Neural Higher-order Pattern (Motif) Prediction in Temporal Networks

Yunyu Liu  
*Purdue University*  
LIU3154@PURDUE.EDU

Jianzhu Ma  
*Peking University*  
MAJIANZHU@PKU.EDU.CN

Pan Li  
*Purdue University*  
PANLI@PURDUE.EDU

**Abstract**

Dynamic systems that consist of a set of interacting elements can be abstracted as temporal networks. Recently, higher-order patterns that involve multiple interacting nodes have been found crucial to indicate domain-specific laws of different temporal networks. This posts us the challenge of designing more sophisticated hypergraph models for these higher-order patterns and the associated new learning algorithms. Here, we propose the first model, named *HIT*, for higher-order pattern prediction in temporal hypergraphs. Particularly, we focus on predicting three types of common but important interaction patterns involving three interacting elements in temporal networks, which could be extended to even higher-order patterns. *HIT* extracts the structural representation of a node triplet of interest on the temporal hypergraph and uses it to tell what type of, when, and why the interaction expansion could happen in this triplet. *HIT* could achieve significant improvement (averaged 20% AUC gain to identify the interaction type, uniformly more accurate time estimation) compared to both heuristic and other neural-network-based baselines on 5 real-world large temporal hypergraphs. Moreover, *HIT* provides a certain degree of interpretability by identifying the most discriminatory structural features on the temporal hypergraphs for predicting different higher-order patterns.

**Keywords:** Link Prediction, Network Applications, Hypergraph, Relational Learning, Representation Learning

**1. Introduction**

Graphs provide a fundamental abstraction to study complex systems by viewing elements as nodes and their interactions as edges (Newman, 2003). Temporal graphs track the interactions over time, which allows for a more elaborate investigation of the underlying dynamics of network evolution (Holme and Saramäki, 2012; Kovanen et al., 2011). Recently, numerous network models have been developed to learn the temporal network dynamics and further predict the future status of the complex system (Snijders et al., 2010; Snijders, 2001). However, most of these models focus on predicting lower-order patterns in the graphs, e.g., edge prediction between a pair of nodes (Liben-Nowell and Kleinberg, 2007; Rahman and Al Hasan, 2016; Zhu et al., 2016; Ma et al., 2018).

Higher-order interaction is a universal but more complicated phenomenon in many real-world systems (Benson et al., 2016; Lambiotte et al., 2019; Li and Milenkovic, 2017; Benson et al., 2021).
For instance, when we investigate the social circle of a family, one of the observed patterns we could study is that either one or two of the family members know the newly added member (Fig. 1b-d), where the later case obviously contains even more subtle different network patterns. It could be the case that both of the family members know the new one simultaneously in a common event (Fig. 1d), or separately via different social events (Fig. 1c).

To fully characterize such higher-order patterns, it is necessary to use hypergraphs to model the complex system, in which hyperedges are elements involved in the same event (Berge, 1984). Recently, network scientists have confirmed the value of higher-order patterns in understanding network dynamics of multiple domains such as, social networks, transportation networks, and biological networks (Benson et al., 2016; Lambiotte et al., 2019; Li and Milenkovic, 2017), which posts us a great need to develop new algorithms to model the evolution of temporal hypergraphs. Successful models may be deployed in many multi-nodes-related applications, such as group recommendation (Amer-Yahia et al., 2009) or monitoring synergic actions of multiple genes (Pérez-Pérez et al., 2009).

As shown in Fig. 1, many new patterns might appear when we inspect how the interaction patterns of a group of nodes might change in a time-series manner and across different time scales. Identifying the dynamics that induces these subtle differences needs a more expressive network model. To the best of our knowledge, no previous works have successfully addressed this challenge. Heuristics proposed for edge prediction (Liben-Nowell and Kleinberg, 2007) have been generalized and confirmed for hyperedge prediction (Benson et al., 2018; Yoon et al., 2020), but those models marginally outperform random guessing and are far from ideal to be used in a practical system. Neural networks (NN) have more potential to encode complex structural information and recently achieved great success in various graph applications (Hamilton et al., 2017; Meng et al., 2018; Jiang et al., 2019; Sankar et al., 2020; Trivedi et al., 2019; Kumar et al., 2019; Xu et al., 2020; Rossi et al., 2020a). However, previous works focused on either hyperedge prediction in static networks (Rossi et al., 2020b; Yadati et al., 2020; Cotta et al., 2020; Alsentzer et al., 2020) or simple edge prediction in temporal networks but not for temporal higher-order patterns prediction. Moreover, none of the previous works (neither heuristic nor NN-based ones) are able to predict the entire spectrum of higher-order patterns (e.g., Fig. 1 b-d) let alone their temporal evolution.

Here, we propose the first neural network model, termed HIgher-order Temporal structure predictor (HIT) to predict higher-order patterns in temporal hypergraphs. As proof of the concept, we focus on modeling the interaction patterns among three nodes. We intent to understand and model the interaction expansion procedure on how a two-node interaction incorporates the third node via patterns such as a wedge, an (open) triangle or a (simplicial) closure (Benson et al., 2018) (Fig. 1 b-d). The idea can be further generalized to learn more complicated patterns involving more nodes.

The interaction expansion on temporal hypergraphs could be viewed as an incremental process as when the social circle expands, the number of interaction expansion rarely reduces. We target at predicting the first interaction expansion among a triplet as it is of the most interest. This is far more challenging than conventional edge/hyperedge predictions in temporal networks, as no previous interactions within the same group of nodes can be leveraged.

Figure 1: Incremental processes of the interaction expansion among \( \{u,v,w\} \). The four patterns are formally defined in Sec. 3.
Observation from the data
- At timestamp 6, \( u,v \) form an interaction for the first time.
- Neither of \( u,v \) has interacted with \( w \) before.

Questions on interaction expansion
1. What type of interaction between \( w \) and \( u,v \) will appear (Fig. 1b-d) if there is any?
2. When will such a type of interaction appear?
3. Why will such a type of interaction appear?

A temporal hypergraph with timestamped hyperedges

| t | hyperedges       |
|---|------------------|
| 1 | \{u,c\}, \{w,d,e\} |
| 2 | \{b,u\}, \{w,c\}, \{e,c\} |
| 3 | \{u,c\}, \{a,b,v\} |
| 4 | \{b,u\}, \{w,c,d\} |
| 5 | \{v,e\}, \{w,d,e\} |
| 6 | \{u,v,a\} |

Fig. 2 uses a toy example to demonstrate the input data of our model and the three questions that we are about to answer.

The key intuition of HIT is to encode the temporal network structure around the three nodes of interest using an NN to predict the patterns for their future interactions (Fig. 3). HIT consists of three major components. First, HIT samples a few temporal random walks (TRW) to efficiently collect the contextual structure. In order to model the network dynamics that leads to the first interaction expansion, no historical interactions within the same group of nodes should be referred to, and thus higher-order patterns must be captured in an inductive way via learning from historical behaviours of other node triplets. Towards this goal, the second step of HIT is to represent the node triplets of interest without tracking the node identities by generalizing the recent distance encoding (DE) technique (Li et al., 2020). Our new DE is built upon the counts that one node appears at a certain position according to the set of sampled TRWs while incorporating symmetry and asymmetry to fit the higher-order patterns. Finally, HIT imposes NNs to learn the representation of these DE-based TRWs. HIT adopts different decoders to solve different tasks regarding higher-order patterns. We conclude our contributions as follows,

1. HIT is the first model to predict the full spectrum of higher-order patterns in interaction expansion from two nodes to a third node (Fig. 1 b-d). HIT achieves a significant boost in predicting accuracy (averaged 20% gain in AUC) compared to both heuristic and NN-based baselines.

2. HIT is the first model designed for temporal hypergraphs. HIT provides elaborate time estimation about when interaction expands in certain patterns (uniformly better than baselines).

3. HIT can automatically learn how different types of TRWs contribute to higher-order pattern prediction, which provides certain data insights. To the best of our knowledge, HIT is the first NN-based model that may characterize the discriminatory power of subtle dynamic structural features in temporal networks. We discover some insightful structures that yield a simplicial closure (Fig.1 d), a wedge (Fig.1 b), and no interaction expansion.

2. Related Works

Recently, higher-order patterns in networks have attracted much research interest because of their tremendous success in many real-world applications (Milo et al., 2002; Alon, 2007) including
Figure 3: Different components of HIT encoders: temporal random walks, distance encoding and neural network encoders.

discovering data insights (Benson et al., 2016; Paranjape et al., 2017; Benson et al., 2018; Lambiotte et al., 2019; Do et al., 2020) and building scalable computing algorithms (Yin et al., 2017; Paranjape et al., 2017; Fu et al., 2020; Veldt et al., 2020). Previous works on higher-order structure prediction can be generally grouped into two categories, predicting multiple edges/subgraphs in graphs (Lahiri and Berger-Wolf, 2007; Meng et al., 2018; Nassar et al., 2020; Cotta et al., 2020) and predicting hyperedges in hypergraphs (Zhang et al., 2018; Benson et al., 2018; Zhang et al., 2019; Yoon et al., 2020; Yadati et al., 2020; Alsentzer et al., 2020). Subgraphs, e.g., cliques of nodes (Benson et al., 2016), could be used to describe higher-order patterns. However, they introduce a certain level of modeling confusion. For instance, current subgraph-based approaches were all found to fail to distinguish between triangles (Fig. 1 c) and closures (Fig. 1 d) (Benson et al., 2018), as they may reduce to the same subgraphs 3-cliques. Hypergraph modeling could avoid such ambiguity but previous works mainly focus on predicting whether a hyperedge exists among a node set, which ignores other crucial interaction patterns. Different from these two categories of works, our method aims to predict the entire spectrum of higher-order patterns.

On the other hand, learning dynamics from temporal networks is also challenging. It is typically hard to incorporate simple heuristics such as commute time and PageRanks to encode graph structures to elaborate the complex patterns embedded in the temporal network (Liben-Nowell and Kleinberg, 2007; Benson et al., 2018; Rossi et al., 2020b; Yoon et al., 2020; Nassar et al., 2020) due to their limited model expressivity. Therefore, more powerful NN-based models have been introduced to this domain but mostly are designed for normal temporal graphs instead of temporal hypergraphs. Many NN-based methods need to aggregate edges into network snapshots, which loses important temporal information (Meng et al., 2018; Jiang et al., 2019; Pareja et al., 2020; Manessi et al., 2020; Goyal et al., 2020; Sankar et al., 2020). Other methods that learn node embeddings in the continuous time space may be able to predict higher-order temporal patterns by combining the node embeddings (Trivedi et al., 2019; Kumar et al., 2019; Xu et al., 2020; Rossi et al., 2020a), but their performance was far from ideal due to their undesired inductivity (See the experiments in Sec. 5 and further explanations in Appendix B). Our work of encoding the contextual structure is inspired by a very recent work CAW-N (Wang et al., 2021). However, their method is constructed for the task of edge prediction and not applicable to predicting higher-order patterns in temporal hypergraphs.
3. Preliminaries and Problems Formulation

Temporal hypergraphs are hypergraphs with time information, in which each hyperedge corresponds to an interaction associated with a timestamp.

**Definition 1 (Temporal hypergraph)** A temporal hypergraph $E$ can be viewed as a sequence of hyperedges with timestamps $E = \{(e_1, t_1), (e_2, t_2), \cdots, (e_N, t_N)\}$, $t_1 \leq t_2 \leq \cdots \leq t_N$, where $N$ is the number of hyperedges, and $e_i, t_i$ are the $i$–th hyperedge and its timestamp. Each hyperedge $e$ can be viewed as a set of nodes $\{v_{e_1}^{(e)}, v_{e_2}^{(e)}, \cdots, v_{|e|}^{(e)}\}$ where $|e| \geq 2$.

**Definition 2 (Hyperedge covering)** A node set $S$ is called covered by a hyperedge $e$ if $S \subseteq e$.

For convenience, we use the following two concepts interchangeably in the rest of the paper: Nodes appear in an interaction and nodes are covered by a hyperedge.

Edge prediction is to predict whether two nodes connect to each other. Here we generalize the concept of edge prediction to the higher-order pattern prediction for interaction expansion in hypergraphs.

**Definition 3 (Node triplets of interest)** For three nodes $u, v, w$, any two of them have never interacted before timestamp $t$ and a hyperedge covers $u, v$ at time $t$ but not $w$, then we call $(\{u, v\}, w, t)$ a node triplet of interest.

**Definition 4 (Interaction expansion)** Given a node triplet of interest, say $(\{u, v\}, w, t)$, interaction expansion refers to hyperedges connecting $u, v, w$ and they follow a pattern presented in Fig. 1 b-d.

Compared with conventional edge prediction, interaction expansion prediction has three significant differences. First, instead of making a simple binary decision, the interaction expansion prediction becomes complicated as we incorporate multiple nodes, e.g., three patterns in interaction expansion from two nodes $\{u, v\}$ to another $w$ may happen (Fig. 1 b-d). Second, edge prediction in temporal graphs allows predicting the repetitive edges between two nodes at different timestamps. In contrast, interaction expansion is an incremental process where we need to predict how and when two nodes could extend their interaction for the first time. Although our model can also predict repetitive patterns among the same triplets, it is not the focus of this work. Third, this incremental process implies that any group of nodes may form an hyperedge at some time point (even after infinite time) (Pishro-Nik, 2014). Therefore, it is more reasonable to set a time window $T_w$ and make predictions only within the time window. In particular, we propose three research problems based on these major differences.

**Q1: For a triplet $(\{u, v\}, w, t)$, what type of high-order interaction will be most likely to appear within $[t, t + T_w]$?**

To address this problem, we need to consider four possible interaction patterns as shown in Fig. 1: 1) **Edge.** There is no hyperedge covering $\{u, w\}$ or $\{v, w\}$ in $[t, t + T_w]$. 2) **Wedge.** Either $\{u, w\}$ or $\{v, w\}$ is covered by a hyperedge but not both. 3) **Triangle.** $\{u, w\}$ and $\{v, w\}$ are covered by two hyperedges separately but there does not exist any hyperedge covering all three nodes $\{u, v, w\}$. 4) **Closure.** All three nodes appear in at least one hyperedge simultaneously.

Note that these four patterns could be organized into a certain hierarchy. For instance, to form the Triangle pattern, the three nodes must first form a Wedge pattern within a shorter time. When there is a Closure pattern, hyperedges covering either $u, w$ or $v, w$ (corresponding to Wedge and Triangle)
may still exist. This introduces some fundamental challenges to any computational models that try to predict these patterns. We further consider the next prediction problem to address this challenge.

**Q2: Given a triplet \( \{u, v\}, w, t \) and an interaction pattern in \{Wedge, Triangle, Closure\}, when will \( u, v, w \) first form such a pattern?**

Q2 asks for the exact timestamp of forming each pattern and thus avoids the potential overlap between these patterns. Q2 also shares some implications with Q1 so their predictions should be consistent. That is, if Q2 predicts that Wedge likely happens in a shorter time, Q1 should also assign a high probability to Wedge and vice versa. Answering Q2 may greatly benefit various time-sensitive applications. For instance, predicting the season when three dedicated travelers are grouped enables more accurate advertisement delivery for travel equipment. Predicting the exact timestamp when three genes mutually interact provides valuable information for molecular biologists to precisely control the process of cell development.

**Q3: For a node triplet of interest, which structural features could be indicative to distinguish two patterns such as, Edge v.s. Wedge, Wedge v.s. Triangle, Triangle v.s. Closure?**

Conventional heuristics generalized from edge prediction (e.g., Adamic–Adar Index (Adamic and Adar, 2003)) have been proved as useful features to distinguish Triangle and Closure in static hypergraphs (Benson et al., 2018). We want to investigate whether HIT may leverage representation learning to discover more complicated and insightful structural features that distinguish higher-order patterns in temporal hypergraphs.

## 4. Methodology

In this section, we introduce the framework of HIT with an encoder-decoder structure. The structural encoders are designed to encode the contextual network structure around a triplet of interest. The problem-driven decoders are dedicated to providing answers to the three questions raised in Sec.3.

### 4.1. StructuralEncoders

To encode the structure of temporal hypergraphs, there are two fundamental challenges. First, temporal hyperedges that cover the same group of nodes may appear multiple times over time, which makes conventional structural encoders such as hypergraph GNNs (Zhang et al., 2019; Feng et al., 2019; Yadati et al., 2020) not scalable to model the entire contextual network. Second, as we always focus on the first time when interaction expanding appears, no historical behaviors we could rely on to make the predictions. Therefore, the encoder must be inductive and able to sufficiently represent the network context of the node triplets, which also helps to represent the fundamental dynamics governing network evolution. To address these challenges, we propose to use TRWs to extract the temporal structural features and pair them with asymmetric distance encoding (DE) to obtain the inductive representations, which later could be passed to a neural network to encode the DE-based TRWs.

**Temporal Random Walks.** Given a triplet of interest \( \{u, v\}, w, t \), our model HIT leverages TRWs to efficiently extracts the temporal structural features from its contextual network. Specifically, for each node \( z \in \{u, v, w\} \), HIT samples \( M \) many \( m \)-step TRWs and group them into a set \( S_z \) with only historical hyperedges sampled. The contextual structure can be represented by these three sets of TRWs \( \{S_u, S_v, S_w\} \).
In a temporal hypergraph, more recent hyperedges with more nodes may be more informative. Hence, we consider the \((|e| - 1) \exp(\alpha (t - t_l))\) as the probability to sample the hyperedge for some \(\alpha \geq 0\), where \(t\) is the timestamp of the corresponding hyperedge and \(t_l\) is the timestamp of the previous hyperedge. A larger \(\alpha\) implies the tendency to sample more recent ones while \(\alpha = 0\) gives uniformly sampling. Each walk \(W \in S_z\) can be viewed as a series of (node, time) pairs.

**Asymmetric Distance Encoding.** To be well generalized across the entire network, HIT cannot directly leverage any node identity in the sampled TRWs. However, if node identities are removed, we lose the structural information underlying TRWs. Inspired by the recently proposed DE (Li et al., 2020), we compensate such information loss by generalizing the DE originally designed for static graphs to a new one that could fit hypergraphs. First, we encode a node \(a\) based on the set \(S_z\) of TRWs, denoted by \(g(a; S_z)\), where the node \(a\) is in \(S_u \cup S_v \cup S_w\) and \(z \in \{u, v, w\}\). The encoding \(g(a; S_z), \, z \in \{u, v, w\}\) is computed based on how many times the node \(a\) has appeared on a certain position in \(S_z\). In particular, we set \(g(z; \, S_u)\) as a \((m + 1)\)-dim vector in which the \(i\)-th component can be computed as

\[
g(a; S_z)[i] = |\{W| W \in S_z, \, a \text{ appears at } W[i]\}|, \quad i = 0, 1, 2, ..., m
\]  

Second, we aggregate the encodings \(\{g(a; S_z)\}_{z \in \{u, v, w\}}\) into a vector representation \(I(a|S_u, S_v, S_w)\) by considering the symmetry and asymmetry properties within the triplet \((\{u, v\}, w, t)\).

It is important to properly model the symmetry (inductive) bias of a learning task, which has already yielded successful neural network architectures such as CNN for translating symmetry and GNN for permuting symmetry. In our setting, for a triplet \((\{u, v\}, w, t)\), we have symmetry between \(u\) and \(v\) but they are asymmetric to \(w\). The new DE function \(I(a|S_u, S_v, S_w)\) is designed to maintain such symmetry for all the hyperedges, which could be written as

\[
I(a|S_u, S_v, S_w) = F(\{g(z; S_u), g(z; S_v), g(z; S_w)\})
\]  

Here \(F\) is a mapping and the brace implies the permuting symmetry. As an instantiation, we set \(I(a|S_u, S_v, S_w) = F_1((b + c) \oplus |b - c|)\), where \(b = F_2(g(a; S_u) \oplus g(a; S_v))\), \(c = F_2(g(a; S_u) \oplus g(a; S_w))\), \(\oplus\) denotes the concatenation operation, both \(F_1\) and \(F_2\) are multi-layer perceptrons (MLP) and \(|\cdot|\) is to compute element-wise absolute value.

The new DE function \(I(a|S_u, S_v, S_w)\) ensures HIT is inductive and generalizable. For two triplets of the same form of contextual structure, they induce the same \(I(a|S_u, S_v, S_w)\). Moreover, compared with symmetric DE (Li et al., 2020), asymmetric DE is more expressive because it can encode the direction of interaction expansion, i.e., distinguish between triplets \((\{u, v\}, w, t)\) and \((\{u, w\}, v, t)\). More detailed discussion and theoretical analysis on the generalization capability and expressivity of our DE are left in Appendix A.

Then, we can represent each walk \(W\) by replacing the node identity \(a\) on \(W\) with \(I(a|S_u, S_v, S_w)\):

\[
((z_0, t_0), (z_1, t_1), ..., (z_m, t_m)) \rightarrow W = ((I(z_0), t_0), (I(z_1), t_1), \cdots , (I(z_m), t_m));
\]
here and later, we use \( I(a) \) to present \( I(a|S_u, S_v, S_w) \) for brevity if causing no ambiguity. \((z_i, t_i)\) denotes the \(i\)-th (node, time) pair in the original walk.

**Neural-Network Encoding TRWs.** We adopt a RNN (Rumelhart et al., 1986; Hochreiter and Schmidhuber, 1997) to encode the random walk \( W \) collected for \( S_u \cup S_v \cup S_w \) by \( \text{Enc}(W) = \text{RNN}\{\{I(z_i) \oplus F_3^i(t_i)\}_{i=0}^m\} \) where \( F_3^i(\cdot) \) is a time encoder based on learnable Fourier features (Kazemi et al., 2019; Xu et al., 2019). The learnable Fourier features can transform the scaled time value to a time vector. Generally, it can be viewed as a dimension incremental technique. Compared with other techniques, it has the following advantages: 1) it captures both the periodic and non-periodic features; 2) it is invariant to time-scaling. These two properties are ensured by the Bochner’s Theorem (Bochner, 1992). Therefore, the learnable Fourier features outperform the traditional heuristic time feature as illustrated by Kazemi et al. (2019); Xu et al. (2019).

\[
F_3^i(t_i) = [\cos(\beta_1 t_i) + \phi_1, \cos(\beta_2 t_i) + \phi_2, \cdots, \cos(\beta_d t_i) + \phi_d],
\]

where \( \beta_j, \phi_j, j = \{1, 2, \cdots, d\} \) are learnable parameters. Furthermore, in our instantiation, we simply concatenate this Fourier feature and distance encoding. We then apply a set pooling method \( \text{AGG} \) to calculate the encoding of nodes \( u, v \) and \( w \).

\[
\psi(z) = \text{AGG}\{\{\text{Enc}(W)\}_{W \in S_z}\}, \text{ for } z \in \{u, v, w\}. \tag{4}
\]

Both self-attention pooling and mean pooling may be applied here. For self-attention pooling, we aim to learn a weight for each walk pairs. Then we aggregate the embeddings of all walks by using a weighted sum according to the learned weights. For mean pooling, we simply calculate the mean of the embeddings of all walks.

The detailed implementation is as follows. Let \( z \in \{u, v, w\} \) and recall \( M \) is the number of walks start from one node in \( u, v, \) or \( w \). \( S_z \) is the set of walks starting from \( z \). \( W_i \), where \( 1 \leq i \leq M \) is the \( i \)-th walk in the set \( S_z \). Then the two pooling operations are as follows:

- **Self-Att-AGG(\( S_z \)):** \( \frac{1}{M} \sum_{i=1}^{M} \text{softmax}\{\{\text{Enc}(W_i)^T Q_1 \text{Enc}(W_j)\}_{1 \leq j \leq M}\} \text{Enc}(W_i) Q_2 \),

where \( Q_1, Q_2 \) are two learnable parameter matrices.

- **Mean-AGG(\( S_z \)):** \( \frac{1}{M} \sum_{i=1}^{M} \text{Enc}(W_i) \).

In our experiments, we find using self-attention pooling and mean pooling as AGG both significantly outperform the baselines and self-attention pooling always achieves a slightly higher performance than mean pooling. However, the mean pooling operation is more interpretable. Therefore, we choose mean pooling in our analysis for the question Q3 while self-attention pooling for Q1, Q2.

### 4.2. Problem-driven Decoders

Unlike the encoder, we need to design different decoders to answer three questions defined in Sec.3.

**Decoder for Q1.** Q1 is to predict the types of higher-order patterns of the interaction expansion and thus can be formalized as a standard multi-class classification problem. Similar to DE, we also consider the symmetric property of the problem when designing the decoder function and predict the probability distribution over the four different patterns as \( \hat{Y}(\tau) \equiv \text{softmax}(F_4(\psi(u) + \psi(v) + \psi(w))) \), where \( F_4 \) is an MLP, \( \hat{Y}(\tau) \) is a normalized 4-dim vector (for four classes). We train the model by minimizing the cross entropy loss between \( \hat{Y}(\tau) \) and the ground truth pattern \( Y(\tau) \).
**Decoder for Q2.** In this task, we aim to model the distribution of the time when a certain interaction pattern appears after observing a node triplet of interest. We adopt a log-normal mixture distribution to model the potential multimodality of the distribution of the time. Given a pattern $p \in \{\text{Wedge, Triangle, Closure}\}$, we assume the predicted timestamp $\hat{t}$ follows the distribution, $\log(\hat{t}(\tau, p)) \sim \sum_{i=1}^{k} w_{p,i} \mathcal{N}(\mu_{p,i}, \sigma_{p,i}^2)$, where the weight $w_p$, the mean $\mu_p$ and the standard deviation $\sigma_p$ are parameterized as follows.

$$w_p = \text{softmax}(F_{wp}(h)), \quad \mu_p = F_{\mu_p}(h), \quad \sigma_p^2 = \exp(F_{\sigma_p^2}(h)), \quad \text{where } h = (\psi(u) + \psi(v)) \oplus \psi(w).$$

Here $F_{wp}, F_{\mu_p}$ and $F_{\sigma_p^2}$ are three MLPs with different parameters for different pattern $p$’s.

To train the time predictor, we adopt the maximum likelihood rule. That is, if $\tau$ shows the pattern $p$ first time at $t + t_{\tau,p}$, we minimize the negative log-likelihood loss (NLL) as follows:

$$\text{NLL}(\tau, p) = -\log \left[ \frac{1}{\sqrt{2\pi\sigma_{p,i}^2}} \exp \left( -\frac{(t_{\tau,p} - \mu_{p,i})^2}{2\sigma_{p,i}^2} \right) \right].$$ (5)

**Decoder for Q3.** The decoder for Q3 is expected to hold certain interpretability. We aim to find the most discriminatory structural features, TRWs in particular, for different higher-order patterns. TRWs could be viewed as network motifs (Kovanen et al., 2011) or potentially informative temporal structures. Given two different patterns $p_1, p_2$, the model interpretation is built upon the following procedure: (a) When classifying the patterns $p_1$ and $p_2$, we learn one score $C_W$ for each TRW; (b) we use the value of $C_W$ to characterize how likely the TRW $W$ may induce $p_1$ as opposed to $p_2$; (c) Those TRWs with the largest and smallest $C_W$ are the most discriminative features.

To accomplish the above procedure, we need to solve two computational challenges: (a) We need to have a reasonable way to categorize different TRWs by defining a reasonable feature space; (b) We expect each category of TRWs to hold a linear impact characterized by $C_W$ on the final prediction such that the interpretability of the model can be well reserved. To address the challenge (a), DE gives a good categorization of TRW: Two different TRWs with the same DE, i.e., $I(W[i]|S_u, S_v, S_w) = I(W'[i]|S_{u'}, S_{v'}, S_{w'})$, $\forall i$, belong to the same category. To obtain more visualizable results, we adopt an even simpler DE by replacing $g(a; S_z)$ (Eq.(1)) with an indicator of an estimated shortest path distance (SPD) between $a$ and $z$, i.e.,

$$\tilde{g}(a; S_z) \triangleq i \quad \text{where } i = \min \{j | \exists W \in S_z, a \text{ appears at } W[j] \}. \quad \text{(6)}$$

To solve the challenge (b), we adopt a simple linear logistic regression with DE-based TRWs as features. Note that $\tilde{g}$ can be obtained via a surjective mapping of the original $g$ (Eq.(1)) and thus has less representation power. However, we observe HIT equipped with $\tilde{g}$ still significantly outperforms all baselines to identify higher-order patterns.

In order to find indicative TRWs for patterns $p_1$ v.s. $p_2$, we use $\tilde{g}(a; S_z)$ to compute DE and then minimize the following loss function, which is essentially a logistic regression

$$\sum_{\tau : Y(\tau) \in \{p_1, p_2\}} [x(\tau) \ast 1_{Y(\tau) = p_1} - \log(1 + \exp(x(\tau)))] \quad \text{where } x(\tau) = \sum_{W \in S_{u} \cup S_{v} \cup S_{w}} C_W + b \quad \text{(7)}$$

where $C_W = B^T Enc(W)$ ($Enc(W)$ defined in Eq.(4)), $B, b$ are learnable weights and bias, and $Y(\tau)$ is the ground truth label of a node triplet $\tau$. Eq.(7) can be achieved by setting AGG in Eq.(4) as sum pooling plus a learnable bias. A larger $C_W$ refers to a TRW that is more indicative to $p_1$ as opposed to $p_2$ and vice versa.
Table 1: Basic statistics of the datasets. a) The numbers of different patterns in \{Wedge, Triangle, Closure\} within a time window \(T_{w} = 0.1\) * the total time range after initial interactions between the first two nodes get observed. *Ave. size is the averaged size of hyperedges. b) shows the distributions (PDF) of the times used to expand such interactions to the third nodes.

5. Experiments

Here, we conduct an extensive evaluation of HIT to answer the three questions raised in Sec. 3.

Datasets. We use five real-world temporal hypergraph datasets collected at Benson et al. (2018): Tags-math-sx, Tags-ask-ubuntu, Congress-bills, DAWN, Threads-ask-ubuntu. Their detailed description is left in Appendix C.2. The basic statistics of these datasets are shown in Table 1. Note that repetitive temporal hyperedges significantly enlarge the network. Since the complexity is mainly dominated by the number of hyperedges (from 192K to 2.27M), scalable models are needed. Moreover, the higher-order patterns can be extremely unbalanced. We also show the distributions of time used in interaction expanding for different patterns in Table 1 b). Triangles as expected take more time than Wedges while Closures unexpectedly share a similar tendency as Wedges.

Data Preprocessing and Setup. Ideally, in temporal hypergraphs, hyperedges continue appearing over time but the datasets only include a certain time range \(T\). To avoid the effect of the boundary truncation, we focus on all the node triplets between \([0.4T, 0.9T]\). We use the hyperedges in \([0T, 0.4T]\) to set up the initial network. Node triplets in \([0.4T, 0.75T]\) are selected for model training, \([0.75T, 0.825T]\) for validating and \([0.825T, 0.9T]\) for test. We also set the time window defined in Q1 as \(T_{w} = 0.1T\), so the hyperedges in \([0.9T, T]\) essentially complete the interaction expansion for the node triplets in the test set. We enumerate all the Edges, Wedges, Triangles, and Closures as the data instances with ground truth labels and properly balance all classes (see more details in Appendix C.3). The performance of all methods is averaged across five-time independent experiments.

Baselines. To evaluate our method, we compare HIT with ten baseline methods, including both heuristic methods and NN-based encoders. The detailed description of them is left in the Appendix C.4.

Heuristic methods are the metrics previously used for edge prediction (Liben-Nowell and Kleinberg, 2007) and recently got generalized for hyperedge prediction (Benson et al., 2018; Yoon et al., 2020; Nassar et al., 2020), including 3-way Adamic Adar Index (3-AA), 3-way Jaccard Coefficient (3-JC), 3-way Preferential Attachment(3-PA), and the Arithmetic Mean of traditional AA, JC and PA. They apply to static hypergraphs instead of temporal ones. So we need to do following pre-processing. Given a node triplet \(\{u, v, w, t\}\), we project hyperedges before \(t\) into a static hypergraph, compute these metrics, and then impose a two-layer nonlinear neural network for prediction.

NN-based encoders show great power in learning graph representations. However, none of the previous models can be directly applied to hyperedge prediction in temporal hypergraphs let alone the full spectrum of higher-order patterns. We adopt a hypergraph representation learning model NHP (Yadati et al., 2020) which is proposed for hyperedge prediction in static hypergraphs. For temporal hypergraphs, we aggregate hyperedges into 10 hypergraph snapshots. We train an NHP over...
5.1. Predicting Higher-order Patterns (for Q1)

The results are shown in Table 2. All baselines perform much worse than HIT, which confirms the need for more powerful models to extract subtle structural features to indicate higher-order patterns.

As the training and test datasets are collected from the networks in different time ranges and do not share the exact same data distribution, some baseline approaches do not generalize well and achieve only around 0.5 AUCs. Previous works (Benson et al., 2018; Yoon et al., 2020) demonstrate some possibilities of using heuristic metrics (AA, PA, JC) to distinguish Triangles and Closures in static hypergraphs. However, as shown in Table 2, these metrics fail to give desired predictions when time constraints are incorporated in the experimental setting. We further do some detailed analysis of these heuristic methods. We observe that the measures given by these heuristic methods for different higher-order patterns easily get entangled. More detailed discussion is given in Appendix B.

JODIE (Kumar et al., 2019) and TGN (Rossi et al., 2020a) do not work well because they track node embeddings over time which accumulate noise and do harm to transfer the knowledge of indicative patterns across different node triplets. TGAT (Xu et al., 2020) cannot distinguish nodes that have isomorphic contextual structures and thus fail to predict higher-order patterns. NHP (Yadati et al., 2020) uses network snapshots that lose much temporal information. More detailed discussion on these methods is left in Appendix B.

To better interpret the results, we show the confusion matrices of HIT in Fig. 4 to predict the higher-order patterns in the five networks in Table 2. HIT identifies four patterns reasonably well in all these five datasets. Interestingly, HIT tends to incorrectly predict true Edges more as Triangles/Closures than as Wedges in the two tag datasets. We see that in these two tag datasets, triplets of interest share similar contextual structures are likely to either evolve to Triangles/Closures in a short time or avoid interaction expansion rather than keep being Wedges. Moreover, we see that...
for the DAWN and threads-ask-ubuntu datasets, Triangle and Closure get confused sometimes, while the other three datasets have clearer patterns to distinguish Triangles and Closures.

We also show some results of the ablation study of HIT in Table 3. We demonstrate the significance of time-based TRW sampling. Moreover, DEs that capture both the symmetry and asymmetry of the patterns perform much better than the sum pooling operation adopted in Li et al. (2020), which is crucial to distinguish the patterns such as Wedge v.s. Triangle.

### 5.2. Time Prediction (for Q2)

HIT can predict when a triplet of interest expands the interaction into a certain higher-order pattern for the first time. In this study, we only compare our method with NN-based baselines except NHP because heuristic baselines are found not expressive enough to capture elaborate temporal information, and NHP uses network snapshots and cannot be applied. All the baselines share the same maximum-likelihood decoder for Q2 as HIT defined in Sec. 4.2. We use NLL (Eq. (5)) as the evaluating metrics, which is superior in evaluating how different models learn the underlying distributions of the time. We report the results in Table 4.

HIT almost outperforms all the baselines. This indicates that HIT is able to capture temporal information in the hypergraphs. It’s interesting that for congress-bills and DAWN, all models have a similar performance. The reason is that these two networks are collected by using large time granularity (i.e., timestamps are rough), which naturally limits the time prediction accuracy of

| | Δ AUC | α = 0 | Rv. DE | Sym. DE (Li et al., 2020) |
|---|---|---|---|---|
| tags-ask-ubuntu | -1.59 | -4.42 | -3.13 |
| threads-ask-ubuntu | -2.16 | -3.44 | -2.68 |

Table 3: Ablation study (AUC). Revise each part of HIT and compare with the full model in Table 2. α = 0: Uniformly sample each step of TRWs. Rv. DE: Remove DEs (Eq. 1). Sym. DE: Ignore asymmetry between DEs in Eq. (2). Aggregate three DEs via sum pooling (Li et al., 2020).
different methods. Moreover, there is a hierarchy of different higher-order patterns. That is, to form a triangle, the triplet must first form a wedge. Thus, we need more accurate temporal information about the temporal networks to make precise prediction. Our model, albeit more expressive and accurate, is unable to capture enough temporal information to achieve a better prediction.

### 5.3. Most Discriminatory TRWs (for Q3)

Searching for the important features that induce certain prediction is always a challenging problem in machine learning, especially for NN-based models. Identifying network structural features is even more challenging as they are irregular and strongly correlated by nature. HIT provides possibilities to identify the most discriminatory structural features in the format of TRWs. As illustrated in Sec.4.2 the decoder for Q3, HIT allows categorizing TRWs into different types and summarized the impact of each individual type as a single score from the higher-order pattern prediction.

We evaluate HIT over tags-ask-ubuntu. We rank different types of TRWs based on their discriminatory scores $C_W$ (Eq.(7)) and list the top-2 ranked TRWs in Table 5. Because the NN-based encoder encodes each TRW separately to achieve $C_W$ and the decoder is simply a linear logistic regression, HIT yields very stable model interpretations. The top-2 ranked TRWs always stay in the top-5 ranked ones in the five random experiments.

We further sample 3,000 examples for each classification task and compute the ratio of each type of TRWs that appear as the features for the first class and both two classes (first + second). We see a good correlation between the discriminatory score $C_W$’s and the ratios, which indicates the effectiveness of HIT in selecting indicative TRW features.

Table 5: The TRWs with the top-2 largest and smallest scores $C_W$ (Eq. (7)) to distinguish two patterns in tags-ask-ubuntu. TRWs are categorized based on DEs: Given a node triplet $(u,v,w,t)$, a TRW $W$ is categorized based on $W = (I(W[0]), I(W[1]), I(W[2]))$ where $I(a) = (\{g(a;S_u), g(a;S_v)\}, \{g(a;S_w)\}, \{g(a;S_z)\})$ uses SPD (Eq.(6)) and $x$ indicates SPD $\geq 3$. HIT uses the decoder for Q3. The ratio besides each TRW refers the number of the corresponding TRWs that appear as features for the first class over that for both classes.

| Tasks (AUC) | Closure v.s. Triangle (65.63) | Closure and Triangle v.s. Wedge (57.58) | Wedge v.s. Edge (71.10) |
|------------|-------------------------------|------------------------------------------|------------------------|
| Largest $C_W$ |
| $((x, x), (1, x), (2, x), 2)\{1\}$ | 100.00% | 84.26% | 49.14% |
| $((x, x), (1, x), (2, x), 2)\{2\}$ | 100.00% | 66.13% | 59.26% |
| Smallest $C_W$ |
| $((0, x), (1, x), (1, x), (1, x), 3)\{5\}$ | 38.94% | 34.08% | 24.60% |
| $((0, x), (1, x), (2, x), 3)\{5\}$ | 54.49% | 49.74% | 28.81% |

![Figure 5: Illustration of TRWs based on those blue or green TRWs in Table 5.](image)
Those most discriminatory TRW features shown in Table 5 demonstrate interesting patterns. We further study these patterns by using three examples as illustrated in Fig. 5 (TRWs with blue or green font in Table 5 are drawn). Consider a node triplet \( \{u, v\}, w, t\).

**TRWs that indicate Closures as opposed to Triangles:** They start from \( u, v \) (or \( w \)) and jump to some nodes that are directly connected to \( w \) (or \( u, v \)). This means that \( u, v, w \) tend to form a Closure pattern if both \( u, w \) and \( v, w \) have common neighbors before. We may interpret it as a higher-order pattern closure that generalizes triadic closure in graphs (Simmel, 1950). Without hypergraph modeling, we cannot distinguish Closures and Triangles, and thus cannot observe this higher-order pattern closure.

**TRWs that indicate Wedges as opposed to Closures + Triangles:** They start from \( u \) (or \( v, w \)) and jump to some nodes that are still far away from the other two nodes. These TRWs can be viewed as the opposite of the above higher-order pattern closure. An interesting question is whether they are the most indicative ones for Edges. See the next bullet.

**TRWs that indicate Edges as opposed to Wedges:** Interestingly, they follow a uniform pattern by jumping around \( u \) and \( v \). Therefore, the most indicative features that forbid any interaction expansion are the structures that densely connect \( u \) and \( v \) but disconnect \( w \). On the other hand, combined with the previous bullet, the most indicative features for interaction expansion but from only one node (Wedges) are the structures that connect a single node in the node triplet.

### 5.4. Parameter Sensitivity

For parameter sensitivity, we mainly focus on three parameters \( M, m, \) and \( \alpha \). \( M \) is the number of total walks, \( m \) is the length of each walk. \( \alpha \) is the hyperparameter for hyperedge sampling. A larger \( \alpha \) indicates that HIT is not going to sample the hyperedges that appeared a long time ago. \( \alpha = 0 \) indicates that HIT is going to sample all the hyperedges uniformly.

The sensitivity study of \( M \) is shown in the Fig. 6 a). It is easy to find that under most circumstances, the performance of \( M = 64 \) is higher than \( M = 32 \) and \( M = 16 \). This indicates that a larger \( M \) can ensure good performance. Fig. 6 b) shows that with the fixed \( M = 64 \), different \( m \)'s yield similar results as long as the first-hop neighbors \( (m \geq 1) \) get sampled. For more complicated networks, we expect a larger \( m \) may be needed. The sensitivity study of \( \alpha \) is shown in Fig. 6 c). We choose \( \alpha \) from \( 1e - 8, 1e - 7, 1e - 6, \) and \( 1e - 5 \). The choice of \( \alpha \) should be aligned with the average edge intensity (defined later), where we control the ratio between the average edge intensity and \( \alpha \) between 1 and 1000. This ratio indicates how many edges within a time unit may get sampled.
The average edge intensity is defined as follows. We first expand each hyperedge to a general graph and then compute the average edge intensity according to \(2|E||e|^2/(|V|T)\), where \(|E|\) is the number of hyperedges, \(|e|\) is the average size of hyperedge, \(|V|\) is the number of nodes, \(T\) is the entire time range. For simplicity, we normalize the average edge intensity to \(1e^{-5}\) by rescaling the entire time range \(T\). For threads-ask-ubuntu, the case \(\alpha = 1e^{-6}\) slightly outperforms the other three cases while in general different \(\alpha\)'s give similar results. For tags-ask-ubuntu, a too small \(\alpha\) may introduce performance decay, which demonstrates that most recent hyperedges may provide more informative patterns.

6. Conclusion and Future Works

In this work, we proposed the first model HIT to predict higher-order patterns in temporal hypergraphs to answer what type of, when, and why interactions may expand in a node triplet. HIT can be further generalized to predict even higher-order patterns. In the future, for applications, HIT may be adapted to study higher-order pattern prediction in scientific disciplines such as biology (Zhang et al., 2019; Alsentzer et al., 2020). Methodologically, HIT may be further generalized to predict the entire lifecycles of higher-order patterns as shown in Fig. 5 of Benson et al. (2018).

References

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. Social networks, 25(3):211–230, 2003.

Uri Alon. Network motifs: theory and experimental approaches. Nature Reviews Genetics, 8(6):450–461, 2007.

Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. Subgraph neural networks. In NeurIPS, 2020.

Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. Group recommendation: Semantics and efficiency. Proceedings of the VLDB Endowment, 2(1):754–765, 2009.

Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. Science, 353(6295), 2016.

Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. Proceedings of the National Academy of Sciences, 115(48):E11221–E11230, 2018.

Austin R Benson, David F Gleich, and Desmond J Higham. Higher-order network analysis takes off, fueled by classical ideas and new data. arXiv preprint arXiv:2103.05031, 2021.

Claude Berge. Hypergraphs: combinatorics of finite sets, volume 45. Elsevier, 1984.

Salomon Bochner. A theorem on Fourier-Stieltjes integrals. Collected Papers of Salomon Bochner, 2, 1992.
Leonardo Cotta, Carlos HC Teixeira, Ananthram Swami, and Bruno Ribeiro. Unsupervised joint $k$-node graph representations with compositional energy-based models. In NeurIPS, 2020.

Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. Structural patterns and generative models of real-world hypergraphs. In KDD, 2020.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In AAAI, 2019.

Dongqi Fu, Dawei Zhou, and Jingrui He. Local motif clustering on time-evolving graphs. In KDD, 2020.

Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems, 187, 2020.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. IEEE Data Engineering Bulletin, 40(3), 2017.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In International Conference on Learning Representations, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8), 1997.

Petter Holme and Jari Saramäki. Temporal networks. Physics reports, 519(3), 2012.

Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In IJCAI, 2019.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. arXiv preprint arXiv:1907.05321, 2019.

Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. Temporal motifs in time-dependent networks. Journal of Statistical Mechanics: Theory and Experiment, 2011.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In KDD, 2019.

Mayank Lahiri and Tanya Y Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. In CIDM, 2007.

Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. From networks to optimal higher-order models of complex systems. Nature physics, 15(4):313–320, 2019.

Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. In NeurIPS, 2017.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In NeurIPS, 2020.
David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7), 2007.

Xiaoke Ma, Penggang Sun, and Yu Wang. Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks. *Physica A: Statistical mechanics and its applications*, 496, 2018.

Franco Manessi, Alessandro Rozza, and Mario Manzo. Dynamic graph convolutional networks. *Pattern Recognition*, 97, 2020.

Changping Meng, S Chandra Mouli, Bruno Ribeiro, and Jennifer Neville. Subgraph pattern neural networks for high-order graph evolution prediction. In *AAAI*, 2018.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Huda Nassar, Austin R Benson, and David F Gleich. Neighborhood and pagerank methods for pairwise link prediction. *Social Network Analysis and Mining*, 10(1):1–13, 2020.

Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *WSDM*, 2017.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Scharld, and Charles E Leiserson. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, 2020.

José Manuel Pérez-Pérez, Héctor Candela, and José Luis Micol. Understanding synergy in genetic interactions. *Trends in genetics*, 25(8):368–376, 2009.

H. Pishro-Nik. *Introduction to Probability, Statistics, and Random Processes*. Kappa Research, LLC, 2014. ISBN 9780990637202.

Mahmudur Rahman and Mohammad Al Hasan. Link prediction in dynamic networks using graphlet. In *ECML PKDD*. Springer, 2016.

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*, 2020a.

Ryan A Rossi, Anup Rao, Sungchul Kim, Eunyee Koh, Nesreen K Ahmed, and Gang Wu. Higher-order ranking and link prediction: From closing triangles to closing higher-order motifs. In *WWW*, 2020b.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088), 1986.

Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. DySAT: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM*, 2020.
Georg Simmel. *The sociology of georg simmel*, volume 92892. Simon and Schuster, 1950.

Tom AB Snijders. The statistical evaluation of social network dynamics. *Sociological methodology*, 31(1):361–395, 2001.

Tom AB Snijders, Gerhard G Van de Bunt, and Christian EG Steglich. Introduction to stochastic actor-based models for network dynamics. *Social networks*, 32(1):44–60, 2010.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *ICLR*, 2019.

Nate Veldt, Austin R Benson, and Jon Kleinberg. Minimizing localized ratio cut objectives in hypergraphs. In *KDD*, 2020.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal graphs via casual anonymous walk. In *ICLR*, 2021.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Self-attention with functional time representation learning. In *NeurIPS*, 2019.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *ICLR*, 2020.

Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *KDD*, 2020.

Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *KDD*, 2017.

Se-eun Yoon, Hyungseok Song, Kijung Shin, and Yung Yi. How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In *Proceedings of The Web Conference*, 2020.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS*, 2018.

Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI*, 2018.

Ruoichi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *ICLR*, 2019.

Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10), 2016.
Appendix

This appendix contains the supplementary material for the paper “Neural Higher-order Pattern (Motif) Prediction in Temporal Networks”. In Section A, we discuss why the technique asymmetric DE our model HIT adopts is inductive and generalizable. In Section B, we further discuss the shortcomings of the heuristic baselines and neural network baselines and illustrate the potential reasons why they work poorly in the experiments. Experiments settings and baselines are introduced in Section C. Potential social impact is discussed in Section D.

Appendix A. Discussion on Asymmetric Distance Encoding (Eq. (2))

Many advantages of HIT are built upon the novel asymmetric distance encoding Eq. (2). Next, we will provide a discussion on why it provides good generalization capability and strong expressive power. In particular, the theoretical insights explicitly demystify why HIT achieves far better performance than the heuristic baselines and NN-based baselines.

A.1. Why is HIT that adopts DE inductive and generalizable?

As discussed in the main text, our problem is to predict the first interaction among a node triplet of interest. So the models cannot learn from the historical behaviour among the same node triplet and make such a prediction. Successful models have to be able to learn from the interactions of other triplets in history and further transfer/generalize such knowledge.

Interestingly, our DE is able to capture such transferable/generalizable knowledge. That is, if two node triplets share the same historical contextual network structure, then any nodes that appear on TRWs sampled for these two node triplets hold the same DE distribution. We give the mathematical characterization as follows. First, we need to give several definitions.

Definition 5 (Historical $m$-hop subgraphs) Given a node triplet $\tau = \{u,v\}, w, t$, consider the set of hyperedges that appear before $t$ and hold a shortest path distance (defined below) to either $u$, $v$ or $w$ less than or equal to $m$. Denote this set of hyperedges as $E_{\tau,m}$. The historical $m$-hop subgraph of $\tau$ is the subgraph that consists of the nodes covered by the hyperedges in $E_{\tau,m}$ and the hyperedges in $E_{\tau,m}$. We denote this subgraph as $G_{\tau,m}$.

The above definition needs to define shortest path distance between two hyperedges.

Definition 6 (Shortest path distance between hyperedges) The shortest path distance between two hyperedges $e$ and $e'$ is the minimal number of hyperedges $e_1, e_2, ..., e_k$ where $e_1 = e$, $e_k = e'$ and $e_i \cap e_{i+1} \neq \emptyset$ for $i = 1, 2, ..., k - 1$.

Next, we define subgraph isomorphism between two historical $m$-hop subgraphs as follows, which is used to characterize that the historical behaviors of two node triplets are the same.

Definition 7 (Subgraph isomorphism) We consider two triplets $\tau_1 = \{u,v\}, w, t$ and $\tau_2 = \{u', v'\}, w', t'$. Denote their historical $m$-hop subgraphs as $G_{\tau_1,m}$ and $G_{\tau_2,m}$ respectively. $G_{\tau_1,m}$ and $G_{\tau_2,m}$ are called isomorphic if there exists a bijective mapping $\pi$ from the node set of $G_{\tau_1,m}$ to the node set of $G_{\tau_2,m}$ such that for every temporal hyperedge $(\bar{e}, \bar{t})$ in $G_{\tau_1,m}$, there is a corresponding temporal hyperedge $(\bar{e}', \bar{t}')$ in $G_{\tau_2,m}$ where the time differences are same $t - \bar{t} = t' - \bar{t}'$, and hyperedges of two subgraphs hold the correspondence $\bar{e}' = \{\pi(v) | v \in \bar{e}\}$. Moreover, the two node triplets hold the correspondence that $\{\pi(u), \pi(v)\} = \{u', v'\}, \pi(w) = w'$. 


Next, we give the formal statement of generalization/transferability of DE. An illustration of Theorem 8 is shown in Fig. 7.

**Theorem 8 (DE is transferrable/generalizable)** Consider two triplets \( \tau_1 = (\{u, v\}, w, t) \) and \( \tau_2 = (\{w', v'\}, w', t') \). Suppose the historical \( m \)-hop subgraphs \( G_{\tau_1,m} \) and \( G_{\tau_2,m} \) of \( \tau_1 \) and \( \tau_2 \) are isomorphic where the bijective mapping between the two node sets that preserves such isomorphism is \( \pi \). Denote the sampled sets of \( m \)-step temporal random walks for these two triplets as \( S_1 = S_u \cup S_v \cup S_w \) and \( S_2 = S_{u'} \cup S_{v'} \cup S_{w'} \). The following two statements are true:

For any node \( a \) in \( G_{\tau_1,m} \) and thus \( \pi(a) \) is in \( G_{\tau_2,m} \), the probability that \( a \) appears in one TRW of \( S_1 \) is the same as the probability that \( a' = \pi(a) \) appears in one TRW of \( S_2 \).

Suppose the above two nodes \( a \) and \( a' = \pi(a) \) appear in TRWs. Then, their conditional probabilities are also equal, i.e., \( P(I(a|\{S_u, S_v, S_w\})|a \text{ appears in } S_1) = P(I(a'|\{S_{u'}, S_{v'}, S_{w'}\})|a' \text{ appears in } S_2) \).

**Proof** The above theorem is a result from the fact that the distance encoding \( I(a) \) does not depend on the node identity \( a \) but only depends on the relative position of \( a \) in \( G_{\tau,m} \) given the triplet \( u, v \) and \( w \).

Note that the way to sample TRW according to Algorithm 1 is translation invariant in time: Given a timestamp \( t_l \), the probability of a previous hyperedge appearing at \( t_{l-1} \) and getting sampled is only related to the time difference \( t_{l-1} - t_l \). Sampling TRW does not depend on the absolute timestamps. Moreover, sampling TRW does not need to be variant according to node identities, and it only depends on the network structure. Furthermore, because \( G_{\tau_1,m} \) and \( G_{\tau_2,m} \) are isomorphic, the probability to sample a TRW \( W \) from \( G_{\tau_1,m} \) is equal to that to sample \( \pi(W) \) (specified next) from \( G_{\tau_2,m} \).

More specifically, given a sampled \( W \) from \( G_{\tau_1,m} \), where \( W = ((z_0, t_0), (z_1, t_1), ..., (z_m, t_m)) \), the probability that \( W \) appears in \( S_u \cup S_v \) (or \( S_w \)) is the same as the probability that \( \pi(W) = ((\pi(z_0), t' - t + t_0), (\pi(z_1), t' - t + t_1), ..., (\pi(z_m), t' - t + t_m)) \) appears in \( S_{u'} \cup S_{v'} \) (or \( S_{w'} \) resp.). Furthermore, because each TRW is sampled independently, for any node \( a \) in \( G_{\tau_1,m} \), the probability
that \( a \) appears in the set of sampled TRWs \( S_1 \) must equal the probability that \( \pi(a) \) appears in the set of sampled TRWs \( S_2 \). So we have proved the statement 1.

The above analysis also indicates the probability that we sample three sets of TRWs \( \{S_u, S_v, S_w\}, S_{w'} \) from \( G_{T_{1,m}} \) is equal to the probability to sample three sets of TRWs \( \{S_{u'}, S_{v'}, S_{w'}\} \) from \( G_{T_{2,m}} \) as long as \( \{S_{u'}, S_{v'}\} = \{\pi(S_u), \pi(S_v)\} \) and \( S_{w'} = \pi(S_w) \). We denote this result as condition a).

Moreover, given \( \{S_{u'}, S_{v'}\} = \{\pi(S_u), \pi(S_v)\} \) and \( S_{w'} = \pi(S_w) \), we have \( I(a|\{S_u, S_v, S_w\}) = I(a'|\{S_{u'}, S_{v'}, S_{w'}\}) \) if \( a' = \pi(a) \) because of the definition of DE (Eq.(2)). We denote this result as condition b).

Combining condition a) and condition b), we have \( \mathbb{P}(I(a|\{S_u, S_v, S_w\})|a \text{ appears in } S_1) = \mathbb{P}(I(\pi(a)|\{S_{u'}, S_{v'}, S_{w'}\})|\pi(a) \text{ appears in } S_2) \), which proves the statement ii).

Theorem 8 directly indicates that as long as two triplets show the same historical \( m \)-hop subgraphs, the distributions of the DEs computed based on sampled TRWs must be equal in the sense of probability. This guarantees that DEs will have good generalization capability and transferability.

A.2. Why is asymmetric DE more expressive?

Note that the DE originally proposed in Li et al. (2020) is purely symmetric. Similarly, the labeling trick previously adopted by Zhang and Chen (2018), as a special design of DE for lower-order link prediction in static graphs, is also symmetric. We rephrase the definition of the DE defined in Li et al. (2020) in the context of this work as follows.

Definition 9 (Symmetric DE (Li et al., 2020)) The symmetric DE of a node \( a \) on TRW should follow the function \( I_{\text{sym}}(a|\{S_u, S_v, S_w\}) = \text{set-pool}\{g(a|S_z)|z \in \{u, v, w\}\} \).

A possible choice of symmetric DE is \( \sum_{z \in \{u, v, w\}} \text{MLP}(g(a|S_z)) \), where MLP is a multi-layer perception shared by all \( g(a|S_z) \).

Symmetric DE obviously holds less expressive power. Because it cannot distinguish whether we want to predict the interaction expansion from \( \{u, v\} \) to \( w \) or from \( \{u, w\} \) to \( v \) because \( I_{\text{sym}}(a|\{S_u, S_v, S_w\}) = I_{\text{sym}}(a|\{S_u, S_w, S_v\}) \). However, our asymmetric DE can distinguish them (See the illustration in Fig. 8).

Moreover, setting DE in a concatenation form, i.e., \( I_{\text{cat}}(a|S_u, S_v, S_w) = g(a|S_u) \oplus g(a|S_v) \oplus g(a|S_w) \) can also distinguish whether the interaction expansion to be predicted is from \( \{u, v\} \) to \( w \) or from \( \{u, w\} \) to \( v \). However, it makes the DE lose certain symmetry that the problem keeps: When we switch the node identities of \( u \) and \( v \) in a node triplet of interest \( \{u, v, w, t\} \), the DE should be unchanged because the order of \( u \) and \( v \) is irrelevant. Although our experiments show that when we use the above concatenation-based DE \( I_{\text{cat}}(a) \), the obtained performance is marginally worse than HIT, we may expect performance degeneration of concatenation-based DE to be much larger when the order of the patterns increases. This is because the dimension of concatenation-based DE linearly increases with respect to the order of patterns. Overall, our asymmetric DE captures both the symmetry and asymmetry of the problem and thus is the most proper choice.

Appendix B. The Advantages of HIT over Baselines

B.1. Heuristic Baselines

Traditional heuristic features, including AA, JC, and PA are proved to be able to capture the structural information. Benson et. al. and Yoon et. al. extended them to the hypergraph heuristic
N \ EURAL M \ O TIF P \ REDICTION

Figure 8: Illustration of Symmetric DE (Definition 9) and Asymmetric DE (Eq.(2)). For simplification, we use graphs instead of hypergraphs to illustrate. Moreover, the $g(a|z)$ here adopts the shortest path distance from $z$ to $a$ as Eq.(6) instead of Eq.(1). MLP indicates multi-layer perception.

feature (Benson et al., 2018; Yoon et al., 2020) for higher-order pattern prediction, though for static hypergraphs. They showed that these extended versions may be helpful for higher-order prediction problems.

However, the heuristic features do not perform well in our task. There are two main reasons: First, these methods are heuristically designed and are not expressive enough to represent complex network patterns. They summarize the structural information into several scalars. Most likely, these scalars simply characterize how many neighbors every pair of nodes in the triplet share (in some variant forms), which are rather limited. The problem becomes even more severe when they are used to capture higher-order structural patterns.

Second, the heuristic methods are proposed for static hypergraphs and ignore the temporal information. These methods aggregate all the historical hyperedges before $t$ into a static hypergraph and then calculate the extended or original heuristic features. Temporal information is therefore lost. However, temporal information is crucial in our setting. Intuitively, if two nodes have common neighbors that have appeared more recently, they will have a higher probability to build a connection. On the other hand, although they have common neighbors that have appeared previously a long time ago, the connection between them may be less likely to appear unless something new happens. Apparently, the projected hypergraph without temporal information can not distinguish the differences between these two situations.

Fig. 9 shows that the heuristic features are entangled due to the lack of temporal information. We found that “Closure”, “Triangle”, and “Wedge” entangles severely because these patterns highly depend on temporal information. One example is that to form a “Triangle”, the node triplet must form a “Wedge” first. Without the temporal information, the model will fail to distinguish them.

Overall, these features not only miss capturing the temporal information but also are limited in capturing higher-order structural patterns. HIT, however, can capture both and get a good result.

B.2. Neural-network-based Baselines

As discussed above, for our task, successful models must have good generalization and transferability. NN-based models including continuous-time models JODIE (Kumar et al., 2019), DyRep (Trivedi
The empirical distributions (PDF) of 3-AA for node triplet $\{\{u, v\}, w\}$ under the time constraint $t$ over five datasets.

Figure 10: Illustration of the generalization issue when the model tracks and updates node embeddings.

TGN (Rossi et al., 2020a) and snapshot-based models, which keep tracking and iteratively update node embeddings over time. These models risk losing generalization because of the following reasons. The prediction of these models is a transformation of node embeddings. However, the node embeddings used to predict a certain higher-order pattern in the test dataset could be very different from the node embeddings used for the same type of higher-order pattern in the training dataset. This is because these models keep updating node embeddings, noise may be included in the node embeddings, and the training and test datasets cover entirely different time ranges. Node embeddings keep updated and the distributions of node embeddings during the training and the test time could be very different. It is commonsense that neural network models are over-parameterized and may be sensitive to the noise (Hendrycks and Dietterich, 2018). Small differences in node embeddings may lead to very different predictions, which severely hurts the generalization of the models and thus deteriorates the downstream performance of these models (Fig. 10). This problem becomes even more severe when these models apply to higher-order pattern prediction because higher-order pattern prediction requires more complex neural networks to capture complex features and thus the predictors tend to be more sensitive to the noise. Also, in our setting for interaction expansion prediction, no historical interactions among the node triplets of interest being referred to by these models also have a much higher requirement on model generalization and transferability. Domain adaptation methods may be applied here to relieve the above problem to some extent but they are out of the scope of this work. In contrast, according to the analysis in Appendix A.1, our model HIT yields exactly the same prediction due to the extremely good generalization of DE. So our model HIT does not encounter the above issue on model generalization and transferability.

TGAT (Xu et al., 2020) does not keep track of historical node embeddings anymore and thus is claimed to be inductive. However, TGAT still works very poorly in our experiments. There are two main reasons: First, there is a node ambiguity issue as illustrated in Wang et al. (2021) (See Fig. 11).
To solve this issue, TGAT needs to have high-quality node or edge attributes, which in practice, are not easy to collect such attributes. HIT solves the node ambiguity issue by using DE.

Note that the very recent work CAW-N (Wang et al., 2021) also uses DE to solve the node ambiguity issue but CAW-N only works for lower-order link prediction. Moreover, CAW-N adopts symmetric DE which is not expressive enough for predicting higher-order patterns, as explained in Sec.A.2. This also gets demonstrated by our ablation study in Table 3, where HIT adopting asymmetric DE indeed further improves the performance, also as discussed in A.2.

Second, TGAT is supervised by lower-order link prediction. Basically, given a link of interest, the representations of two end-nodes are computed. The obtained representations may be more powerful to reconstruct the lower-order links. When applied to higher-order pattern prediction as our case, those node representations do not fit well anymore. However, HIT aggregate structural features (via DE) and temporal features that are directly useful to predict higher-order patterns.

Appendix C: Experiments Settings and Baselines

C.1. Environment
We ran all of our experiments on a server with CPU Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz. We try two GPUs, one is Quadro RTX 6000, the other is GTX Titan XP. Although there is a slight fluctuation among the performances from two GPUs, both of our performances outperform the baseline models. The code will be released on GitHub after the paper gets accepted.

C.2. Dataset Description
We use five datasets for evaluation. They are Tags-math-sx, Tags-ask-ubuntu, Congress-bills, DAWN, and Threads-ask-ubuntu. All datasets can be downloaded from https://www.cs.cornell.edu/~arb/data/.

- **Tags-math-sx** is a collection of Mathematics Stack Exchange. Users can post questions on the website and attach up to 5 relative mathematics areas as tags to each question. We denote the areas as nodes and each question as a hyperedge. The timestamps are recorded when questions were posted.

- **Tags-ask-ubuntu** is similar to Tags-math-sx but it is collected from the website Ask Ubuntu. The nodes are the areas related to computer science and the questions posted by the users are the hyperedges connecting different computer science areas.

- **Congress-bills** is a dataset of the bills from sponsors and co-sponsors. These bills are put forth in Congress. We denote the sponsors and co-sponsors as the nodes and the bills as hyperedges. The timestamps are recorded when the bills are introduced.

- **DAWN** records the drug information taken by the patients. Each patient reports the drugs they used before when they visit the emergency department. Here, the drugs are nodes and patients are hyperedges. The timestamps are recorded when patients visit the emergency department.
Threads-ask-ubuntu shows the relationship between the users and threads from the website Ask Ubuntu. The nodes are the users. The users participating in a thread form a hyperedge. The timestamps are recorded when the thread is posted.

C.3. Dataset Preprocessing

There are two different settings in the 5 datasets. For tags-math-sx, tags-ask-ubuntu, and threads-ask-ubuntu the timestamps are recorded at millisecond resolution. For congress-bills and DAWN, the timestamps are recorded at day and quarter resolution respectively. For the timestamps in all datasets, we set the initial times of them to 0 by subtracting the minimum timestamp in that dataset. For tags-ask-ubuntu, tags-math-sx, and threads-ask-ubuntu, we normalize the entire time ranges of all datasets to the same value.

C.4. Baselines and Details of the Experiment Setup

First, no previous baselines can be directly applied to temporal hypergraphs, let alone to predict the full spectrum of higher-order patterns. We properly revise their setting up to make them fit temporal hypergraphs and the settings of higher-order pattern prediction.

Heuristic Methods: Given a triplet of interest \( \{u, v\}, w, t \), we first project all the hyperedges before \( t \) into a static hypergraph. Then we transform the hypergraph to a traditional graph and calculate the corresponding features, including 3-way Adamic Adar Index (3-AA), 3-way Jaccard Coefficient (3-JC), 3-way Preferential Attachment(3-PA), and the arithmetic mean of traditional AA, JC and PA. We have not adopted the PageRank scores generalized for higher-order pattern prediction proposed by Nassar et al. (2020) because computing those scores in temporal hypergraphs may take much time since the hypergraph structures change over time. The next step is to encode these features.

Given a node triplet \((u, v, w)\), the 3-way JC, AA, and PA, can be calculated as follows.

- For 3-AA, \( f_{uvw} = \sum_{i \in N(u) \cap N(v) \cap N(w)} \frac{1}{\log |N(i)|} \).
- For 3-JC, \( f_{uvw} = \frac{|N(u) \cap N(v) \cap N(w)|}{|N(u) \cup N(v) \cup N(w)|} \).
- For 3-PA, \( f_{uvw} = |N(v)| \cdot |N(v)| \cdot |N(w)| \).

where \( f_{uvw} \) is the corresponding feature, \( N(i) \) means the set of neighbours of node \( i \).

For the arithmetic mean of AA, PA, and JC, we calculate the three pairwise features in the traditional way, denote as \( f_{uv}, f_{uw}, \) and \( f_{vw} \). Then we compute the arithmetic means of these features to make predictions. We also try the geometric and harmonic means. The results are similar so we only report the arithmetic means here.

Since for all heuristic methods, the feature is a scalar. We impose a two-layer nonlinear network to first expand the dimension and then do the prediction.

\[
 f_{new} \triangleq F_5(f_{uvw}), \tag{8}
\]

where \( F_5 \) is a nonlinear neural network with hidden dimension 10 and output dimension 4. We use Adam to optimize the neural network with learning rate \( 1e-4 \).

NHP (Yadati et al., 2020) with code is provided here. NHP is designed for hyperedge prediction in static hypergraphs. NHP can also predict the unseen hyperedges when testing. NHP generalizes...
the GCN method to hypergraphs and proposes a novel negative sampling method for hyperedges. Since it cannot handle temporal information, we manually separate the temporal hypergraph into snapshots. We set the 10% of the total time as a snapshot, so totally 10 snapshots. Then we train each snapshot to get the node embeddings for the corresponding time period. Note that, our testing set is from 0.825T to 0.9, our design of splitting snapshots promises that the testing data will not be seen in the training set. We tune the model to get a reasonable embedding. We set the score function to maxmin. We search the GCN hidden size in \{256, 512\} and the number of input node features in \{16, 32, 64\}. After getting the best performance, we export all the node embeddings. We apply the same decoders in our paper to get the corresponding results. We apply the same evaluation metrics as our model. We use Adam to optimize the neural network with learning rate $1e^{-4}$.

**JODIE** (Kumar et al., 2019) with code is provided here. JODIE is proposed to learn the embeddings of users and items in a temporal graph. Here, we adapt it to the temporal hypergraph problem. We first transform the hypergraph to a traditional graph by expanding all the hyperedges to a complete graph. The timestamp of each edge is the same as that of the original hyperedge. Thus, the temporal hypergraph becomes a traditional temporal graph. Then we train JODIE on the traditional graph. For each node, JODIE aims to minimize the $\ell_2$ distances between the predicted embedding and the embeddings of the connected nodes. JODIE processes edges in the temporal order. After processing one edge, JODIE will update the current embeddings with temporal information. Thus, JODIE can capture both temporal and structural information. For testing, the negative distances are considered as the prediction score. Based on the prediction score, we calculate AUC. We split the data according to the setting in Sec. 5. The model is trained for 50 epochs. For tags-math-sx, and tags-ask-ubuntu, we search the dynamic embedding dimension in \{64, 128\} and the best performance is reported. For DAWN and congress-bills, we search the dynamic embedding dimension in \{32, 64\}. For threads-ask-ubuntu we search the dynamic embedding dimension in \{50, 100\}. We choose the setting with the highest AUC. After that, we export all the node embeddings including training set, validating set, and testing set and apply the same decoders in our paper to get the corresponding result. We apply the same evaluation metrics as our model. We use Adam to optimize the neural network with learning rate $1e^{-4}$.

**TGAT** (Xu et al., 2020) with code is provided here. TGAT is designed for link prediction tasks and node embedding tasks in the temporal network. TGAT generalizes the GAT mechanism to the temporal network by selecting both the topological neighbour and temporal neighbour in history. They also apply the Fourier random feature to encode the temporal information. For our setting, we first apply the same strategy with JODIE to transfer the temporal hypergraph to the traditional temporal graph. We train TGAT to get node embeddings. We tune the parameter to get reasonable embeddings for all the nodes. We apply some default settings, which are the bi-linear-type attention and set the number of attention heads to 2, set the number of graph attention layers to 2, and use 100 as their default hidden dimension. We search the degree of their neighbor sampling in \{10, 20, 30\} and choose the best model. We export all the node embeddings and apply the same decoders in our paper to get the corresponding results. We apply the same evaluation metrics as our model. We use Adam to optimize the neural network with learning rate $1e^{-4}$.

**TGN** (Rossi et al., 2020a) with code is provided here. TGN is designed for temporal network problems. It introduces memory modules and combines them with graph attention or other embedding methods to substantially increase efficiency and performance. The memory module stores a vector for each node that appears in the model at time $t$. Each interaction among the nodes is a message. The memory module is updated according to the messages. We search the degree of neighbours to
sample in \{10, 20, 30\} and the dimension of message in \{50, 100\} and choose the setting with the best AUC. Other settings are by default. Then we output all the node embeddings and impose the same neural network as our model HIT for prediction. We use Adam to optimize the neural network with learning rate \(1e^{-4}\).

**HIT** For our model, we search the TRW hyperparameter \(\alpha\) in \{1\(e\) \(-6, 1\(e\) \(-5, 1\(e\) \(-4\}\), the number of TRWs \(M\) in \{64, 128\} and the step length \(m\) in \{2, 3\} and report the best result. Further experiments about parameter sensitivity are conducted and showed in Appendix 5.4. The experiments show that our model is robust when applying different parameters. For the encoder, we set the distance encoding feature dimension to 108. \(F_2\) is a one-layer nonlinear neural network, with output dimension 108. \(F_1\) is a two-layer nonlinear neural network and both hidden dimension and output dimension are 108. For the time encoder in \(F_3\), we set the dimension to 172, i.e., \(d = 172\). For the decoders, \(F_4\) is a two-layer nonlinear neural network, with a hidden dimension 172. Note that the dimension we choose is not crucial and our model is robust enough to a wide range of dimensions. \(F_{w_p}\), \(F_{\mu_p}\), and \(F_{\sigma^2_p}\) are one-layer nonlinear neural networks. We use Adam to train our networks with a learning rate \(1e^{-4}\).

Since the data instances in different classes are extremely unbalanced (as illustrated in Table 1 a)), we uniformly sample the same number of data instances from each class for training/evaluation/testing. The number of sampling instances is governed by the minimum size of the four classes. The sampled balanced dataset is shared across different models. All prediction performance is averaged across five experiments with the same random sampling process.

### Appendix D. Potential Social Impact

In this paper, we try to build machine learning models to predict the higher-order patterns by answering three questions, including what type of, when, and why the interaction expansion could happen. This could be useful in a real life scenario. For example, we can predict whether three men will travel together and send them the advertisement. However, there could be some potentially negative impacts. One issue is about model fairness. One example can be the job market and social networks. Assume we work on the social network data of users from Linkedin with a computer science background. There can be some bias in the data. Since most users on Linkedin with computer science backgrounds are young and male. If we apply HIT to these data, HIT will probably output some biased patterns, e.g., males are more willing to apply for certain types of jobs, such as software engineers, or two men are more easily to become friends, and so on. These patterns ignore the tendency among the female and thus are biased and unfair. Another issue regards privacy. To train a general and accurate model, we will need a lot of information, including the hypergraphs that model simultaneous interactions, precise timestamps, node and edge embeddings, etc. Some of these data may be related to users’ confidential information. Thus, HIT has a potential problem to leak the users’ data when aggregating them all. One method to solve this is to apply federated learning, which we leave as a research topic in the future.