encoders, training objectives, and inference strategies. A video demonstrating its usage is available at^3.

2 PyTorch-Lifestream Overview

PyTorch-Lifestream offers a complete model development pipeline, enabling them to derive embeddings or scores for downstream tasks directly from raw data. Built on top of PyTorch Lightning, it is designed to be modular and easily extendable. This foundation simplifies distributed training, experiment reproducibility, and integration of custom components, such as additional checkpoints, evaluation metrics, and training logic. The library implements three core stages in the modeling process: preprocessing, training using SSL followed by fine-tuning, and inference (Figure 1).

PyTorch-Lifestream is a highly flexible framework that has been successfully deployed in real-world industrial scenarios involving millions of users. Its modular architecture and scalable data pipeline support make it well-suited for large-scale applications such as financial scoring, churn prediction, and user modeling. Details about our framework are provided in the following Subsections.

2.1 Preprocessing and Data Loading

In most business systems, data is stored in a "event-per-row" format, with several categorical and numerical features accompanying each action. However, this format makes it challenging to leverage such data in machine learning, as it lacks an explicit sequence structure.

In PyTorch-Lifestream, this shortcoming is addressed by various preprocessors that transform raw data into sequence-friendly formats and provide functions for handling categorical, numerical, and temporal features. These preprocessors support the fit-transform interface, similar to scikit-learn[Pedregosa *et al.*, 2011], ensuring seamless integration into machine learning pipelines. For instance, PTLS includes a FrequencyEncoder component to efficiently convert the original categories into a training-ready format through frequency-based encoding. The library provides three preprocessors, each employing different toolsets for working with tabular data: pandas, Dask, and PySpark. After all transformations, a table is generated in the "sequence-per-row" format.

To improve computational efficiency, PTLS enables structured storage of preprocessed data in the Parquet format, which provides efficient compression and rapid retrieval into memory [Apache Arrow, 2016].

The framework offers an extensible Dataset interface that supports map-style and iterable-style datasets, facilitating flexible data handling strategies. This dual-mode support allows seamless adaptation to large-scale datasets that exceed memory constraints. Various options are available to transform raw data during loading. Users can limit the number of events, select a random data slice, shuffle the sequence of events, etc.

^{3}https://youtu.be/C1opqDCIfhI

2.2 Model Architecture

Our framework employs two key architectural components: the event encoder (TrxEncoder) and the sequence encoder (SeqEncoder). TrxEncoder consists of an embedding layer for categorical features and normalization for numerical features. The embeddings of categorical and numerical features are concatenated and passed through a linear layer. Custom numerical feature processing can be implemented. For example, to apply a custom piecewise linear encoding for numerical features [Gorishniy *et al.*, 2022], it is sufficient to pass it as a scaler argument to TrxEncoder, allowing seamless integration of advanced feature engineering techniques.

Two main architectures are implemented for SeqEncoder: RNN (Recurrent Neural Network) and Transformer. The library supports modular replacement of standard sequence encoders with more advanced architectures. For instance, the SeqEncoder can use state-of-the-art transformers from Hugging Face [Wolf, 2019] or the x-transformers [Wang, 2020] library. This ensures seamless integration with popular Transformer-based architectures.

As some tasks require an embedding for each token in the input sequence, while others require a single embedding for the entire sequence, a parameter (is_reduce_sequence) controls this behavior. When set to true, the model outputs a single embedding suitable for predicting a single label. Otherwise, the model produces an embedding for each input token, enabling sequence-level processing.

PTLS also provides functionality for working with multimodal data, including a built-in method for early fusion [Zong *et al.*, 2024], which enables combining different modalities by using separate TrxEncoders for each modality and a shared SeqEncoder. Multimodality in sequential data allows the model to integrate information from clickstream, transactions, and other contextual interactions, leading to a more comprehensive understanding of user activity.

2.3 Implemented Methods

PyTorch-Lifestream implements various state-of-the-art pretraining methods for event sequences, for example:

- CoLES [Babaev *et al.*, 2022] and its modifications: the library provides a wide range of samplers and loss functions for contrastive learning, including conventional contrastive loss [Bromley *et al.*, 1993], triplet loss [Hoffer and Ailon, 2015], and InfoNCE [Oord *et al.*, 2018]. Additionally, regularization techniques such as VicReg [Bardes *et al.*, 2021] and Barlow Twins [Zbontar *et al.*, 2021] are supported, allowing for training embeddings without negative examples.
- GPT-2 [Radford *et al.*, 2019]: prediction of all categorical features of the next event.
- MLM (Masked Language Modeling) [Devlin *et al.*, 2019; Liu *et al.*, 2019]: the event embedding is masked at the event encoder output, as implemented in [Baevski *et al.*, 2020; Yugay and Zaytsev, 2024]. The neural network is trained to reconstruct the masked embedding us-

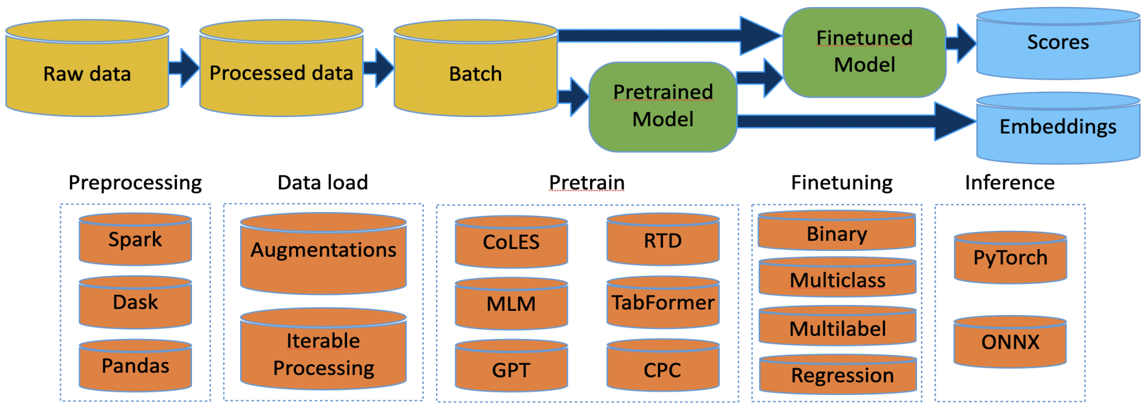

Figure 1: Architecture of the proposed PyTorch-Lifestream framework

ing a contrastive loss function. Additionally, the loss of NSP (next sentence prediction) can be used.

- RTD (Replace Token Detection): the library employs a simplified version of the architecture proposed in [Clark, 2020]. TrxEncoder is used as the encoder in our implementation, and the replaced token is randomly selected from the batch.
- TabFormer [Padhi *et al.*, 2021]: a portion of event features is masked and subsequently reconstructed during model training.
- CPC (Contrastive Predictive Coding) [Oord *et al.*, 2018]: during training, the next n embeddings produced by the event encoder are predicted.

Moreover, we support flexible loss functions. Users are not restricted to built-in loss functions. They can leverage specialized metric learning libraries such as PyTorch Metric Learning [Musgrave *et al.*, 2020] or SSL libraries to incorporate modern loss functions.

A dedicated class, SequenceToTarget, fine-tunes the model, enabling further training for classification, regression, and other tasks. This class supports both training models from scratch and fine-tuning pre-trained models, allowing users to leverage prior knowledge encoded in embeddings while adapting to specific downstream tasks. This class allows training for supervised tasks for a given target at the sequence and individual transaction levels.

3 Our Benchmark

Together with our library, we introduce a benchmark for event sequence modeling. In this paper, we present its part for three datasets: Default Scoring [Boosters, 2025a], Churn prediction [Boosters, 2025b], and Age prediction [ODS.AI, 2025]. These tasks are widely applied in the banking sector and other business areas. Each dataset represents user transactional activity data. However, the datasets differ in scale: the Scoring dataset contains more than 1,400,000 users, the Age dataset has 50,000 users, and the Churn dataset includes less than 11,000 users.

Table 1 presents the results obtained using the methods implemented in PyTorch-Lifestream. Since some of these methods, such as GPT, MLM, and Tabformer, generate multiple embeddings, pooling across all transactions was applied for their use in downstream tasks. The library also includes a model that implements manually engineered features by aggregating various statistics along the sequence. The results show that the implemented methods frequently outperform approaches relying on manually engineered features. Moreover, blending or concatenating embeddings from different models often improves quality.

	Scoring	**Age**	**Churn**
Method	AUROC	Accuracy	AUROC
Hand made features	0.7792	0.629	0.827
Barlow Twins	0.7878	0.643	0.839
VicReg	0.7886	0.598	0.829
CPC	0.7919	0.602	0.792
CoLES	0.7921	0.640	0.841
MLM	0.7791	0.621	0.817
RTD	0.7910	0.631	0.771
Tabformer	0.7862	0.601	0.827
GPT	0.7737	0.589	0.824

Table 1: Comparison results for methods from PyTorch-Lifestream

4 Conclusion

This paper introduced PyTorch-Lifestream, the first publicly available unified framework for representing and modeling discrete event sequences. The integration of robust preprocessing and diverse model architectures, which can be adapted to various tasks and models thanks to the modular structure and code flexibility, allows researchers and practitioners to integrate the framework into their work quickly.

Beyond its strong performance, PyTorch-Lifestream offers practical advantages, including flexible data-loading options, efficient inference, and a modular codebase that facilitates customization. By providing a user-friendly interface for unimodal and multimodal sequence processing, our library lowers the barrier to adopting advanced sequence-based techniques while remaining extensible for evolving research and industry needs.

Acknowledgments

The work of A. Savchenko on Section 2 was supported by the Ministry of Economic Development of the RF (code 25-139-66879-1-0003).

References

[Alexandrov *et al.*, 2020] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[Apache Arrow, 2016] Apache Software Foundation Apache Arrow. Apache arrow, 2016.

[Babaev *et al.*, 2022] Dmitrii Babaev, Nikita Ovsov, Ivan Kireev, Maria Ivanova, Gleb Gusev, Ivan Nazarov, and Alexander Tuzhilin. CoLES: Contrastive learning for event sequences with self-supervision. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD/PODS '22. ACM, June 2022.

[Bacry *et al.*, 2017] E. Bacry, M. Bompaire, S. Gaïffas, and S. Poulsen. tick: a Python library for statistical learning, with a particular emphasis on time-dependent modeling. *ArXiv e-prints*, July 2017.

[Baevski *et al.*, 2020] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: a framework for self-supervised learning of speech representations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[Bardes *et al.*, 2021] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[Bazarova *et al.*, 2025] Alexandra Bazarova, Maria Kovaleva, Ilya Kuleshov, Evgenia Romanenkova, Alexander Stepikin, Aleksandr Yugay, Dzhambulat Mollaev, Ivan Kireev, Andrey Savchenko, and Alexey Zaytsev. Learning transactions representations for information management in banks: Mastering local, global, and external knowledge. *International Journal of Information Management Data Insights*, 5(1):100323, 2025.

[Boosters, 2025a] Boosters. Alfa Battle 2 Overview, 2025. [Online; accessed 11-February-2025].

[Boosters, 2025b] Boosters. Rosbank Challenge 1 Overview, 2025. [Online; accessed 11-February-2025].

[Bromley *et al.*, 1993] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020.

[Clark, 2020] K Clark. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Gorishniy *et al.*, 2022] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004, 2022.

[Hoffer and Ailon, 2015] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *Similarity-based pattern recognition: third international workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pages 84–92. Springer, 2015.

[Karpukhin *et al.*, 2024] Ivan Karpukhin, Foma Shipilov, and Andrey Savchenko. Hotpp benchmark: Are we good at the long horizon events forecasting?, 2024.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[Mollaev *et al.*, 2024] Dzhambulat Mollaev, Alexander Kostin, Maria Postnova, Ivan Karpukhin, Ivan A Kireev, Gleb Gusev, and Andrey Savchenko. Multimodal banking dataset: Understanding client needs through event sequences, 2024.

[Musgrave *et al.*, 2020] Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim. Pytorch metric learning. *ArXiv*, abs/2008.09164, 2020.

[ODS.AI, 2025] ODS.AI. Sberbank Sirius Lesson Competition, 2025. [Online; accessed 11-February-2025].

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[Padhi *et al.*, 2021] Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3565–3569. IEEE, 2021.

[Pedregosa *et al.*, 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron

Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Tavenard *et al.*, 2020] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[Vasilev *et al.*, 2024] Alexey Vasilev, Anna Volodkevich, Denis Kulandin, Tatiana Bysheva, and Anton Klenitskiy. Replay: a recommendation framework for experimentation and production use. In *18th ACM Conference on Recommender Systems*, RecSys '24, page 1191–1194. ACM, October 2024.

[Wang, 2020] Phil Wang. x-transformers: A flexible transformer implementation in pytorch, 2020. GitHub repository.

[Wolf, 2019] T Wolf. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[Yugay and Zaytsev, 2024] Aleksandr Yugay and Alexey Zaytsev. Uniting contrastive and generative learning for event sequences models. *ArXiv*, abs/2408.09995, 2024.

[Zbontar *et al.*, 2021] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

[Zhao *et al.*, 2022] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. Recbole 2.0: Towards a more up-to-date recommendation library. In *CIKM*, pages 4722–4726. ACM, 2022.

[Zong *et al.*, 2024] Yongshuo Zong, Oisin Mac Aodha, and Timothy Hospedales. Self-supervised multimodal learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.