Integrating Sequential and Relational Modeling for User Events: Datasets and Prediction Tasks

Rizal Fathony, Igor Melnyk, Owen Reinert, Nam H. Nguyen, Daniele Rosa, C. Bayan Bruss
Capital One
{rizal.fathony,igor.melnyk,owen.reinert,nam.nguyen,
daniele.rosa,bayan.bruss}@capitalone.com

Abstract

User event modeling plays a central role in many machine learning applications, with use cases spanning e-commerce, social media, finance, cybersecurity, and other domains. User events can be broadly categorized into personal events, which involve individual actions, and relational events, which involve interactions between two users. These two types of events are typically modeled separately, using sequence-based methods for personal events and graph-based methods for relational events. Despite the need to capture both event types in real-world systems, prior work has rarely considered them together. This is often due to the convenient simplification that user behavior can be adequately represented by a single formalization, either as a sequence or a graph. To address this gap, there is a need for public datasets and prediction tasks that explicitly incorporate both personal and relational events. In this work, we introduce a collection of such datasets, propose a unified formalization, and empirically show that models benefit from incorporating both event types. Our results also indicate that current methods leave notable room for improvements. We release these resources[1] to support further research in unified user event modeling and encourage progress in this direction.

1 Introduction

Modeling user events is a central task in machine learning with broad applications across various domains [1–3]. In e-commerce, it is used to capture user preferences for personalized ranking and product recommendation [4, 5]. In social media platforms, event modeling supports feed optimization and engagement prediction by inferring user interests over time [6–8]. Financial systems leverage user behavior data for fraud detection, credit risk assessment, and behavioral profiling [9–12]. Online services such as search and streaming platforms rely on user event sequences for content recommendation under real-time constraints [13–16]. In cybersecurity, modeling user and system events is essential for detecting anomalies and preventing intrusions [17, 18]. These applications demonstrate the importance of building models that can effectively capture complex, context-dependent user behavior from event sequences.

User events can be broadly categorized into personal and relational events. Personal events involve only a single user and reflect individual actions, such as searching for content, viewing items, or posting updates. In contrast, relational events involve interactions between two or more users, such as following another user, co-editing a document, or exchanging messages. Traditionally, these two types of events are often modeled separately. Relational events are commonly modeled using graph-based approaches that capture structural dependencies and interaction patterns among users [19–22]. On the other hand, personal events are typically modeled as sequences using recurrent or attention-based architectures to capture temporal dependencies in personal event histories [23–29].

[1] Link to the datasets and prediction tasks. Link to the dataset construction and experimentation code.

Fathony et al., Integrating Sequential and Relational Modeling for User Events: Datasets and Prediction Tasks. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 338, Arizona State University, Phoenix, USA, December 10–12, 2025.

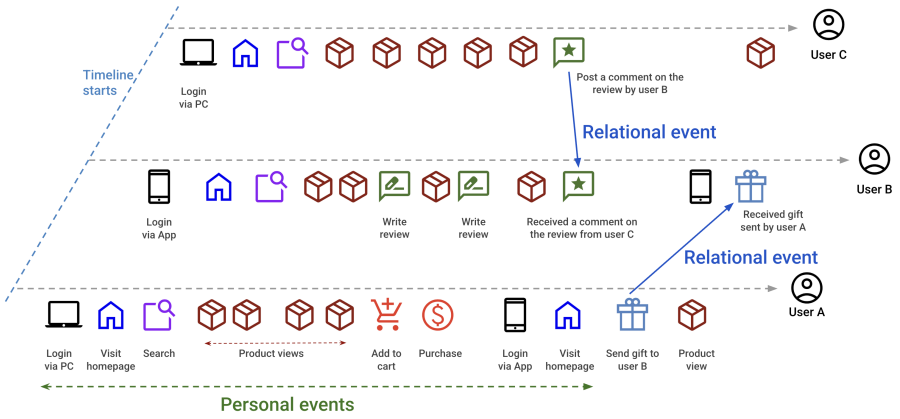


Figure 1: An illustration of personal and relational events in e-commerce. Personal events involve a single user, such as login, search, view, or purchase. Relational events involve interaction between two users, such as sending a gift or commenting on another user's review.

There have been efforts in the graph area to capture both structural and temporal dependencies using temporal graph formalizations (such as CTDG [30–32]) and models built on top of these formalizations (such as TGAT [33], TGN [31], DyRep [34], TNCN [35], and others [36–38]). However, these approaches primarily focus on the temporal dependencies of relational events while neglecting personal events. For example, the formalization used in the Temporal Graph Benchmark (TGB) papers [39, 40] and recent temporal graph models [35–38] defines a temporal graph as a stream of triplets consisting of source, destination, and timestamp. Personal events that involve only a single entity cannot be directly represented under this formulation. One workaround is to convert all personal events into nodes and define personal events as triplets of user node, event node, and timestamp. However, this construction is not as straightforward for capturing temporal dependencies in personal event histories compared to sequence-based modeling.

Going back to the personal and relational event category, in many application domains, the number of personal events is typically much larger than that of relational events. For example, in e-commerce platforms, as illustrated in Figure 1, users often view products, search for items, or add products to their cart, whereas relational interactions, such as referrals, sending gifts, or socially engaged reviews, are less frequent. In financial systems, customers routinely perform account queries, check balances, or initiate transactions, while peer-to-peer interactions such as money transfers or joint account actions are relatively infrequent. In cybersecurity systems, personal events may include actions like logging in, accessing files, or executing processes, while relational events, such as remote connections to other users, or file sharing between users, occur less frequently. Despite their higher volume, personal events are often underrepresented in existing graph-based formulations, which tend to prioritize relational structure. In practice, however, both personal and relational events carry complementary signals, and many predictive tasks, such as item recommendation, fraud detection, customer profiling, and behavior forecasting, benefit from capturing both types of information.

Even though there is a need to capture both personal and relational events in many application domains, prior work has rarely considered them together. Practitioners often simplify the complexity of user event modeling by adopting either a graph or a sequence formalization, as most machine learning models are developed within one of these frameworks. As a result, one type of event, typically the less convenient to represent, is often ignored entirely, leading to an incomplete view of user behavior. To build a more comprehensive understanding of user event modeling, there is a need for public datasets and benchmark tasks that explicitly incorporate both event types. Such resources would provide a foundation for developing and evaluating models that integrate these complementary signals.

Summary of Contributions. In this work, we aim to support the study of user event modeling that incorporates both personal and relational events. Our contributions are as follows:

- We curate, pre-process, and release a collection of public datasets and prediction tasks that explicitly include both personal and relational events.

- We introduce a new formalization for user event modeling that captures both personal and relational events.
- We empirically demonstrate that incorporating both personal and relational events improves performance on a range of prediction tasks.
- We show that existing models, originally developed for either sequential or relational data only, are less well suited for this event modeling setting, leaving room for future improvements.
- We invite the community to use these resources and close the gap in unified user event modeling.

2 Related Works

We discuss the most relevant works in this section, while a more comprehensive list of the related works can be seen in Appendix D.

Event sequence modeling, such as temporal point processes [41–43], deals with event stream data $(e_1, t_1), (e_2, t_2), \cdots$, where each e_i is an event type drawn from an event set $\mathcal{E} = \{1, 2, \cdots, |\mathcal{E}|\}$, and $0 \leq t_1 \leq t_2 \leq \cdots$ are the times of occurrence. In **user event sequence** modeling, such as personalized event prediction tasks [44–46], we have N users, denoted as $\mathcal{U} = \{u_1, u_2, \cdots, u_N\}$. Each user has their own sequence of events, denoted $\text{Seq}(u) = [(e_1, t_1)^{(u)}, (e_2, t_2)^{(u)}, \cdots]$. In some applications, exact timestamps are replaced with discrete time steps, simplifying the definition to $\text{Seq}(u) = [e_1^{(u)}, e_2^{(u)}, \cdots]$. In other variants of sequence modeling, the event space may be restricted to a single domain, for example, products consumed by a user in sequential recommendation tasks [26, 47, 48]. While these user event sequence models capture the complexity of event sequences within a user, they lack the ability to encode user-to-user interactions.

Graph modeling, on the other hand, explicitly encodes user-to-user interactions through its node and edge abstraction, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the node and edge sets, respectively. Temporal graph abstractions, such as CTDG [30, 31], incorporate temporal dynamics by representing a graph as a sequence of time-stamped events $\mathcal{G} = \{x(t_1), x(t_2), \cdots\}$. Each event $x(t)$ can be either (1) a *node-wise event* (e.g., node addition or feature update) represented by a feature vector $\mathbf{v}_i(t)$, where i is the node index, or (2) an *interaction event* between node i and node j represented by a temporal edge $\mathbf{e}_{i,j}(t)$. Another CTDG formulation by Kazemi et al. [32] describes a temporal graph as a pair consisting of an initial graph and an observation set $(\mathcal{G}, \mathcal{O})$. The observation set describes the evolution of the graph, with each observation is represented as a tuple (*event type, event, timestamp*), where the event type may include edge addition, edge deletion, node addition, node deletion, node splitting, node merging, etc. In practice, however, many temporal graph models [35–38] omit node-level events altogether and instead define a temporal graph as a sequence of interaction events expressed as triplets, $\mathcal{G} = \{(u_1, v_1, t_1), (u_2, v_2, t_2), \cdots\}$, where u and v represent the source and destination nodes.

3 Problem Formalization

Notations. For our **P**ersonal and **R**elational User **E**vent **S**equence (PRES) modeling, we take inspiration from the standard user event sequence modeling. We denote a set of users (which can also be a customer, account, entity, etc.) as $\mathcal{U} = \{u_1, u_2, \cdots, u_N\}$, where N is the number of users. Each user has their own sequence of events that occur over time. For example, the sequence for user u_i is denoted as $\text{Seq}(u_i) = [(e_1, t_1), (e_2, t_2), \cdots, (e_{M_i}, t_{M_i})]$, where e describes an event, t describes the time at which the event occurs, and M_i denotes the number of events for user u_i. Each user may have a different number of events in their event sequence. We denote the set of all user sequences by $\mathcal{S} = \{\text{Seq}(u) \mid u \in \mathcal{U}\}$. An event may come from two different event sets: the *personal event* set and the *relational event* set. The personal event set contains a set of events that can occur for an individual user; $p \in \mathcal{P} \triangleq \{1, 2, \cdots, |\mathcal{P}|\}$. The relational event set contains a set of all possible events $r \in \mathcal{R} \triangleq \{1, 2, \cdots, |\mathcal{R}|\}$, which involve a relation from one user to another. Thus, an event can be defined by a personal event $e = p$, or a relational event tuple $e = (r, v)$, where v is another user.

Difference from temporal graph formulation. Although motivated by a similar goal of integrating temporal and relational information, our formulation differs from temporal graphs in several ways:

- In the temporal graph formulation, events are ordered globally across all users, similar to the standard (non-user-based) event sequence modeling. In contrast, PRES follows personalized user-event sequence modeling, where each user has their own separate sequence of events.

Table 1: Dataset Statistics

Properties	BRIGHTKITE	GOWALLA	AMAZON-CLOTHING	AMAZON-ELECTRONICS	GITHUB
Personal Events	check-in	check-in	product rating	product rating	github activity
Relational Events	friendship	friendship	co-review	co-review	collaboration
# Users	58,228	196,591	185,986	254,064	3,669,079
# Events	5,130,866	8,342,943	1,591,947	2,938,178	102,878,895
# Personal Events	4,702,710	6,442,289	1,573,869	2,281,128	95,974,149
# Relational Events	428,156	1,900,654	18,078	657,050	6,904,746
# Unique Events	628,519	1,169,154	846,052	529,198	24
# Unique Timestamps	4,506,822	5,561,957	3,464	5,373	2,675,990
# Users w. pers. events	51,406	107,092	185,986	254,064	3,669,079
# Users w. rel. events	58,228	196,591	5,017	49,852	441,958
# Users w. both events	51,406	107,092	5,017	49,852	441,958

- The temporal graph formulation places greater emphasis on modeling interaction events between two nodes, often neglecting user events that do not involve another user. Many recent temporal graph models, such as TNCN [35], NAT [36], DyGFormer [37], and others [38, 49, 50]; recent temporal graph benchmarks, such as TGB [39] and TGB 2.0 [40]; as well as popular graph frameworks such as PyTorch Geometric [51], even reduce the formulation to a sequence of triplets representing interaction events with no support for encoding personal events. By contrast, PRES treats personal events as first-class components. This is particularly relevant in many areas mentioned in the introduction, where the number of personal events is far larger than that of relational events.
- Even for the temporal graph formulation that accommodate node-wise events, it does so by updating node feature vectors. This differs from event sequence modeling (including PRES), where each event is drawn from a discrete set of events, a property that may not be easily encoded as feature changes. To represent discrete personal events, temporal graph models may convert the events into nodes and then build a heterogeneous dynamic graph that encodes the event as an interaction between a user node and this newly created event node. In some events, such as 'viewing product', this approach works fine. However, it does not work well for representing personal events that describe intransitive actions (e.g., 'sign in', 'login', or 'subscribe') or status changes (e.g., 'payment successful', 'login unsuccessful', or 'request denied').
- In graph models, every entity must be a node. This becomes problematic when the entity has a hierarchical structure. For example, in a location-based social network application, when modeling a user's location check-in event, one may want to capture hierarchical information such as continent, country, state, city, neighborhood, and block. Representing the location check-in as a single node in a graph model loses this hierarchical structure. In contrast, PRES allows more flexibility: using a sequence model, events can be freely tokenized. A check-in event can be split into multiple tokens, each corresponding to a different level of location granularity.

When we compare with the CTDG formulation by Kazemi et al. [32], all the concerns above remain. While this formulation allows more flexible observations, it is still restricted to events that describe the evolution of the graph through changes in nodes or edges, such as edge additions, edge deletions, node additions, node deletions, node splitting, or node merging.

4 Datasets and Prediction Tasks

4.1 Dataset Information

We curated user event datasets from multiple domains[2], following recent graph dataset curation works [39, 40, 52–54], and then processed them according to our formalization in Section 3. The data is stored in CSV format with the columns: `uid`, `timestamp`, `event_set`, `event`, and `other_uid` (See Appendix B for details). The `uid` is a numerical user ID, whereas `event_set` indicates whether the event is *personal* or *relational*. For relational events, `other_uid` refers to the other user involved in the relation; for personal events, this column is null. The statistics of each datasets is shown

[2]The datasets presented in this paper are used solely to demonstrate the claims made in this work and are not reflective of any datasets leveraged by Capital One.

in Table 1. More details on collection and dataset license are available in Appendix A. The data processing code are open-sourced and descriptions are available in Appendix C.

PRES-BRIGHTKITE. This dataset contains location check-ins and friendship history of Brightkite users, a location-based social networking platform. It was originally collected by Cho et al. [55] and published in the SNAP Dataset Repository [56]. Personal events consist of sequences of location check-ins. We convert the original latitude and longitude coordinates into Geohash-8 representations [57, 58], short alphanumeric strings encoding geographic locations. Nearby locations share similar geohash prefixes, while distant ones differ. Example geohashes include 9v6kpmr1, gcpwkeq6, and u0yhxgm1. Relational events capture friendship connections among users. The dataset includes 58,228 users and 5,130,866 events. In this dataset, only the personal events include timestamp information; relational events do not have associated timestamps.

PRES-GOWALLA. The dataset also contains the location check-in and friendship history of another social network platform, Gowalla. It was also originally collected by Cho et al. [55] and published in the SNAP Repository [56]. We processed and formatted the data following the same approach used for PRES-BRIGHTKITE. The dataset contains personal events from geohash check-ins and relational events from friendship connections, totaling 8,342,943 events from 196,591 users.

PRES-AMAZON-CLOTHING. The dataset contains Amazon product reviews and ratings in the *Clothing, Shoes and Jewelry* category, spanning from May 1996 to July 2014. The raw data was originally collected by McAuley et al. [59]. In this dataset, we define personal events as sequences of product IDs and ratings reviewed by a user, for example: B000MLDCZ2:5 and B001OE3F08:3. Relational events represent co-review patterns, where two users have co-reviewed at least three products. The dataset contains event sequences from 185,986 users, with a total of 1,591,947 events.

PRES-AMAZON-ELECTRONICS. The dataset contains Amazon product reviews and ratings in the *Electronics* category, originally collected by McAuley et al. [59]. As in PRES-AMAZON-CLOTHING, personal events are defined as sequences of product IDs and ratings, while relational events capture co-review patterns. In total, the dataset contains 2,938,178 events from 254,064 users.

PRES-GITHUB. This dataset contains GitHub user activity from January 2025, collected from the GH Archive. Personal events include actions such as Push, CreateBranch, CreateRepository, PullRequestOpened, IssuesOpened, and Fork. Relational events represent project collaboration, where two users are linked if both contributed at least five commits or pull requests to the same repository. The dataset includes 102,878,895 events from 3,669,079 users. In this dataset, only personal events include timestamp information; relational events do not have associated timestamps, similar to PRES-BRIGHTKITE and PRES-GOWALLA.

Variability of the datasets. As shown in Table 1, the PRES datasets vary significantly across multiple aspects. The number of users ranges from around 58 thousand in PRES-BRIGHTKITE to more than 3.5 million in PRES-GITHUB. The number of events also varies, from approximately 1.5 million in PRES-BRIGHTKITE to over 100 million in PRES-GITHUB. The ratio between relational and personal events ranges from around 1:3 in PRES-GOWALLA to approximately 1:80 in PRES-AMAZON-CLOTHING. The number of unique events also differs widely, from just 24 in PRES-GITHUB to more than 1 million in PRES-GOWALLA. In addition, we observe variability in the number of users having personal events, relational events, and both. Some datasets have more users with relational events than with personal events (e.g., PRES-BRIGHTKITE, PRES-GOWALLA), while others show the opposite trend (e.g., PRES-AMAZON-CLOTHING, PRES-AMAZON-ELECTRONICS, PRES-GITHUB). These differences in dataset properties present distinct challenges for modeling user events in each dataset.

4.2 Prediction Tasks

From the PRES datasets, we define two prediction tasks: one for relational events and one for personal events. These tasks are designed to enable fair comparisons between graph-based, sequence-based, and hybrid models. Relational event prediction focuses on predicting future or held-out subset of user-to-user interactions, similar to link prediction. Personal event prediction aims to predict the likelihood of future occurrence of personal events without requiring exact timestamps, for example, predicting the next 20 personal events given a user's first 100. In both tasks, observed events are compared against negative samples drawn from events not associated with the user. For reproducibility, pre-generated negative samples for validation and test sets are provided in the dataset repository.

Relational event prediction tasks. The corresponding tasks for PRES-BRIGHTKITE and PRES-GOWALLA involve friend recommendation. We construct the training data by randomly splitting all relational events into 70% training, 10% validation, and 20% test sets. We also generate negative samples for the validation and test sets. Following Gastinger et al. [40], we adopt a *1-vs-1000* negative sampling scheme, in which 1,000 negative events are sampled for each relational event in the prediction set. Negative samples are drawn via uniform random sampling of users, excluding those who already have relational events with the target user in the training set.

For the PRES-GITHUB dataset, the relational event prediction task is defined as collaboration prediction, which involves predicting which users collaborate with a given user. The train, validation, and test splits follow the same procedure as in PRES-BRIGHTKITE, including the sampling method. However, due to the large size of the dataset, we adopt a *1-vs-300* negative sampling scheme.

For the PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS datasets, the task is predicting co-review relationships, i.e., which users share at least three products they reviewed. Co-review patterns can reveal how one account may be related to another, which in some cases can help detect fraudulent review syndicates. In these datasets, relational events have timestamp information, i.e., the first time the co-review condition is met. As such, the train, validation, and test splits respect event timestamps. Specifically, we split each user's relational events by taking the last 20% for test, the previous 10% for validation, and the rest for training. To manage large histories of some users, we cap test and val sets at 20 and 10 events per user, respectively. Personal events are also split into 'observed' and 'unobserved' sets based on the timestamp cut-off in the relational event split, with only the observed set used for training. As in PRES-BRIGHTKITE, we adopt a *1-vs-1000* negative sampling scheme.

Personal event prediction tasks. The task for PRES-BRIGHTKITE and PRES-GOWALLA is to predict the likelihood of a user checking in at a given geohash location in the future. We split each user's personal events by taking the last 20% for test, the previous 10% for validation, and the rest for training. We also cap the number of events in the test and validation sets to at most 20 and 10 per user, respectively. Since personal events are more frequent than relational ones, we adopt a *1-vs-500* negative sampling scheme. As geohash strings encode hierarchical spatial information (e.g., earlier characters represent broader regions), we apply stratified hierarchical sampling. Specifically, negatives are stratified by shared geohash prefixes, from matching the first five characters to none, ensuring a mix of nearby and distant locations.

For the PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS datasets, the task is to predict future products a user will review and the corresponding ratings, as denoted in their personal event data. We adopt the same train/val/test split strategy as in PRES-BRIGHTKITE, along with a *1-vs-500* negative sampling scheme. Negative samples for each personal event (e.g., B001OE3F08:3) are drawn from three sources: (1) the same product with different ratings (e.g., B001OE3F08:1, B001OE3F08:5); (2) other personal events not in the user's training data; and (3) samples from the second set with randomly perturbed ratings.

In the PRES-GITHUB dataset, the number of unique events in the personal event set is only 24, corresponding to the list of possible GitHub activities. Thus, the task construction used in the previous datasets is not applicable to PRES-GITHUB. We decided to omit this dataset from the set of datasets used for creating personal event prediction tasks.

Full event sequence. In addition to the datasets containing prediction tasks described above, we also publish a version of each dataset that includes all personal and relational events for all users, without any assigned tasks, train/val/test splits, or pre-specified negative samples. This is intended to facilitate future works that may wish to generate other prediction tasks not covered in this paper.

5 Experiments

5.1 Relational event prediction tasks

Experiment setup. We perform relational event prediction experiments on all five PRES datasets, following the task setup described earlier. We evaluate several sets of baseline methods:

1. In the first set, we use only relational event data. We construct a user graph where edges represent relational events between two users, ignoring timestamp information. We then run static graph methods, GCN [60], GAT [61], and Graph Transformer (TConv) [62] on this graph.

Table 2: Performance results for relational event prediction tasks (all metrics are in percent).

Method	PRES-BRIGHTKITE					PRES-GOWALLA				
Metric	MRR	Hits@5	Hits@10	Hits@50	Hits@100	MRR	Hits@5	Hits@10	Hits@50	Hits@100
Static graph models on relational event graph										
GCN	37.3±0.8	50.8±1.0	61.7±0.9	83.2±0.4	89.5±0.3	40.3±0.9	54.5±0.9	65.8±0.8	86.5±0.4	92.0±0.2
GAT	36.2±1.4	48.7±1.4	59.5±1.2	81.4±0.8	88.5±0.6	40.7±1.5	54.1±1.6	64.9±1.5	85.3±1.3	91.1±1.0
TConv	40.2±2.0	53.0±2.4	63.5±2.3	84.1±1.1	90.0±0.6	47.4±1.1	62.0±1.1	72.1±0.8	88.9±0.3	93.1±0.2
Static graph models on relational event graph + sequence embedding from personal event data										
GCN+S	43.9±0.7	57.8±0.8	67.8±0.8	86.5±0.3	91.5±0.1	44.9±1.0	59.4±1.1	69.8±1.0	88.1±0.5	92.8±0.3
GAT+S	44.8±1.1	58.5±1.1	68.2±1.1	86.2±0.5	91.5±0.4	44.9±0.9	58.8±0.6	69.0±0.4	87.0±0.4	92.0±0.5
TConv+S	**47.4±1.5**	**61.6±1.7**	**71.4±1.4**	**88.2±0.4**	**92.5±0.2**	**49.9±1.1**	**64.6±1.1**	**74.4±0.9**	**90.0±0.5**	**93.8±0.3**
Static graph models on relational event graph + personal event nodes										
GCN_RP	8.7±0.9	11.0±1.2	15.7±1.7	35.6±3.7	49.8±4.5	17.0±0.9	22.1±1.2	29.8±1.6	56.4±3.0	70.8±2.9
GAT_RP	10.7±1.0	13.5±1.2	18.2±1.4	35.6±2.3	47.8±2.8	14.9±1.4	19.0±1.6	25.8±2.0	50.7±3.1	66.2±3.2
TConv_RP	15.8±1.6	21.3±2.3	29.2±2.7	55.7±3.2	70.4±2.7	21.0±1.5	28.2±2.2	38.1±2.7	67.4±3.0	80.4±2.4
Temporal graph models on relational event graph + personal event nodes										
TGN	12.2±0.7	15.9±0.9	23.5±1.0	50.2±1.3	63.5±1.3	15.4±2.6	20.6±3.7	27.8±4.6	51.8±5.6	64.9±5.4
DyRep	7.1±0.4	8.9±0.6	13.7±0.9	36.0±1.7	50.7±2.1	8.8±1.0	11.2±1.3	15.8±1.7	34.8±3.5	48.6±5.1
TNCN	28.5±1.3	34.5±1.6	40.7±1.9	62.6±2.1	72.8±1.8	25.1±1.6	29.2±2.0	33.7±2.4	52.2±2.5	63.4±2.4

Method	PRES-AMAZON-CLOTHING					PRES-AMAZON-ELECTRONICS				
Metric	MRR	Hits@5	Hits@10	Hits@50	Hits@100	MRR	Hits@5	Hits@10	Hits@50	Hits@100
Static graph models on relational event graph										
GCN	6.1±1.6	7.4±2.1	10.0±2.5	23.4±3.0	35.3±1.3	13.1±0.6	15.9±0.7	21.5±0.6	45.9±1.3	60.6±1.6
GAT	7.2±2.5	7.8±2.7	10.2±2.9	23.8±3.6	38.4±1.9	13.2±0.7	15.5±0.9	20.7±1.0	45.2±1.2	61.0±1.5
TConv	4.2±0.3	4.8±0.7	7.7±1.0	23.5±1.6	35.5±2.0	17.3±0.3	24.1±0.4	34.3±0.5	63.0±0.6	73.7±0.4
Static graph models on relational event graph + sequence embedding from personal event data										
GCN+S	4.5±0.3	5.5±0.6	9.3±0.8	29.0±0.5	40.4±0.4	14.7±0.5	19.1±0.8	27.2±1.5	57.9±1.9	70.6±1.5
GAT+S	7.7±2.1	8.5±2.1	12.0±1.8	31.3±1.3	46.3±0.6	14.4±1.7	16.7±1.8	21.6±1.4	43.6±2.5	58.4±3.4
TConv+S	7.5±0.5	9.6±0.7	14.7±0.9	**37.9±0.9**	**50.6±0.4**	22.3±1.6	32.1±2.4	**44.5±2.3**	**70.1±1.1**	**78.1±0.8**
Static graph models on relational event graph + personal event nodes										
GCN_RP	**8.7±1.4**	9.2±1.4	10.5±1.4	18.1±1.2	25.8±2.5	7.5±0.6	8.3±0.6	10.9±0.8	21.8±2.3	29.0±3.2
GAT_RP	6.5±1.0	7.9±1.0	10.9±1.4	25.7±3.2	39.8±3.7	15.5±0.5	17.2±0.4	20.5±0.7	35.5±3.0	46.9±3.6
TConv_RP	4.7±0.3	5.7±0.4	8.4±0.7	22.9±0.9	34.3±0.9	13.7±0.7	18.2±0.9	25.5±1.2	50.8±1.5	64.3±1.1
Temporal graph models on relational event graph + personal event nodes										
TGN	3.5±0.5	4.1±1.2	6.9±1.2	23.7±0.8	39.0±1.3	13.8±0.3	19.2±0.6	26.4±0.9	48.8±0.9	61.5±0.7
DyRep	2.9±0.6	3.0±1.1	5.8±1.6	22.8±2.6	39.6±2.4	6.8±0.7	8.9±1.0	13.8±1.3	33.4±2.3	47.5±3.0
TNCN	8.2±1.8	**11.5±2.7**	**16.3±2.6**	31.6±3.2	39.3±3.2	**25.5±1.8**	**32.2±1.8**	38.4±1.5	56.2±1.7	63.3±2.3

2. In the second set, we use a biderectional transformer sequence model (BERT) [63], to encode each user's last 100 personal events from the training set. The resulting user embedding is added as input to the GCN, GAT, and TConv models from the first set, denoted as GCN+S, GAT+S, and TConv+S, respectively.
3. In the third set, we convert each unique personal event into a node and add it to the user graph from the first set, creating edges between users and their personal event nodes. As in the second set, we use only the last 100 personal events per user. We then run GCN, GAT, and TConv on this graph, denoted as GCN_RP, GAT_RP and TConv_RP.
4. Lastly, based on the graph containing user and personal event nodes from the third set, we add timestamp information to construct a temporal graph. For datasets that lack timestamps for relational events, we inject these events randomly into the sequence of personal events. We then run temporal graph models, TGN [31], DyRep [34], and TNCN [35] on this graph.

The sequence model for capturing personal events in the second set is designed as a masked token prediction task using a BERT model with a masking probability of 0.3. A key benefit of using transformer-based models is flexibility in event tokenization. In PRES-BRIGHTKITE and PRES-GOWALLA, personal events are 8-character geohash strings (e.g., 9q8yyk8y|9q8vzj5b|9q8vyzwk). Since geohashes encode hierarchical geographic information, we apply hierarchical tokenization by splitting each into four two-character tokens with added prefixes (e.g., gh12-9q, gh34-8y, gh12-yk, gh12-8y). This roughly mimics hierarchical location modeling, such as identifying continent, country, city, and neighborhood. For the PRES-AMAZON datasets, we apply similar tokenization by splitting each event into three product tokens and one rating token. We do not apply token splitting for PRES-GITHUB.

For performance evaluation, following prior benchmarks [39, 40, 52], we use ranking-based metrics: Mean Reciprocal Rank (MRR) and Hits@k, evaluated at various k depending on the number of negative samples. Each baseline is run five times with different random seeds, and we report the mean and standard deviation of the results.

Experiment results. Table 2 and Table 3 show the experiment results, with additional results available in Appendix F. In each table, bold numbers indicate the best-performing model on a given metric, and underlined numbers indicate the second best. As each dataset has its own characteristics, the results vary across datasets. However, there are some emerging patterns in the results that we highlight below.

- The graph models with personal event sequence embeddings—GCN+S, GAT+S, and especially TConv+S—consistently perform well across all datasets and metrics. On PRES-BRIGHTKITE and PRES-GOWALLA, TConv+S outperforms all models across all metrics, often by a wide margin, especially compared to methods that encode personal events as nodes. On PRES-GITHUB, GAT+S shines, outperforming all methods on all metrics.
- For the AMAZON datasets, TConv+S remains dominant on Hits@k metrics with larger k, achieving the best results at Hits@10, Hits@50, and Hits@100 on PRES-AMAZON-CLOTHING, as well as Hits@50 and Hits@100 on PRES-AMAZON-ELECTRONICS, while holding second place at other k values. At these lower k, where TConv+S ranks second, the best model is TNCN, a temporal graph method. TNCN, however, does not perform well on Hits@k metrics with larger k.
- With the exception of TNCN on the AMAZON datasets with small k metrics, the performance of temporal graph methods (TGN, DyRep, and TNCN) using graphs with personal event nodes is noticeably lower compared to static graph models with sequence embeddings. For the large PRES-GITHUB dataset, they encounter GPU out-of-memory errors, even when using small batch sizes.
- In nearly all cases, converting personal events into nodes and adding them to the relational event graph is less effective than modeling personal events with a sequence model and using the user's sequence embedding as an additional node features to the relational event graph. The performance of the "_RP" versions of static graph models is lower than that of the corresponding "+S" versions across nearly all datasets and metrics, with a few exceptions on the AMAZON datasets for Hits@k metrics with lower k.

Table 3: Relational event predictions on PRES-GITHUB.

Method	MRR	Hits@3	Hits@5	Hits@10	Hits@30
GCN	54.0±7.2	62.9±7.5	69.6±5.1	75.2±2.3	80.1±0.4
GAT	69.3±3.1	73.6±2.2	76.1±1.3	78.1±0.4	80.4±0.3
TConv	54.5±1.9	62.8±2.1	69.2±1.5	74.6±0.7	78.0±0.2
GCN+S	70.8±0.8	75.1±0.2	76.9±0.0	78.5±0.1	80.9±0.4
GAT+S	**74.2±0.6**	**77.0±0.4**	**78.7±0.3**	**80.6±0.1**	**84.5±0.2**
TConv+S	62.7±1.0	70.6±0.7	74.7±0.4	77.4±0.2	78.8±0.1
GCN_RP	22.3±2.6	23.3±2.9	28.8±3.1	37.8±3.3	57.5±3.5
GAT_RP	33.1±4.1	35.7±5.4	43.9±5.6	57.2±4.5	76.7±0.7
TConv_RP	33.6±1.5	39.2±2.2	50.1±2.2	64.2±1.4	76.3±0.2
TGN	Out of GPU Memory				
DyRep	Out of GPU Memory				
TNCN	Out of GPU Memory				

Even though GCN+S, GAT+S, and TConv+S perform relational event prediction in two stages, first generating user embeddings from personal event sequences and then incorporating them into the graph learning process, they still perform well across datasets. In contrast, TGN, DyRep, and TNCN use a single-step approach that directly integrates temporal dynamics but operates on graphs where personal events are represented as nodes. These differences highlight an opportunity for future exploration of how best to represent the temporal dynamics of personal events within a user while jointly modeling the full structure that includes user-to-user relational events, in an end-to-end fashion.

5.2 Personal event prediction tasks

Experiment setup. We perform personal event prediction experiments on all PRES datasets except PRES-GITHUB. In these experiments, we evaluate several sets of baseline methods:

1. The first model is a sequential model that uses only personal event data. We use a bidirectional transformer architecture (BERT) with a prediction head to compute the likelihood of a user having a particular personal event in the future. For each user, we use the last 100 personal events in the training set to predict the likelihood of future events.

Table 4: Performance results for personal event prediction tasks (all metrics are in percent).

Method	PRES-BRIGHTKITE					PRES-GOWALLA				
Metric	MRR	Hits@3	Hits@5	Hits@10	Hits@50	MRR	Hits@3	Hits@5	Hits@10	Hits@50
Sequential models										
BERT	34.2±0.1	35.6±0.2	37.4±0.2	40.1±0.2	50.1±0.3	15.3±0.2	15.7±0.3	18.6±0.3	23.4±0.3	43.1±0.3
BERT-n2v-p	33.8±0.1	35.1±0.2	36.9±0.2	39.6±0.2	49.8±0.2	14.4±0.2	14.8±0.2	17.7±0.2	22.6±0.2	42.4±0.2
BERT-n2v-i	**34.4±0.1**	**35.9±0.1**	**37.6±0.1**	**40.3±0.1**	50.3±0.2	15.0±0.3	15.4±0.3	18.3±0.3	23.2±0.3	42.7±0.3
Graph models on personal event only graph										
GCN	24.9±1.2	27.1±1.5	31.8±1.7	38.8±1.9	55.8±1.0	28.2±3.2	29.7±3.3	34.2±3.0	41.5±2.3	63.8±0.9
GAT	19.0±1.4	20.3±1.7	24.9±1.9	32.1±2.0	52.3±1.6	15.4±1.2	15.4±1.4	20.0±1.5	28.4±1.6	59.3±1.1
TGN	23.5±0.2	24.5±0.3	28.9±0.4	37.1±0.8	54.5±1.1	10.7±0.4	10.6±0.6	14.4±1.1	21.5±1.8	42.7±4.9
DyRep	19.8±2.9	21.4±3.2	26.5±2.5	35.4±1.6	**57.2±1.7**	7.4±0.6	6.4±0.8	10.0±1.0	17.8±1.0	42.9±2.1
Graph models on personal and relational event graph										
GCN_PR	25.4±1.2	27.5±1.3	31.9±1.5	38.2±1.7	54.7±1.6	**30.3±5.1**	**32.0±5.4**	**36.8±5.2**	**44.2±4.4**	**65.4±1.2**
GAT_PR	18.8±0.5	20.3±0.6	25.2±0.6	32.9±0.6	53.3±0.6	16.0±0.6	16.0±0.7	20.5±0.8	28.6±0.9	59.2±0.9
TGN_PR	29.5±2.3	33.5±2.2	35.3±2.2	36.0±2.4	36.1±2.4	14.0±2.0	15.7±2.2	17.0±2.4	17.9±2.5	18.2±2.6
DyRep_PR	23.4±2.7	27.5±3.5	30.5±3.2	32.7±4.2	33.3±4.8	10.5±1.4	11.4±1.8	12.4±2.2	13.1±2.8	13.6±3.4

Method	PRES-AMAZON-CLOTHING					PRES-AMAZON-ELECTRONICS				
Metric	MRR	Hits@3	Hits@5	Hits@10	Hits@50	MRR	Hits@3	Hits@5	Hits@10	Hits@50
Sequential models										
BERT	3.3±0.0	2.3±0.0	3.5±0.1	6.1±0.1	22.4±0.2	8.1±0.2	7.7±0.2	10.7±0.3	16.1±0.3	38.1±0.2
BERT+n2v-p	3.4±0.0	2.4±0.0	3.6±0.0	6.2±0.1	22.7±0.2	8.1±0.1	7.7±0.1	10.7±0.2	16.1±0.2	38.2±0.1
BERT+n2v-i	3.3±0.0	2.3±0.0	3.5±0.1	6.1±0.1	22.6±0.2	8.1±0.1	7.8±0.1	10.7±0.0	16.1±0.1	38.0±0.2
Graph models on personal event only graph										
GCN	10.8±1.7	11.2±2.0	14.1±1.7	19.0±1.1	32.8±0.5	13.3±2.0	13.3±2.5	18.4±2.9	27.6±3.1	55.6±0.9
GAT	3.5±0.0	2.6±0.0	4.0±0.1	7.1±0.1	22.7±0.2	7.4±0.4	6.4±0.4	9.6±0.5	16.1±0.7	44.0±0.8
TGN	9.3±0.9	8.0±0.8	12.8±2.2	25.4±6.8	**44.4±3.4**	16.2±1.6	**17.4±2.0**	22.6±1.8	30.8±1.2	54.2±0.8
DyRep	8.9±1.3	8.1±2.4	13.5±4.1	25.1±6.7	43.5±5.7	11.4±0.4	10.6±0.6	15.3±0.8	25.8±1.0	55.1±0.8
Graph models on personal and relational event graph										
GCN_PR	**10.9±1.3**	**11.4±1.5**	14.3±1.3	19.1±0.8	32.8±0.6	**16.6±1.5**	17.3±1.9	22.2±2.1	30.6±2.0	**56.3±0.8**
GAT_PR	3.5±0.1	2.6±0.1	4.0±0.1	7.1±0.2	22.5±0.2	8.0±0.3	7.2±0.4	10.5±0.5	17.2±0.7	44.8±0.6
TGN_PR	9.3±0.7	7.8±1.1	13.9±3.3	30.4±3.7	41.7±1.9	15.6±0.6	16.8±0.8	**22.8±0.8**	**32.7±0.6**	54.0±1.0
DyRep_PR	10.5±0.2	9.7±0.5	**17.1±0.6**	**32.9±0.3**	41.3±0.4	14.3±0.5	15.0±0.7	21.2±0.9	32.3±1.4	53.3±2.8

2. In the second set, we use node2vec [64] to learn the graph structure of relational events and generate a graph embedding for each user. We then incorporate the embedding into the BERT sequence model. We evaluate two versions of the model: (a) incorporating the graph embedding post transformer module and before the prediction head (BERT-N2V-P), and (b) using the embedding as a special input token to the transformer module (BERT-N2V-I).
3. In the third set, we use graph-based models on personal event–only data by creating a bipartite graph of user nodes and personal event nodes, based on the last 100 personal events per user. We run both static graph models (GCN and GAT) and temporal graph models (TGN and DyRep) on this graph.
4. In the last set, we augment the graph in the third set with relational event data by adding relational event edges between users. We then run GCN, GAT, TGN, and DyRep on this graph, denoted as GCN_PR, GAT_PR, TGN_PR, and DyRep_PR, respectively.

Similar to the sequence embedding used in relational event prediction tasks, we apply split tokenization for the BERT model in personal event prediction to allow more flexibility in modeling events. We use the same tokenization scheme for each dataset as described earlier. For evaluation, we report MRR and Hits@k at various values of k. Each baseline is run five times with different random seeds, and we report the mean and standard deviation of the results.

Experiment results. Table 4 shows the results for the personal event prediction task. As in the relational event task, results vary across datasets due to their unique characteristics, with even more variations in this setting. We discuss some of the results as follows.

- The sequence models perform well on PRES-BRIGHTKITE across all metrics. The base BERT model, which uses only personal event data, already shows strong performance. Adding relational event node2vec embeddings may either improve or degrade performance. In PRES-BRIGHTKITE, adding the embedding after the transformer module reduces performance, while

using it as a special input token improves it. However, the changes are relatively minor but sufficient to make BERT-N2V-I the best-performing model on PRES-BRIGHTKITE. Similar minimal changes are observed in other datasets.

- The static graph model, GCN in particular, performs surprisingly well on PRES-GOWALLA. The best performance is achieved by the GCN_PR model, which is trained on data containing both personal and relational events in a graph with user nodes and personal event nodes. GCN_PR also performs relatively well on PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS. However, the GAT-based models perform noticeably worse than their GCN counterparts.
- The temporal graph models perform relatively well on the PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS datasets, particularly on the Hits@5 and Hits@10 metrics. TGN and DyRep perform better on graphs that include both personal and relational events. A notable exception is the Hits@50 metric.

The results show that there is no single model that consistently performs best across all datasets. Some models work well on certain datasets but not on others. The only consistent pattern is that the best-performing models usually use both personal and relational events. This opens up opportunities for designing better models that can effectively integrate both types of information.

5.3 Hypothesis on native end-to-end model architectures for PRES datasets

In the experiment results described above, we have demonstrated that current models leave notable room for performance improvements in both personal and relational prediction tasks. We hypothesize that a significant performance improvement can be achieved by end-to-end models trained simultaneously on both personal and relational event data. There are many ways to design such models; below are some examples:

1. A possible strategy is to start with a user-level Transformer architecture that takes the user event sequence as input, combined with a message-passing architecture that enables information transfer between two users connected by a relational event and propagates that information across the neighbors via a Graph Neural Network (GNN)-like architecture. This design capitalizes on the strengths of both the Transformer and GNN architectures. Transformer architectures have been shown to be the best model for handling sequential information, whereas GNN architectures excel in transferring information across connected users.
2. Another possible architecture candidate is to also start from the Transformer architecture for the sequence, where each sample represents a user event sequence, and allow the attention mechanism to propagate toward other user sequences that are connected via a relational event, in addition to the attention mechanism inside each user/sequence.

There could also be multiple other promising architectural paradigms for developing a fully end-to-end model. We leave the exploration of these alternative approaches for future work.

6 Conclusions and Limitations

In this work, we aim to advance user event modeling by introducing a unified framework that captures both personal and relational events. We curate and release a collection of public datasets with corresponding prediction tasks, all aligned under a formalization that integrates both event types to provide a more complete view of user behavior. Through empirical evaluation, we demonstrate that models leveraging both event types consistently outperform those using only one. We also show that existing methods, originally developed for either sequential or relational data, even with some adaptations to handle both (e.g., temporal graph models), are less effective across many of our prediction tasks. These findings highlight the need for further study of unified user event modeling.

A key challenge in this work is in the dataset curation process, as many public datasets have already been collapsed into either graph-only or sequence-only formats, often discarding personal or relational events in the process. While we were able to gather and unify a set of datasets that include both event types, they may not fully capture the diversity and complexity of user event modeling across domains. Another limitation is that our current formulation does not support event-level or user-level features, presenting an opportunity for future work to extend the framework toward feature-aware modeling.

References

[1] Erasmo Purificato, Ludovico Boratto, and Ernesto William De Luca. User modeling and user profiling: A comprehensive survey. *arXiv preprint arXiv:2402.09660*, 2024. 1

[2] Alejandro García Martín, Raquel Martínez González, Andrés García, and Gabriel Villarrubia. A survey for user behavior analysis based on machine learning techniques: current models and applications. *Applied Intelligence*, 51:8110–8127, 2021.

[3] Songgaojun Deng, Maarten de Rijke, and Yue Ning. Advances in human event modeling: From graph neural networks to language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6459–6469, 2024. 1

[4] Zhicheng He, Weiwen Liu, Wei Guo, Jiarui Qin, Yingxue Zhang, Yaochen Hu, and Ruiming Tang. A survey on user behavior modeling in recommender systems. *arXiv preprint arXiv:2302.11087*, 2023. 1

[5] Meng Zeng, Hong Cao, Min Chen, and Yujie Li. User behaviour modeling, recommendations, and purchase prediction during shopping festivals. *Electronic Markets*, 29(2):205–217, 2019. 1

[6] Ahmed Abdel-Hafez and Yanchun Xu. A survey of user modelling in social media websites. *Computer and Information Science*, 6(4):59–71, 2013. 1

[7] Guangyuan Piao and John G Breslin. Inferring user interests in microblogging social networks: A survey. *User Modeling and User-Adapted Interaction*, 28(3):277–329, 2018.

[8] Qixiang Fang, Zhihan Zhou, Francesco Barbieri, Yozen Liu, Leonardo Neves, Dong Nguyen, Daniel L Oberski, Maarten W Bos, and Ron Dotsch. General-purpose user modeling with behavioral logs: A snapchat case study. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2431–2436, 2023. 1

[9] Luisa Hernandez Aros, Ximena Bustamante Molano, Francisco Gutierrez-Portela, and Juan Jose Moreno Hernandez. Financial fraud detection through the application of machine learning techniques: a literature review. *Palgrave Communications*, 11(1):1–15, 2024. 1

[10] Akib Mashrur, Wei Luo, Nayyar A Zaidi, and Antonio Robles-Kelly. Machine learning for financial risk management: A survey. *IEEE Access*, 8:203203–203223, 2020.

[11] Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer. Deep learning for financial applications: A survey. *Applied Soft Computing*, 93:106384, 2020.

[12] S Navid Hojaji, Mahmood Yahyazadehfar, and Bahareh Abedin. Machine learning in behavioral finance: A systematic literature review. *The Journal of Financial Data Science*, 4(3): 129–146, 2022. 1

[13] Yan Zhao, Shoujin Wang, Yan Wang, and Hongwei Liu. Mbsrs: A multi-behavior streaming recommender system. *Information Sciences*, 631:1–17, 2023. 1

[14] Daniel Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Information Sciences*, 179(12):1822–1843, 2009.

[15] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, 2016.

[16] Xavier Amatriain and Justin Basilico. Big & personal: data and models behind netflix recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013. 1

[17] Ahmad H Lashkari, Meng Chen, and Ali A Ghorbani. A survey on user profiling model for anomaly detection in cyberspace. *Journal of Information Security and Applications*, 34:38–56, 2017. 1

[18] Gang Wang, Xinyang Zhang, Shaomei Tang, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Clickstream user behavior models. *ACM Transactions on the Web (TWEB)*, 11(4):1–37, 2017. 1

[19] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. A survey of graph neural networks for recommender

systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems*, 1 (1):1–51, 2023. 1, 22

[20] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020.

[21] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[22] Zihao Li, Chao Yang, Yakun Chen, Xianzhi Wang, Hongxu Chen, Guandong Xu, Lina Yao, and Michael Sheng. Graph and sequential neural networks in session-based recommendation: A survey. *ACM Computing Surveys*, 57(2):1–37, 2024. 1

[23] Shu Chen, Zitao Xu, Weike Pan, Qiang Yang, and Zhong Ming. A survey on cross-domain sequential recommendation. *ArXiv*, abs/2401.04971, 2024. 1

[24] Liwei Pan, Weike Pan, Meiyan Wei, Hongzhi Yin, and Zhong Ming. A survey on sequential recommendation. *arXiv preprint arXiv:2412.12770*, 2024.

[25] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems*, page 102427, 2024.

[26] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Berzhang2406efficientt4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019. 3, 21

[27] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2016.

[28] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206, 2018.

[29] Angelica Liguori, Luciano Caroprese, Marco Minici, Bruno Veloso, Francesco Spinnato, Mirco Nanni, Giuseppe Manco, and Joao Gama. Modeling events and interactions through temporal processes–a survey. *arXiv preprint arXiv:2303.06067*, 2023. 1

[30] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018. 2, 3

[31] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020. 2, 3, 7, 22

[32] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020. 2, 3, 4, 22

[33] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020. 2

[34] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019. 2, 7, 22

[35] Xiaohui Zhang, Yanbo Wang, Xiyuan Wang, and Muhan Zhang. Efficient neural common neighbor for temporal graph link prediction, 2024. *URL https://arxiv. org/abs/2406.07926*. 2, 3, 4, 7, 22

[36] Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*, pages 1–1. PMLR, 2022. 2, 4, 22

[37] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36: 67686–67700, 2023. 4, 22

[38] Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt. Long range propagation on continuous-time dynamic graphs. *arXiv preprint arXiv:2406.02740*, 2024. 2, 3, 4, 22

[39] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36:2056–2073, 2023. 2, 4, 8, 22

[40] Julia Gastinger, Shenyang Huang, Michael Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, et al. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. *Advances in neural information processing systems*, 37:140199–140229, 2024. 2, 4, 6, 8, 20, 22

[41] Hongyuan Mei and Jason M. Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, volume 30, pages 6754–6764, 2017. 3, 21

[42] Simiao Zuo, Haoming Jiang, Zitong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702, 2020. 21

[43] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *International conference on machine learning*, pages 11183–11193. PMLR, 2020. 3, 21

[44] Benjamin Letham, Cynthia Rudin, and David Madigan. Sequential event prediction. *Machine learning*, 93(2):357–380, 2013. 3

[45] Alex Boyd, Robert Bamler, Stephan Mandt, and Padhraic Smyth. User-dependent neural sequence models for continuous-time event data. *Advances in Neural Information Processing Systems*, 33:21488–21499, 2020.

[46] Chenxiao Yang, Qitian Wu, Qingsong Wen, Zhiqiang Zhou, Liang Sun, and Junchi Yan. Towards out-of-distribution sequential event prediction: A causal treatment. *Advances in neural information processing systems*, 35:22656–22670, 2022. 3

[47] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015. 3, 21

[48] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018. 3, 21

[49] Xiaodong Lu, Leilei Sun, Tongyu Zhu, and Weifeng Lv. Improving temporal link prediction via temporal walk matrix projection. *Advances in Neural Information Processing Systems*, 37: 141153–141182, 2024. 4

[50] Jian Gao, Jianshe Wu, and JingYi Ding. Hyperevent: Learning cohesive events for large-scale dynamic link prediction. *arXiv preprint arXiv:2507.11836*, 2025. 4

[51] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 4

[52] Lu Yi, Jie Peng, Yanping Zheng, Fengran Mo, Zhewei Wei, Yuhang Ye, Yue Zixuan, and Zengfeng Huang. Tgb-seq benchmark: Challenging temporal gnns with complex sequential dynamics. *arXiv preprint arXiv:2502.02975*, 2025. 4, 8, 22

[53] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020. 22

[54] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021. 4, 22

[55] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011. 5, 17

[56] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 5, 17

[57] Gustavo Niemeyer. Geohash. *Geohash*, 2008. 5

[58] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. *IBM*, 1966. 5

[59] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015. 5, 17

[60] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 6, 22

[61] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 6, 22

[62] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 2021. 6

[63] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *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, 2019. 7

[64] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. 9

[65] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 1971. 21

[66] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010. 21

[67] Tong Zhao, Yozen Liu, Matthew Kolodner, Kyle Montemayor, Elham Ghazizadeh, Ankit Batra, Zihao Fan, Xiaobin Gao, Xuan Guo, Jiwen Ren, et al. Gigl: Large-scale graph neural networks at snapchat. *arXiv e-prints*, pages arXiv–2502, 2025. 22

[68] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 22

[69] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018. 22

[70] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020. 22

[71] Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. Graph hawkes neural network for forecasting on temporal knowledge graphs. *arXiv preprint arXiv:2003.13432*, 2020. 22

[72] Ying Yin, Li-Xin Ji, Jian-Peng Zhang, and Yu-Long Pei. Dhne: Network representation learning method for dynamic heterogeneous networks. *IEEE Access*, 7:134782–134792, 2019. 22

[73] Hansheng Xue, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Yu Lin. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, pages 282–298. Springer, 2021.

[74] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. Network embedding and change modeling in dynamic heterogeneous networks. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 861–864, 2019. 22

[75] Chongjian Yue, Lun Du, Qiang Fu, Wendong Bi, Hengyu Liu, Yu Gu, and Di Yao. Htgn-btw: Heterogeneous temporal graph network with bi-time-window training strategy for temporal link prediction. *arXiv preprint arXiv:2202.12713*, 2022. 22

[76] Ce Li, Rongpei Hong, Xovee Xu, Goce Trajcevski, and Fan Zhou. Simplifying temporal heterogeneous network for continuous-time link prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1288–1297, 2023. 22

[77] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutional representation learning. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 408–417, 2021. 22

[78] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*, 2019.

[79] Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. Complex evolutional pattern learning for temporal knowledge graph reasoning. *arXiv preprint arXiv:2203.07782*, 2022. 22

[80] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3597–3606, 2020. 22

[81] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. *Advances in Neural Information Processing Systems*, 35:11480–11493, 2022. 22

[82] Jiaqi Zhang, Yu Cheng, Yongxin Ni, Yunzhu Pan, Zheng Yuan, Junchen Fu, Youhua Li, Jie Wang, and Fajie Yuan. Ninerec: A benchmark dataset suite for evaluating transferable recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 22

[83] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. Bars: Towards open benchmarking for recommender systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2912–2923, 2022. 22

[84] Dmitry Osin, Igor Udovichenko, Viktor Moskvoretskii, Egor Shvetsov, and Evgeny Burnaev. Ebes: Easy benchmarking for event sequences. *arXiv preprint arXiv:2410.03399*, 2024. 22

[85] Siqiao Xue, Xiaoming Shi, Zhixuan Chu, Yan Wang, Hongyan Hao, Fan Zhou, Caigao Jiang, Chen Pan, James Y Zhang, Qingsong Wen, et al. Easytpp: Towards open benchmarking temporal point processes. *arXiv preprint arXiv:2307.08097*, 2023. 22

[86] Ivan Karpukhin, Foma Shipilov, and Andrey Savchenko. Hotpp benchmark: Are we good at the long horizon events forecasting? *arXiv preprint arXiv:2406.14341*, 2024. 22

[87] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017. 23

[88] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pages 507–523. Springer, 2020.

[89] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

[90] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020. 23

[91] Yan Hai, Dongyang Wang, Zhizhong Liu, Jitao Zheng, and Chengrui Ding. A study of recommendation methods based on graph hybrid neural networks and deep crossing. *Electronics*, 13 (21):4224, 2024. 23

[92] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12026–12035, 2019.

[93] Kelong Mao, Xi Xiao, Tingyang Xu, Yu Rong, Junzhou Huang, and Peilin Zhao. Molecular graph enhanced transformer for retrosynthesis prediction. *Neurocomputing*, 457:193–202, 2021. 23

[94] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. arxiv 2020. *arXiv preprint arXiv:2001.05140*, 2001. 23

[95] Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

[96] Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 119–130, 2024.

[97] Yinan Huang, Siqi Miao, and Pan Li. What can we learn from state space models for machine learning on graphs? *arXiv preprint arXiv:2406.05815*, 2024.

[98] Zifeng Ding, Yifeng Li, Yuan He, Antonio Norelli, Jingcheng Wu, Volker Tresp, Yunpu Ma, and Michael Bronstein. Dygmamba: Efficiently modeling long-term temporal dependency on continuous-time dynamic graphs with state space models. *arXiv preprint arXiv:2408.04713*, 2024.

[99] Dongyuan Li, Shiyin Tan, Ying Zhang, Ming Jin, Shirui Pan, Manabu Okumura, and Renhe Jiang. Dyg-mamba: Continuous state space modeling on dynamic graphs. *arXiv preprint arXiv:2408.06966*, 2024.

[100] Ali Behrouz, Ali Parviz, Mahdi Karami, Clayton Sanford, Bryan Perozzi, and Vahab Mirrokni. Best of both worlds: Advantages of hybrid graph sequence models. *arXiv preprint arXiv:2411.15671*, 2024. 23

[101] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2901–2908, 2020. 23

[102] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

[103] Luyang Huang, Lingfei Wu, and Lu Wang. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5094–5107, 2020. 23

[104] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024. 23

[105] Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *arXiv preprint arXiv:2504.10499*, 2025. 23

A Dataset Documentation

All datasets presented in this paper are intended for academic research purposes, and their corresponding licenses are listed in this section. They are constructed from publicly available resources described below. In all cases, we perform anonymization by removing any personally identifiable information. User IDs in the original data are replaced with auto-incremented ID numbers.

Download links. The datasets and tasks described in this paper are available for download from the following links:

- Datasets and prediction tasks website and documentation: https://huggingface.co/datasets/capitalone/PRES
- Code for dataset creation and experiment runs: https://github.com/CapitalOne-Research/personal-relational-event-sequence

Dataset source and license information. Below, we describe how the source data was obtained and provide license information for each dataset:

- PRES-GITHUB. This dataset is based on GitHub data collected from the GH Archive website (`https://www.gharchive.org/`) using its HTTP JSON download link. It contains GitHub user activity from January 2025, and user IDs have been anonymized. Content from GH Archive is released under the CC-BY-4.0 license, while the associated code is released under the MIT license.
- PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS. These datasets contain Amazon product reviews and ratings in their respective categories. Both are based on Amazon review data collected by McAuley et al. [59] and hosted at: `https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html`. The Amazon review content is licensed under the Amazon license:

  > By accessing the Amazon Customer Reviews Library ("Reviews Library"), you agree that the Reviews Library is an Amazon Service subject to the Amazon.com Conditions of Use and you agree to be bound by them, with the following additional conditions:
  >
  > In addition to the license rights granted under the Conditions of Use, Amazon or its content providers grant you a limited, non-exclusive, non-transferable, non-sublicensable, revocable license to access and use the Reviews Library for purposes of academic research. You may not resell, republish, or make any commercial use of the Reviews Library or its contents, including use of the Reviews Library for commercial research, such as research related to a funding or consultancy contract, internship, or other relationship in which the results are provided for a fee or delivered to a for-profit organization. You may not (a) link or associate content in the Reviews Library with any personal information (including Amazon customer accounts), or (b) attempt to determine the identity of the author of any content in the Reviews Library. If you violate any of the foregoing conditions, your license to access and use the Reviews Library will automatically terminate without prejudice to any of the other rights or remedies Amazon may have.

- PRES-GOWALLA. This dataset contains user activity on the (now defunct) social network Gowalla. It was originally collected by Cho et al. [55] using the platform's public API and published in the SNAP Dataset Repository [56] (`https://snap.stanford.edu/data/loc-Gowalla.html`). The curator confirmed that SNAP datasets are free to use, but no specific license information is available.
- PRES-BRIGHTKITE. This dataset contains user activity on the (also now defunct) social network Brightkite. It was also originally collected by Cho et al. [55] using the platform's public API and published in the SNAP Dataset Repository [56] (`https://snap.stanford.edu/data/loc-brightkite.html`). The curator confirmed that SNAP datasets are free to use, but no specific license information is available.

B Dataset Contents

Examples of dataset contents. To illustrate the structure of the curated datasets, we provide examples of user event sequences from several PRES datasets. Each table includes both personal and relational events, showing how different types of user activity are represented in our format.

- PRES-BRIGHTKITE and PRES-GOWALLA

Table 5: Example of user event sequence in PRES-BRIGHTKITE and PRES-GOWALLA.

uid	timestamp	event_set	event	other_uid
39	1206596784	personal	9xj6hwkm	<NA>
39	1206596838	personal	9xj3fynm	<NA>
39	1206596871	personal	9xj3fynm	<NA>
39	1235862855	personal	9xj65423	<NA>
39	1250883230	personal	9xj65423	<NA>
39	1254535157	personal	9xj5skbn	<NA>
39	1254535193	personal	9xj5sm00	<NA>
39	1283443369	personal	9q8yyyhs	<NA>
39	<NA>	relational	friendship	0
39	<NA>	relational	friendship	30
39	<NA>	relational	friendship	105
39	<NA>	relational	friendship	1190

- PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS

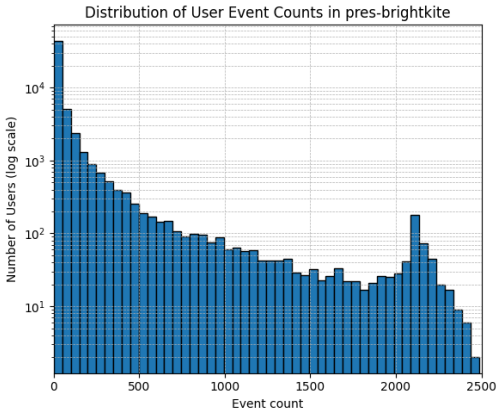
Table 6: Example of user event sequence in PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS.

uid	timestamp	event_set	event	other_uid
254057	1375401600	personal	B000A6PPOK:3	<NA>
254057	1377302400	personal	B003TMPHOU:5	<NA>
254057	1377302400	personal	B004A81PJI:4	<NA>
254057	1377302400	personal	B0054R4AXW:5	<NA>
254057	1377302400	personal	B005CPGHAA:5	<NA>
254057	1377302400	personal	B007FNXMEQ:5	<NA>
254057	1377302400	personal	B007IV7KRU:5	<NA>
254057	1377302400	personal	B007WAWHD4:5	<NA>
254057	1377302400	personal	B008AST7R6:5	<NA>
254057	1377302400	personal	B008R56H4S:5	<NA>
254057	1404086400	relational	co-review_product	107741

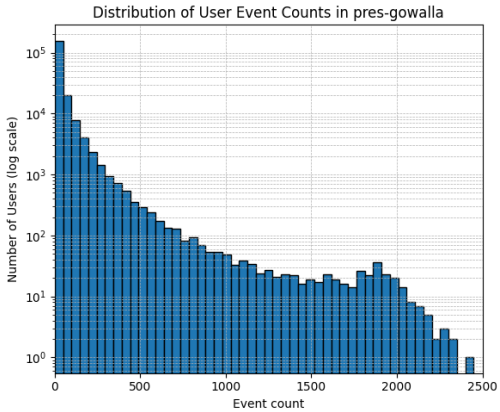
- PRES-GITHUB

Table 7: Example of user event sequence in PRES-GITHUB.

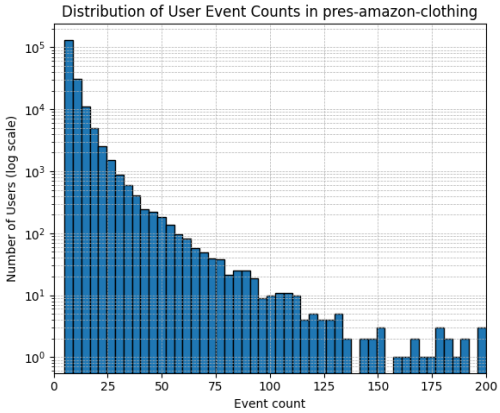
uid	timestamp	event_set	event	other_uid
3669059	1738288160	personal	PullRequestReviewCreated	<NA>
3669059	1738288191	personal	PullRequestReviewCreated	<NA>
3669059	1738288198	personal	PullRequestClosed	<NA>
3669059	1738288200	personal	Push	<NA>
3669059	1738288206	personal	PullRequestClosed	<NA>
3669059	1738288207	personal	Push	<NA>
3669059	1738288217	personal	DeleteBranch	<NA>
3669059	1738288219	personal	DeleteBranch	<NA>
3669059	<NA>	relational	collaborate	824409
3669059	<NA>	relational	collaborate	3126262



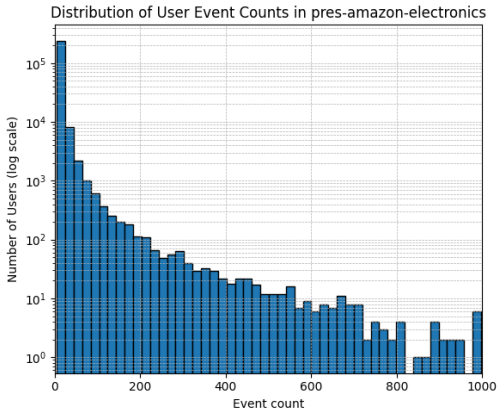
(a) PRES-BRIGHTKITE



(b) PRES-GOWALLA



(c) PRES-AMAZON-CLOTHING



(d) PRES-AMAZON-ELECTRONICS

Figure 2: Histogram of the number of events per user in each dataset.

Event statistics. To characterize user events, we include histograms in Figure 2 and Figure 3 showing the distribution of event counts per user in each dataset. These histograms are constructed by computing the number of events associated with each user and aggregating how many users fall into each count bucket. The y-axis is log-scaled to highlight the long-tailed nature of user behavior, where the majority of users generate only a small number of events, while a much smaller group contributes disproportionately large volumes of activity. This skew is common across datasets and presents both challenges and opportunities for modeling.

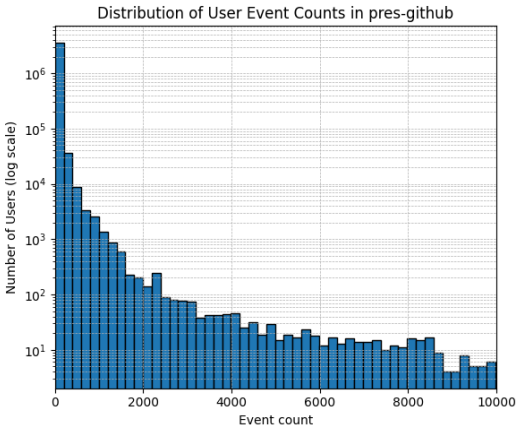


Figure 3: Histogram of the number of event per user in PRES-GITHUB.

C Dataset Construction

This section provides more details about how all of the PRES datasets are created. In short, the data processing files are available in our GitHub repo, following the format `create_datasets/process_X.ipynb` for first generating the processed files, and `create_datasets/task_X.py` for subsequently generating the prediction tasks (including negative sampling for test set), where X is the dataset name.

C.1 PRES-BRIGHTKITE and PRES-GOWALLA

The raw data of users' friendships and check-in histories are stored in text files. We read the friendship files into a dataframe and reformat them into our standard format. The check-in history file contains the latitude and longitude of check-in locations. After reading the files, we convert the latitude-longitude encoding of locations into Geohash-8 string representations and convert the check-in times into Unix timestamp format. Relational events denote friendship between two users, and there are no timestamps associated with this friendship from the raw data. We combine both event types, sort the data by uid and timestamp, and save it into a CSV file (in our dataset repository on Kaggle, this is `processed/X_all_events.csv`).

Now that we have all events saved in a standardized format, we can create our splits and pre-generate negative samples for reproducibility within `create_datasets/task_brightkite.py` and `create_datasets/task_gowalla.py`. For both tasks, we begin by splitting all events into separate dataframes for personal and relational events.

Relational Task: Friendship Prediction. To generate the relational event prediction task, we perform a random split of 70%/10%/20% on the relational event dataframe to generate the train/eval/test datasets, and store them into CSV files. Following Gastinger et al. [40], we adopt a *1-vs-1000* negative sampling scheme, in which 1,000 negative events are sampled for each relational event in the prediction set. Negative samples are drawn via uniform random sampling of users, excluding those who already have relational events with the target user in the training set.

Personal Task: Check-in Prediction. To generate the personal event prediction task, we perform timestamp-based splitting on the personal event dataframe into train/val/test sets. We split each user's personal events by taking the last 20% for the test set, the previous 10% for validation, and the remaining 70% for training. We cap the number of events in the test and validation sets to at most 20 and 10 per user, respectively. Because personal events are more frequent than relational ones, we adopt a *1-vs-500* negative sampling scheme. As geohash strings encode hierarchical spatial information (e.g., earlier characters represent broader regions), we apply stratified hierarchical sampling. Specifically, negatives are stratified by shared geohash prefixes, from matching the first five characters to none. Varying the number of matching characters ensures our negatives contain a mix of nearby and distant locations.

C.2 PRES-AMAZON-ELECTRONICS and PRES-AMAZON-CLOTHING

The input zipped JSON containing the raw data is loaded into a Pandas DataFrame. Each row corresponds to a single review by a single user. We drop rows belonging to users with fewer than 5 product reviews, and create anonymous user IDs from the input 'reviewID' column.

Personal events here are simply all the remaining rows; the 'event' column is created by concatenating the product ID with the rating given to it in that review. Relational events are created by finding users who have co-reviewed at least 3 of the same products. Timestamps for personal events are naturally the time at which the review was posted. For relational events, the timestamp corresponds to when this co-review condition is first met.

The schema for both of these personal and event dataframes are homogenized. At the end of `process_amazon-X.ipynb`, we combine both the relational and personal event data and write it out as a single CSV. This is now the input into `task_amazon.py` for each dataset.

Both tasks use a 70% train, 10% validation and 20% test split paradigm, where the test events are the latest 20% of events (either personal or relational) per user, with the validation events immediately preceding the test events. To manage large histories of some users, we cap test and val sets at 20 and 10 events per user, respectively.

Relational Task: Co-Review Prediction. We apply the above logic to only the relational events, and can now identify the maximum timestamp (aka cut-off time) per user in the training data. This is used to split all personal events into 'observed' (before cut-off time) and 'unobserved' (after cut-off time). Only observed personal events are used for training. Mirroring the logic for PRES-BRIGHTKITE and PRES-GOWALLA's relational task, we adopt a *1-vs-1000* negative sampling scheme. Each negative corresponds to a uniform random sampling of user IDs that do not have a true co-review relationship with the user ID of interest.

Personal Task: Product Prediction. Similarly to the relational task, we apply the splitting logic only to personal events. We use the timestamp of the latest train event per user to identify which of that user's relational events are included for training. We adopt a *1-vs-500* negative sampling scheme, aligning with PRES-BRIGHTKITE and PRES-GOWALLA's personal task logic. Negative samples for each personal event (e.g., B001OE3F08:3) are drawn from three sources: (1) the same product with different ratings (e.g., B001OE3F08:5); (2) other personal events not in the user's training data; and (3) samples from the second set with randomly perturbed ratings.

C.3 PRES-GITHUB

The raw data of GitHub users' activities comes as a compressed JSON. We unpack this into a CSV in `create_datasets/github_extract.py`. We aim to remove bot accounts that could add substantial noise by dropping any rows where the user contains *[bot]* in their name or if that user conducted 100,000 or more actions in the dataset.

Next, in `create_datasets/process_github.ipynb` we additionally remove users with fewer than 3 actions. Relational events connect users who have pushed or opened pull requests at least 5 times in the same repository; predicting relational events in this context is about predicting collaborators. Personal events correspond to the rows in the original dataset, which represent only 24 unique GitHub activities. Thus, the personal event task construction used in the previous datasets is not applicable to PRES-GITHUB. We decided to omit this dataset from the set of datasets used for creating personal event prediction tasks. These personal and relational events have their schemas homogenized and are concatenated. The result is saved as `processed/github_all_events.csv` on Kaggle datasets.

Finally, `create_datasets/task_github.py` is used to create train/validation/test splits and negatives for the relational event prediction task. The logic here follows PRES-BRIGHTKITE, including the split proportions and limitations on the maximum number of events withheld for validation and test. The notable difference is that we adopt a *1-vs-300* negative sampling scheme due to the large size of the dataset.

D Additional Related Works

D.1 Event sequence.

Event sequence modeling is a broad topic that covers many different domains which share a similar goal of understanding and potentially predicting future events given past history. In healthcare, being able to predicting patient's upcoming medical visits enables proactive care and better resource allocation for healthcare providers. In manufacturing and industrial setting, modeling sequences of equipment sensor readings or machine states allow for early detection of faults, enabling predictive maintenance and reducing downtime. These problems share common challenges such as modeling temporal dependencies, handling irregular and asynchronous observations.

Temporal point processes (TPPs) provide a powerful tool for modeling discrete events as stochastic processes in continuous time. Classical models like the Poison process and the Hawkes process [65] allow for explicit modeling of event dynamics, including self-excitation and mutual inhibition, through parameterized intensity functions that govern the likelihood of future events. Recent work has introduced neural extensions such as Neural Hawkes Process [41] [42] and the Self-Attentive Hawkes Process [43], which integrate RNNs or attention mechanisms into the intensity function. These models are particularly well suited for fine-grained timestamp prediction and have found applications in finance, healthcare, and user behavior modeling. However, TPPs typically assume a simple structure over events and focus solely on the temporal dimension, making them less suitable for capturing structured dependencies across users or networks.

Sequential recommendation. A closely related application domain is sequential recommendation, where the goal is to predict the next item a user will interact with based on their historical behavior. Early methods employed Markov chains or matrix factorization over time-slided data [66], while more recent models use deep sequence encoders such as GRU4Rec [47], SASRec [48], and BERT4Rec [26], which apply Transformer-based architectures to item sequences. These models have shown strong performance by capturing user preferences over time. However, they typically model each user independently and do not account for interactions among users.

D.2 Graph models.

In parallel, there has been significant progress in graph-based modeling of user interaction, especially through Graph Neural Networks (GNN). While static graphs lack explicit timestamps and do not capture the order of interactions, they can still represent temporal data through careful graph construction, such as building graphs over specific time windows or pruning outdated edges [67]. This setup allows for framing personal and relational event prediction tasks as link prediction (e.g. forecasting probability of user-item interactions) or node classification (e.g. fraud detection). Early GNNs like GCN [60] introduced neighborhood aggregation but were limited by their transductive nature and the requirement of full graph knowledge during training. GraphSAGE [68] addressed these issues by introducing inductive learning via stochastic sampling, while PinSAGE [69] scaled GNNs to billions of nodes via relaxing memory constraints. GAT [61] incorporated attention mechanisms and later HGT [70] extends GAT to heterogeneous graphs. These developments have established GNNs as a powerful tools for relational modeling [19].

Temporal graph. As described in Gastinger et al. [40], temporal graph methods fall into two broad categories: discrete-time and continuous-time. Discrete-time methods exist for both homogeneous [71] and heterogeneous datasets [72–74]. Continuous-time methods arguably preserve more information and can be converted into discrete graphs, but the reverse is not possible [32]. TGN [31] introduces a general framework for modeling continuous-time dynamic graphs, categorizing DyRep [34] as a special case, followed by several other models such as DyGFormer [37], NAT [36], TNCN [35], and CTAN [38]. Other temporal graph models such as HTGN-BTW [75] and STHN [76] propose different extensions of TGN to handle heterogeneous data. Beyond the standard temporal graphs, several methods have also been proposed for modeling temporal knowledge graphs [77–79].

D.3 Benchmark datasets.

Several benchmark efforts have been proposed across related areas. The temporal graph benchmarks include the TGB [39], its heterogeneous and knowledge graph extension (TGB 2.0) [40], and TGB-Seq [52], which include a more complex sequence of edge dynamics. For static graph learning, OGB [53] and its large-scale extension OGB-LSC [54] provide widely used benchmarks. In the recommendation domain, large-scale user-item interaction datasets have been released through benchmarks such as MIND [80], TenRec [81], NineRec [82], and BARS [83]. For event sequence modeling and temporal point processes (TPP), recent benchmarks include EBES [84], EasyTPP [85], and HOTPP [86].

The closest dataset or benchmark work from our paper is the temporal graph benchmark (TGB) by Huang et al. [39]. It contains several datasets that model user behaviors. These datasets can be roughly divided into two categories: (1) user-to-user interaction datasets, such as TGBL-COIN, TGBL-COMMENT, and TGBN-TRADE; (2) user-to-item bipartite interaction datasets, such as TGBL-WIKI, TGBL-REVIEW, TGBN-GENRE, TGBN-REDDIT, and TGBN-TOKEN. The first category of user-to-user interaction datasets are similar to the relational event part of our datasets; however our datasets also contain personal event sequences, in addition to relational events, which are not present in the TGB dataset.

The second category of TGB datasets are bipartite temporal graphs. In our PRES formulation, the interaction of a user to an item can be encoded as a personal event, where an item is represented as an event or a token. However, the personal event abstraction in the PRES formulation can also encode other types of events. An example is illustrated in Figure 1, where User A has both user-item events (product views) and other types of events such as 'Add to cart' and 'Purchase'. Our formulation also contains relational events that model use-to-user interactions.

In addition, formulating an item as a personal event instead of a node enables the flexibility of encoding items that have hierarchical information such as Geohash in our Brightkite dataset. Each character in the geohash encodes increasingly detailed location information. When we encode an 8-letter geohash as a node, we lose the hierarchical information encoded in the geohash. In contrast, if we are not forced to represent an event as a node, we have more flexibility to encode the hierarchical structure of the geohash. For example, in a sequence model, one could tokenize the event freely. A single event could be encoded into multiple tokens.

In terms of the size, the TGB dataset ranges from a small size of 255 node graph (TGBN-TRADE) to nearly a million node graph (TGBL-COMMENT). Our PRES dataset also ranges from a small-to-

Table 8: Hyperparameter configurations for personal event prediction tasks

Model Name	Learning Rate	Batch Size	Epochs	Emb Dim	Heads	Layers	Channels	Max Events	Max Examples	Num Neg Samples	Num Neighbors
BERT	3e-4	1024	10	64	4	4	–	100	50	10	–
BERT-n2v-p	3e-4	1024	10	64	4	4	–	100	50	10	–
BERT-n2v-i	3e-4	1024	10	64	4	4	–	100	50	10	–
GCN	1e-3	1024	10	128	–	2	128	100	–	5	10
GCN_PR	1e-3	1024	10	128	–	2	128	100	–	5	10
GAT	1e-3	1024	10	128	2	2	64	100	–	5	10
GAT_PR	1e-3	1024	10	128	2	2	64	100	–	5	10
TGN	1e-3	4096	10	16/32	–	–	–	100	–	5	10
TGN_PR	1e-3	4096	10	16/32	–	–	–	100	–	5	10
DyRep	1e-3	4096	10	32/64	–	–	–	100	–	5	10
DyRep_PR	1e-3	4096	10	32/64	–	–	–	100	–	5	10

medium size of 58 thousand users (PRES-BRIGHTKITE) to a relatively large dataset of PRES-GITHUB with 3.6 million users. In terms of the number of events (or number of edges in TGB dataset) our PRES datasets are comparable with TGB datasets, and in some cases larger than TGB datasets. The number of edges in TGB datasets range from around 150 thousands edges (TGBL-WIKI) to 44 million edges (TGBL-COMMENT); whereas the number of events in our datasets range from 1.5 million (PRES-AMAZON-CLOTHING) to more than 100 million events (PRES-GITHUB).

D.4 Other research on graph and sequence.

Several studies have been conducted on different settings on temporal and structural dynamics. Some models focus on modeling graph and time series data using spatio-temporal graph [87–90]. Other model combine graph model output and sequence model output in various application areas [91–93]. A recent study focus on tokenizing graph and applying transformers or state space models (SSMs) for graph learning [94–100]. Another studies works on incorporating knowledge graph into language model. [101–103], as well as performing graph-augmented language generation [104, 105].

E Experiment Details

E.1 Hyperparameters

Personal event prediction task. In Table 8, we present the hyperparameters used during the training of various models for personal event prediction tasks. We use the following notations: **Emb Dim** denotes the dimensionality of token embeddings; **Heads** is the number of attention heads; **Layers** refers to the number of hidden layers; **Channels** indicates the number of hidden channels per layer in GAT and GCN models; **Max Examples** is the maximum number of training samples generated per user; **Num Neg Samples** represents the number of negative samples for each (positive) sample; and **Num Neighbors** is the number of neighbors sampled per layer for GNN models. Additionally, due to GPU memory limitations, we reduce the embedding dimensions for the TGN and DyRep models to 16 and 32, respectively, for the PRES-BRIGHTKITE and PRES-GOWALLA datasets, and to 32 and 64 for PRES-AMAZON-CLOTHING and PRES-AMAZON-ELECTRONICS.

Relational event prediction task. In Table 9, we present the hyperparameters used across all models for relational event prediction tasks. Due to memory and time constraints, batch size, number of epochs, and embedding dimensions were adjusted per dataset. All datasets used a batch size of 4096, except for PRES-GITHUB, which used 512. The number of training epochs was set to 5 for PRES-GITHUB, 20 for PRES-GOWALLA and PRES-AMAZON-ELECTRONICS, 100 for PRES-AMAZON-CLOTHING, and 1000 for PRES-BRIGHTKITE. The model checkpoint with the best validation MRR was saved and used for testing. As shown in our results, TGN, DyRep and TNCN could not be run on PRES-GITHUB. For the remaining datasets, the embedding dimension for TGN and DyRep was 128, except for PRES-GOWALLA, which used 64 to avoid GPU out-of-memory errors. TNCN used ‘NCN_mode‘ of 1, an embedding dimension of 64 and a smaller batch size (1024) for all datasets.

Table 9: Hyperparameter configurations for relational event prediction tasks

Model Name	Learning Rate	Batch Size	Epochs	Emb Dim	Heads	Layers	Channels	Num Neg Samples	Num Neighbors
GCN	1e-3	512/4096	5-1000	128	–	2	128	5	10
GCN_PR	1e-3	512/4096	5-1000	128	–	2	128	5	10
GCN+S	1e-3	512/4096	5-1000	128	–	2	128	5	10
GAT	1e-3	512/4096	5-1000	128	2	2	128	5	10
GAT_PR	1e-3	512/4096	5-1000	128	2	2	128	5	10
GAT+S	1e-3	512/4096	5-1000	128	2	2	128	5	10
TConv	1e-3	512/4096	5-1000	128	2	2	128	5	10
TConv_PR	1e-3	512/4096	5-1000	128	2	2	128	5	10
TConv+S	1e-3	512/4096	5-1000	128	2	2	128	5	10
TGN	1e-3	4096	20-1000	64/128	–	–	128	5	10
DyRep	1e-3	4096	20-1000	64/128	–	–	128	5	10
TNCN	1e-3	1024	20-1000	64	–	–	128	5	10

Table 10: Computational Time (in hours) for Different Models and Datasets

Method	Time (h)				
	AMAZON-CLOTHING	AMAZON-ELECTRONICS	BRIGHTKITE	GOWALLA	GITHUB
Relational event prediction tasks					
GCN	0.06±0.00	0.05±0.00	0.60±0.00	0.26±0.00	8.38±0.11
GCN_RP	0.10±0.01	0.17±0.00	0.40±0.00	1.98±0.03	7.39±0.06
GCN+S	0.07±0.00	0.05±0.00	0.61±0.00	0.29±0.00	8.58±0.12
GAT	0.07±0.01	0.08±0.02	0.61±0.00	0.29±0.00	8.41±0.12
GAT_RP	0.15±0.03	0.18±0.01	0.49±0.06	2.07±0.02	7.40±0.06
GAT+S	0.09±0.02	0.05±0.00	0.62±0.00	0.32±0.00	2.52±0.20
TConv	0.05±0.00	0.04±0.00	0.31±0.00	0.50±0.00	19.8±5.38
TConv_PR	0.10±0.00	0.21±0.00	1.86±0.00	2.98±0.04	12.5±6.83
TConv+S	0.05±0.00	0.04±0.00	0.33±0.00	0.57±0.00	20.2±8.79
TGN	0.49±0.03	0.32±0.00	1.06±0.01	4.62±0.10	–
DyRep	0.49±0.02	0.31±0.00	1.03±0.01	4.43±0.07	–
TNCN	0.63±0.01	0.52±0.00	1.37±0.02	2.04±0.05	–
Personal event prediction tasks					
GCN	5.81±0.10	7.14±0.29	1.73±0.02	8.01±0.71	–
GCN_PR	5.80±0.11	7.21±0.29	1.73±0.03	7.45±1.82	–
GAT	5.83±0.10	7.17±0.28	1.73±0.03	7.33±1.82	–
GAT_PR	5.82±0.10	7.24±0.30	1.76±0.02	7.51±1.79	–
TGN	3.94±0.40	4.10±0.10	0.33±0.01	3.12±0.75	–
TGN_PR	4.38±0.39	5.75±0.19	0.81±0.03	7.89±1.78	–
DyRep	2.03±0.38	3.23±0.34	0.38±0.01	4.11±1.03	–
DyRep_PR	4.88±0.10	5.96±0.20	0.78±0.03	7.85±1.86	–
BERT	4.67±0.06	6.30±0.15	2.65±0.02	9.21±1.11	–
BERT+n2v-i	3.41±0.14	4.43±0.19	2.54±0.01	6.40±0.19	–
BERT+n2v-p	3.60±0.22	4.78±0.14	2.52±0.01	6.38±0.20	–

E.2 Computing Resources

We conducted all experiments on a server equipped with 8 NVIDIA Ampere A10G GPUs (24 GB each), 16 CPU cores, and a RAM upper limit of 512 GB. To fully leverage all resources, we parallelized the training runs so that each experiment used a single GPU. Each experiment is designed to be run on a single-GPU machine. Table 10 summarizes the average training time (in hours) and standard deviation for each model across five datasets, categorized by task type. For relational event prediction tasks, lightweight GCN and GAT variants exhibit minimal computational overhead, with training times generally under one hour except on the GitHub dataset. In contrast, temporally expressive models such as TGN and DyRep incur significantly higher costs, especially on large-scale datasets like Gowalla. In personal event prediction tasks, training times increase across the board, with most models exceeding 7 hours on larger datasets, again highlighting the computational demands of modeling fine-grained temporal dynamics.

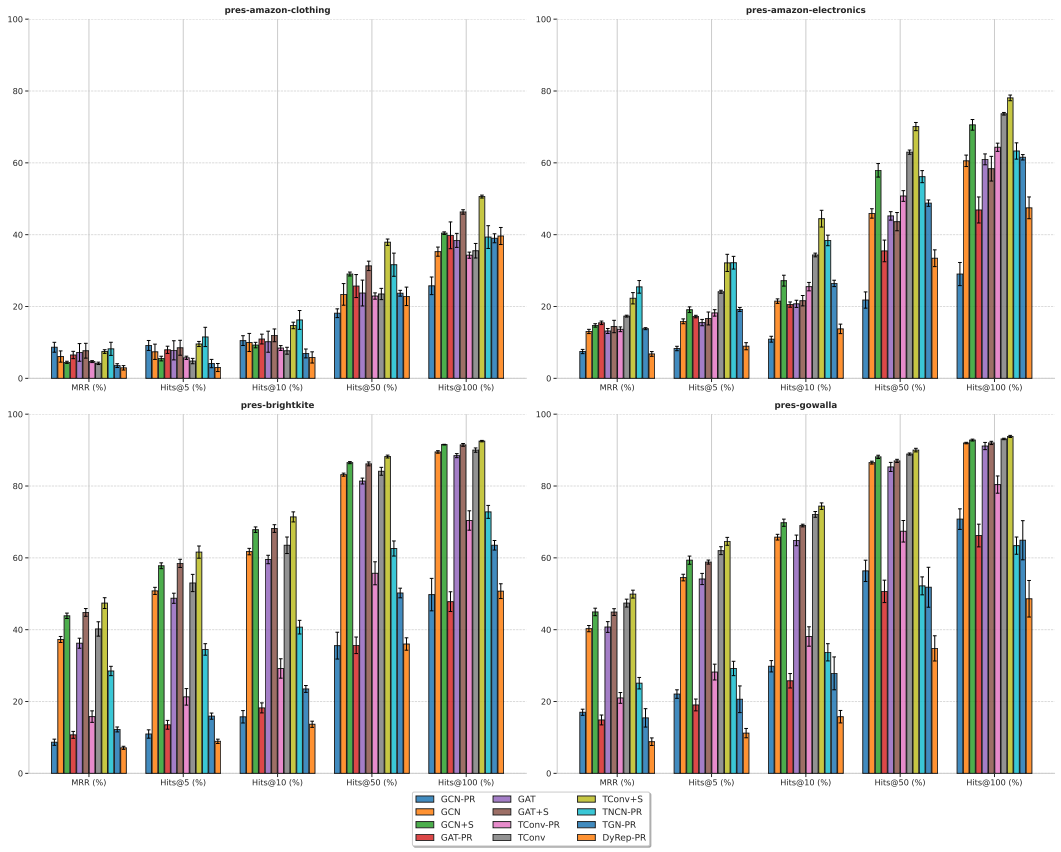


Figure 4: Comparison of relational event predictions across different datasets.

F Additional Experimental Results

Experiment Figures. In Figures 4 and 5, we present the results from the main paper in a more visual format to facilitate comparison across methods. In the relational event prediction tasks, across all datasets and metrics, static GNNs augmented with personal event sequence embeddings (GCN+S, GAT+S, and especially TConv+S) consistently perform well, achieving the best or second-best results. This highlights the benefit of integrating both personal and relational signals. For temporal graph methods, the TGN and DyRep under-perform in most of datasets and most metrics. TNCN perform well on `amazon` datasets on MRR and Hits@k with lower k, but under-perform on other metrics or other datasets. For personal event prediction tasks, BERT+n2v-i offers slight improvements over regular BERT. In particular, BERT-based models exhibit competitive performance in some cases, most prominently on the Brightkite dataset, where they outperform GNN-based counterparts at MMR and lower hit rate thresholds such as Hits@3, Hits@5, and Hits@10.

Additional analysis. One of the main takeaways of the paper is that models leveraging both personal and relational events outperform those using only one type in either relational or personal event prediction tasks. For example, in relational event prediction, the "+S" models (GCN+S, GAT+S, and TConv+S) incorporate sequence embeddings of personal event data into the relational event graph, boosting performance over models that rely solely on relational events (GCN, GAT, or TConv). This highlights the need for models that jointly account for both signals.

We then compare the "+S" strategy to the "_RP" models, which convert each unique personal event into a node and add it to the user-to-user graph used by the GCN model, creating edges between users and their personal event nodes. In most cases, "+S" models outperform "_RP" models, with very few exceptions. In some datasets, such as `pres-brightkite`, `pres-gowalla`, and `pres-github`, adding personal event nodes to the graph even reduces performance. These results may be explained as follows:

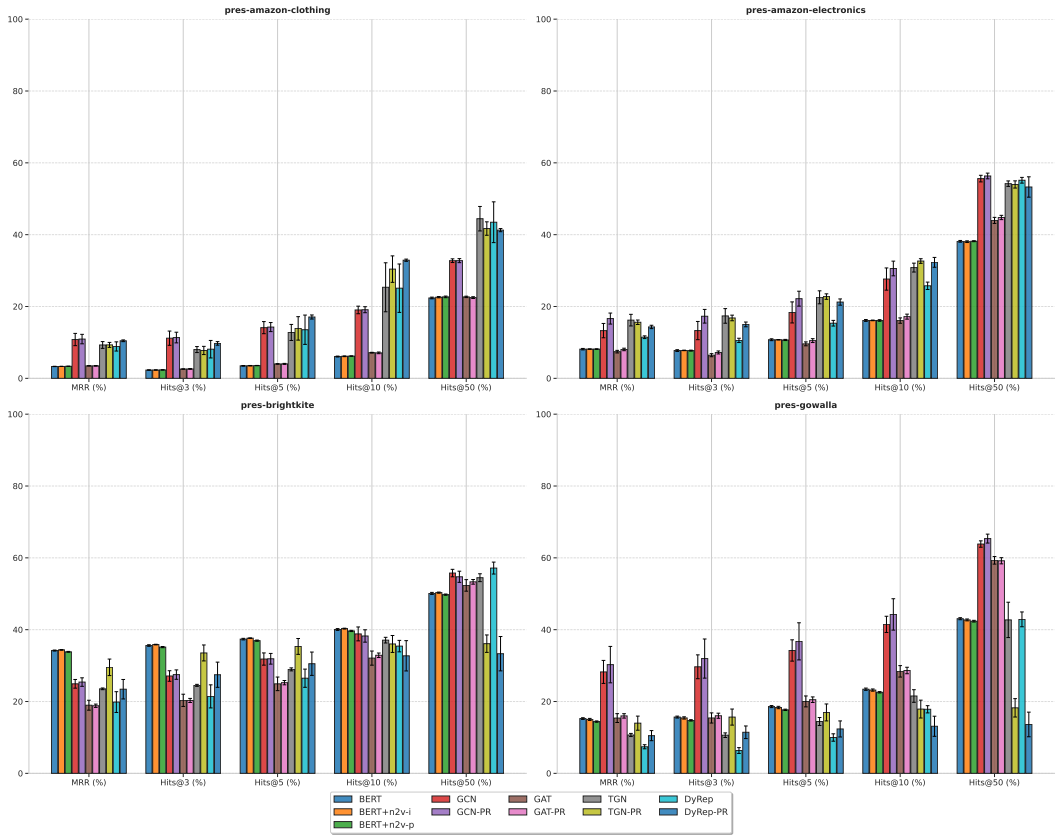


Figure 5: Comparison of personal event predictions across different datasets.

- When we exclude personal events (standard GCN, GAT, and TConv), the model is still able to extract and learn some predictive information from the user-to-user relational events alone to some degree.

- When we include personal events as nodes (GCN_RP, GAT_RP, TConv_RP), this adds more noise than benefit to the system, as the number of personal event nodes is much larger than the number of user nodes. As a result, model performance decreases.

- When we encode personal events as sequence embeddings (GCN+S, GAT+S, and TConv+S), this produces meaningful features without introducing excessive noise. The models are able to capture additional signals from these personal event embeddings, leading to performance improvements.

- In addition, when modeling personal events using a sequence model (BERT) in the "+S" strategy, we retain the hierarchical information of the personal events (such as geohash check-in events in `pres-brightkite` and `pres-gowalla`). In contrast, when we convert personal events into nodes as in the "_RP" strategy, we lose the hierarchical information present in the events.

However, this pattern does not always generalize to every dataset, as we see in `pres-amazon-clothing`, where "_RP" models perform relatively well on MRR and Hits@5, but not on Hits@k metrics with larger k. This suggests that in this dataset, personal event nodes may not merely act as noise in the graphs. Instead, they help the model improve precision on the top candidates (i.e., fewer but more accurate suggestions), at the expense of lower recall coverage for larger k. In addition, the encoded hierarchical information in the product-rating nodes may be less important in this dataset. This observation may influence the architecture design of future models aiming to leverage both personal and relational event signals.

G Broader Impacts

Broader impact. The datasets and prediction tasks we release may support future research on user event modeling, particularly in settings that involve both personal and relational events. Researchers can build models on top of these resources and evaluate them in a consistent way. This can help accelerate empirical progress and facilitate more comparable results. This has potential impact in a range of industry applications where modeling user behavior is critical, such as recommendation, fraud detection, and user interaction analysis.

Potential negative impact. The datasets we release may not cover all use cases of user event modeling, and may reflect only a subset of real-world scenarios. This could introduce bias in model development or evaluation, especially if models are tuned specifically for the structure or properties of our datasets. As a result, there is a risk that future methods may overfit to our datasets and generalize less effectively to other domains or applications.