Toward Better Temporal Structures for Geopolitical Events Forecasting

Anonymous ACL submission

Abstract

Forecasting on geopolitical temporal knowledge graphs (TKGs) through the lens of large language models (LLMs) has recently gained traction. While TKGs and their generalization, hyper-relational temporal knowledge graphs (HTKGs), offer a straightforward structure to represent simple temporal relationships, they lack the expressive power to convey complex facts efficiently. One of the critical limitations of HTKGs is a lack of support for more than two primary entities in temporal facts, which commonly occur in real-world events. To address this limitation, in this work, we study a generalization of HTKGs, *Hyper-Relational Temporal Knowledge Generalized Hypergraphs* (HTKGHs). We first derive a formalization for HTKGHs, demonstrating their backward compatibility while supporting two complex types of facts commonly found in geopolitical incidents. Then, utilizing this formalization, we introduce the `htkgh-polecat` dataset, built upon the global event database POLECAT. Finally, we benchmark and analyze popular LLMs on the relation prediction task, providing insights into their adaptability and capabilities in complex forecasting scenarios.

1 Introduction

Temporal knowledge graphs (TKGs) are one of the most common structures used to express temporal relational facts. In its simplest form, a TKG stores information as quadruples, each consisting of two entities, a relation, and a timestamp, which express a typed, directed temporal edge. Recent extensions of TKGs, such as hyper-relational temporal knowledge graphs (HTKGs), allow additional information to be added to the edge in the form of qualifiers, pairs of relations and entities (Ding et al., 2024). Apart from storing information, these structures are commonly used to run prediction queries, finding the most likely element(s) of a given partial fact. One of the more challenging types of such queries is forecasting, which requires the model to make predictions in a time range it has not seen (see Section 3). Making predictions in the geopolitical domain is one of the recurrent use cases of such forecasting queries (García-Durán et al., 2018; Jin et al., 2020; Gastinger et al., 2025). However, a common occurrence in geopolitical events is the involvement of numerous primary entities in an event, which creates a complex fact that is outside the norms of the two-primary-entity system that HTKG relies on. In this work, we aim to address this issue through a generalization of the HTKG structure (see Figure 1 for an overview).

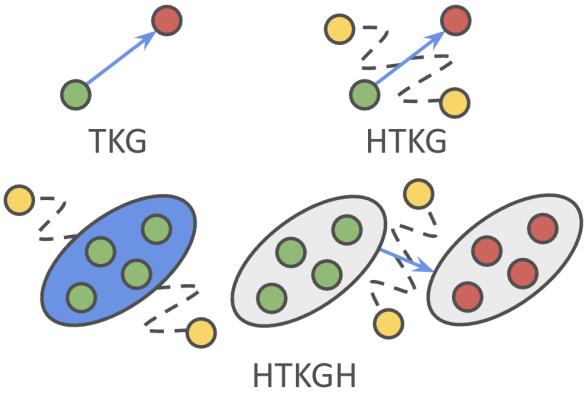


Figure 1: An overview of how facts are expressed in TKGs, HTKGs, and HTKGHs. HTKGHs adds efficient support for complex facts involving many (*i.e.,* more than two) primary entities, expanding the possible expression scenarios such as multi-national treaties.

Examining the stored events in databases such as POLECAT (Halterman et al., 2023a), we observe that many events involve group(s) of entities interacting with each other. Specifically, two types of such events are more common: 1) **Group-Type:** a group of entities (>2) having the same mutual relationship, and 2) **Set2Set-Type:** a group of entities taking an action against another group of entities (>2 total entities). However, trying to express such events within existing structures is difficult,

as the simple structure of directed edges between two entities fails to properly represent them without imposing further issues, such as sparsity (see Section 3.4). To mitigate this problem, this work derives a formalization for hyper-relational temporal knowledge generalized hypergraphs (HTKGHs), which support an arbitrary number of primary entities and second-order edges, enabling efficient expression of more complex geopolitical events (see Section 4). Moreover, we introduce a new dataset, `htkgh-polecat`, which utilizes the HTKGH formalization, based on the POLECAT database. Our statistics show that roughly one in four facts in `htkgh-polecat` follows one of the newly supported complex formats (see Section 5).

In recent years, LLMs have been extensively studied as a new approach to solving TKG queries, as they offer an easy-to-use approach that works out of the box with many structures via small amounts of prompting (Lee et al., 2023; Xia et al., 2024). Such capabilities mainly stem from the extensive pre-training stage they undergo, where they are exposed to copious amounts of information, learning complex syntactical and semantic relationships between concepts and entities. Notably, these capabilities extend beyond simple memorization, as they have demonstrated some level of general pattern recognition even when both semantic and syntactic information are hidden (Lee et al., 2023). One of the common tasks on TKGs is relation prediction, which in many ways mirrors common logical reasoning tasks for LLMs. In this work, we evaluate the reasoning capabilities of LLMs over complex facts using the newly introduced dataset. Our experiments highlight the impressive adaptiveness of LLMs when reasoning over information that is hidden or tampered with in a way that goes against the models' prior beliefs (see Section 6).

To summarize, our contributions are as follows:

- We highlight the shortcomings of HTKG and present an analysis of HTKGH as a generalization that enables efficient expression of complex facts common in geopolitical events.
- We introduce `htkgh-polecat` and its variations, a new dataset based on POLECAT that utilizes HTKGH formalization and employs strict heuristics to ensure validity and density.
- We analyze popular LLMs on the relation prediction task, showcasing their high performance and resilience, while highlighting the effects of various related confounding factors.

2 Related Work

TKG Extensions TKGs are known to struggle with higher-order relational details (Lu et al., 2025). Within the TKG structure, reification (Noy et al., 2006) represents n-ary facts with an intermediate node connected to several binary predicates. Chebba et al. (2018) proposes attributed relations and modeling n-ary relations with blank nodes. However, these approaches unnecessarily increase the graph size, leading to extreme inefficiencies. Beyond the TKG structure, HTKGs (Ding et al., 2024) allow contextual attributes by adding qualifiers, disambiguating between otherwise identical TKG quadruples. N-TKGs (Hou et al., 2023) extend TKGs' quadruples into n-tuples, where relations connect n entities with assigned roles, allowing for n-ary facts. UniHR (Liu et al., 2024) introduces a hierarchical data representation module (HiDR) to jointly model heterogeneous information. However, these extensions fail to move beyond the two-primary-entity system.

TKG Forecasting Traditional methods for TKG forecasting include time-aware embedding models (Goel et al., 2020; Han et al., 2020), temporal random walk-based approaches (Sun et al., 2021; Liu et al., 2022), and graph-based approaches (Zhu et al., 2021; Li et al., 2021; Xu et al., 2023). More recently, a wave of LLM-based approaches has emerged, leveraging the pattern recognition capabilities of LLMs. Lee et al. (2023) studies LLMs' ability to perform TKG forecasting based on in-context learning. GenTKG (Liao et al., 2023) incorporates retrieval-augmented generation (RAG) into their method. zrLLM (Ding et al., 2023) relies on LLMs' semantic reasoning abilities to model unseen relations. Nonetheless, LLMs' reasoning capabilities on complex facts remain understudied.

Event Databases Geopolitical event databases are rich sources for building TKGs, offering chronologically ordered facts involving several actors in one or more actions (*e.g.,* wars, treaties, etc.). Previously, snapshots of GDELT (Leetaru and Schrodt, 2013) and ICEWS (Boschee et al., 2015), both part of broad initiatives to catalog global human behavior and predict conflicts, were used as TKG reasoning benchmarks. Other examples are ACLED (Raleigh et al., 2010; Semnani et al., 2025), containing location and battle event information related to eight conflict countries in West and Central Africa from 1960 to the present,

and UCDP (Sundberg et al., 2012; Sundberg and Melander, 2013), containing global data on non-state conflicts from 1989 to 2024. In this work, we utilize POLECAT (Halterman et al., 2023a,b), which stores cooperative and hostile geopolitical events from 2018 to 2024.

3 Preliminaries

3.1 Definitions

Temporal Knowledge Graph (TKG) Let $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{T}$ be a set of entities, relations, and timestamps, respectively. A TKG $\mathcal{G}$ comprises a set of quadruples representing temporal facts. Formally, we define $\mathcal{G}$ as

$$\mathcal{G} = \{(s, r, o, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T} \,. \quad (1)$$

Hyper-Relational Temporal Knowledge Graph (HTKG) Following the definition by Ding et al. (2024), we define an HTKG $\mathcal{H}$ as an attributed TKG where each quadruple is accompanied by a set of tuples containing additional information about the temporal fact. Formally, let $\mathcal{G}$ be a TKG containing the *primary quadruples*, we define $\mathcal{H}$ as

$$\mathcal{H} = \{(f, Q) \mid f \in \mathcal{G}, Q \subseteq \mathcal{R} \times \mathcal{E}\} \,. \quad (2)$$

Since HTKGs can also represent any TKG, we only utilize them from this point forward.

3.2 Queries

Link Prediction The most common query on HTKGs, which aims to find a missing entity to complete the partial fact at a particular time. Formally, a link prediction query is defined as

$$((s, r, ?, t), Q) \quad \text{or} \quad ((?, r, o, t), Q) \,. \quad (3)$$

Relation Prediction The second most common query on HTKGs, which aims to find the missing relation in the partial fact at a particular time. Formally, a link prediction query is defined as

$$((s, ?, o, t), Q) \,. \quad (4)$$

While other queries, such as qualifier or time prediction, are possible, since they are not as commonly used, we will not discuss them in this work.

3.3 Completion vs. Forecasting

Orthogonal to the above query distinctions, each query can be either a 1) completion or 2) forecasting query based on the extent of access to historical information. In a completion query, the predictive model has access to all temporal facts (*i.e.*, the whole HTKG), before, concurrent, or after the query's timestamp. However, in a forecasting query, the predictive model only has access to temporal facts before the query's timestamp. Generally, forecasting queries are far more helpful in real-world decision-making scenarios, and this temporal constraint on information makes them much more challenging. Throughout this work, we focus on the forecasting tasks. Note that, a predictive model $\mathcal{M}$ is a forecaster capable of running forecasting queries if and only if it satisfies two constraints: 1) if the model has a training stage, all training datapoints must have a timestamp strictly smaller than all test samples, and 2) during the testing stage, the model does not have access to datapoints with equal or greater timestamp than the test datapoint.

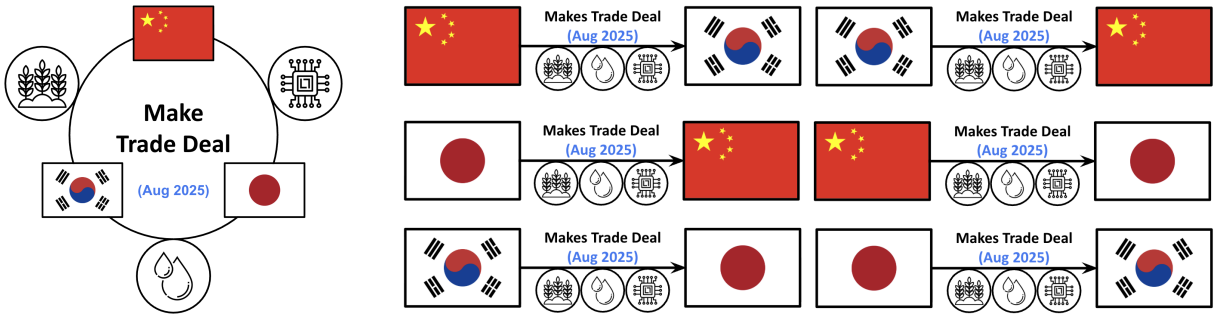
3.4 HTKG Limitations

While HTKG extends TKG by including attributes/qualifiers, it still lacks natural support for efficiently expressing complex temporal facts involving more than two primary entities. In this work, we highlight two types of such facts:

Group-Type Facts involving more than two primary entities related to the same relation, which conceptually form a graph clique. Typical examples of such facts include recurring political or financial summits that involve multiple nations attending simultaneously. Expressing such an event in HTKG requires decomposing a singular fact into multiple facts and, in some cases, introducing new entities (*i.e.,* reification). However, these transformations are inefficient in terms of redundancy, sparsity, and even inference, as the predictor must aggregate information across many facts rather than a single one. Figure 2 illustrates an example of an original fact concerning multiple entities along with the decomposed version adapted for HTKGs[1]. Moreover, if a new entity is not introduced to represent the event or the group of entities, which itself greatly amplifies the sparsity issues, HTKG struggles to distinguish between decomposed facts and separate co-occurring facts.

Set2Set-Type Facts that involve a group of entities (*i.e.,* actors) doing something to another group of entities (*i.e.,* recipients), which conceptually form a biclique. For example, in the geopolitical domain, numerous ongoing, emerging, and shift-

[1] China–Japan–South Korea Free Trade Agreement



(a) Original Fact (b) Decomposed Facts

Figure 2: **(a)** The original temporal fact representing a joint trade deal between {China, Japan, and South Korea} on {Cars, Chips, and Oil}. **(b)** The decomposed version of the original fact designed for HTKGs. Notably, the decomposed version requires a lot of redundancy to represent, and without an additional entity representing the group of countries, it is indistinguishable from three separate trade deals between these countries.

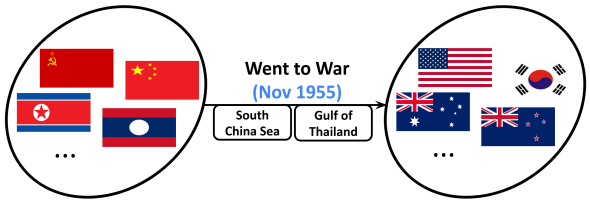


Figure 3: Vietnam War is an example of a complex geopolitical event that involved two coalitions of countries (*i.e.,* two sets of entities) engaging in a war (*i.e.,* an action) in locations such as the South China Sea and Gulf of Thailand (*i.e.,* a set of qualifiers).

ing coalitions form among different entities due to shared interests or other contemporary factors. These coalitions engage with each other in various settings, such as economic or ideological warfare, causing complex events that require proper temporal structures for accurate representation. Similar to the group-type, expressing such facts in HTKGs requires decomposing the original fact into multiple facts and/or introducing new entities, which leads to the same redundancy, sparsity, etc. issues. Figure 3 illustrates an example of this type of fact concerning two groups of entities[2]. Note that in this type of fact, at least one of the two groups of entities must be unnamed (*i.e.,* unlike NATO and the European Union); otherwise, we can replace both groups with named entities representing them, which is a supported type of fact in HTKG.

While it is possible to extend the temporal structure further to support more complex and higher-order facts, such as those involving multiple groups of entities within one relation, given their rare occurrence in POLECAT and generally in real-world scenarios, we leave them for future investigations.

[2]Vietnam War

4 Hyper-Relational Temporal Knowledge Generalized Hypergraph (HTKGH)

Hypergraph In contrast to regular graphs, edges can connect more than two vertices in hypergraphs, making them ideal to represent group-type facts. By simply replacing regular graphs in HTKGs with hypergraphs, we arrive at the hyper-relational temporal knowledge hypergraph (HTKH) structure. Formally, we define an HTKH $\mathcal{K}$ as

$$\mathcal{K} = \{(\Gamma, Q) \mid \Gamma \in \mathbb{P}^+(\mathcal{E}) \times \mathcal{R} \times \mathcal{T}, Q \subseteq \mathcal{R} \times \mathcal{E}\}$$

where $\mathbb{P}^+(\mathcal{E})$ is the non-null power set of entities. While HTKHs can properly represent group-type facts, they still struggle with expressing the set2set-type facts. Moreover, all edges in HTKHs are bidirectional, which is incompatible with existing unidirectional edges supported in HTKG.

Generalized Hypergraphs To address the shortcomings of HTKHs, we utilize a generalization of hypergraphs that allows second-order edges between first-order edges (*i.e.,* edges in HTKH), deriving the hyper-relational temporal knowledge generalized hypergraph (HTKGH) structure. By doing so, we reintroduce unidirectional edges to the structure and add support for set2set-type facts. Formally, we define an HTKGH $\mathcal{W}$ as

$$\begin{aligned}\mathcal{W} = \{(\Lambda, Q) \mid \Lambda \in \mathbb{P}^+(\mathcal{E}) \times \mathcal{R} \times \mathbb{P}(\mathcal{E}) \times \mathcal{T}, \\ |\Lambda_{\text{actors}}| + |\Lambda_{\text{recipients}}| > 1, \\ Q \subseteq \mathcal{R} \times \mathcal{E}\}\end{aligned}$$

where $\mathbb{P}(\mathcal{E})$ is the power set of entities. In this formalization, we assume a special type of null first-order edge, $e_\varnothing$, that allows us to connect entities without any pre-determined relationship. Moreover,

we denote the first non-empty group of entities as "*actors*" and the second group of entities as "*recipients*". Following these denotations, a group-type fact is expressed with an empty group of recipients, showcasing a relationship between the actors. Moreover, a set2set-type fact is expressed with a non-empty group of recipients.

Unidirectional vs. Bidirectional Edges While traditionally HTKG facts are unidirectional, allowing group-type facts has a side effect of introducing bidirectional edges. Specifically, since there is no inherent head or tail among the group of entities in these facts, the relation is semantically bidirectional, which enables us to express more diverse relations compactly within the same HTKGH structure. Formally, the unidirectional (r_u) vs. bidirectional (r_b) relations manifest as follows:

$$(\{v\}, r_u, \{u\}, t), Q) \text{ vs. } (\{v, u\}, r_b, \{\}, t), Q)$$

Note that these bidirectional relations only apply to first-order edges, as second-order edges remain unidirectional in our formalization.

5 Dataset

POLitical Event Classification, Attributes, and Types (POLECAT) (Halterman et al., 2023a) is a new global events database built to succeed the Integrated Conflict Early Warning System (ICEWS) (Boschee et al., 2015). POLECAT utilizes the Political Language Ontology for Verifiable Event Records (PLOVER) codebook for coding event data, a new event-mode-context ontology designed to be more general, easy to implement, and extendable, replacing the Conflict and Mediation Event Observations (CAMEO) framework (Gerner et al., 2002) utilized in ICEWS. Compared to CAMEO, PLOVER defines 18 event types, aggregating over many of the more than 250 CAMEO codes. To create `htkgh-polecat`, we utilize the POLECAT data from Jan 2018 to Jul 2024. Appendix C provides more details on dataset construction and statistics, such as entity and relation composition.

Data Filtering To enforce HTKGH's requirements and reduce the noise in the samples, we remove all facts that have zero actors or have one actor and no recipients. Doing so reduces the number of facts from 2.23M in the original database to approximately 556K. While this is a significant reduction, such filtering ensures dataset integrity and enhances information quality.

Anonymous Variation One of the challenges of evaluating LLMs is dealing with information leaks. Specifically, when the collected data is gathered from publicly accessible sources before the training cutoff date of an LLM, there is a chance that the model has seen that data during the training phase. Given the highly public nature of political events, this problem is exacerbated in this domain. As such, we create anonymous versions of `htkgh-polecat` by shuffling among entities and relations. These variations enable us to evaluate LLMs' pattern recognition capabilities accurately 1) without compromising information leaks or memorization from pre-training and 2) in counterintuitive scenarios (see Appendix E for more details).

6 Experiments

In these experiments, we focus on the relation prediction task, which is similar to the logical reasoning tasks, a desirable ability for LLMs.

6.1 Evaluation Setup

Test Set To test our models, we first filter out samples in 2018 to ensure the existence of sufficient context and then create a stratified dataset along temporal and relational axes, sampling a 1% test set (*i.e.,* approximately 5.5k facts).

Historical Context To build a historical context of size h for a given test fact f, we first use a combination of the following filters:

1. **Entity Filter:** Filter out facts that do not have any primary entity in common with the primary entities of f.
2. **Location Filter:** If the location is available in f, filter out facts that do not have the same location.
3. **Context Filter:** If there is any context available in f, filter out facts that do not share at least one context with f.

Then, we take the h most recent facts that have a timestamp strictly smaller than f. Appendix F provides the details of the prompt creation process.

Models We select 13 models from common publishers such as Google and Qwen, based on size and type, enabling further analysis on these critical factors (see Appendix G for more details). For size, our models range from 270M to 20B. For type, we focus on two common types: non-thinking (*i.e.,*

Variation	Filters			Heuristics			Non-thinking					Thinking			
	Entity	Location	Context	$\mathcal{F}$	$\mathcal{R}$	$\mathcal{C}$	L	Q4N	Q8N	G4	G12	D	Q4T	Q8T	O
Regular	True	True	True	49.6	34.5	48.7	42.3	**58.1**	**55.4**	49.0	**60.6**	25.5	**64.7**	**63.1**	**63.9**
	True	True	False	43.3	28.9	23.8	35.4	**44.4**	40.4	39.9	**45.9**	15.4	**50.0**	**48.3**	**47.2**
	True	False	True	45.8	28.5	32.9	40.1	**50.5**	**47.9**	43.4	**52.0**	18.8	**56.1**	**54.8**	**54.1**
	True	False	False	41.2	25.4	6.6	28.1	36.9	32.3	34.5	37.8	11.5	40.7	39.8	36.4
	False	True	True	38.5	21.4	22.8	34.8	**40.8**	36.7	35.6	**44.3**	12.4	**48.4**	**44.8**	**43.0**
	False	True	False	30.5	11.9	4.0	26.4	30.1	23.7	26.5	**32.2**	8.3	**35.5**	**31.4**	29.2
	False	False	True	35.6	14.8	4.7	25.8	30.2	27.0	25.5	32.8	7.9	**37.7**	34.3	30.3
	False	False	False	27.3	5.0	0.0	25.5	**28.4**	21.3	22.0	**29.4**	6.6	**34.2**	**28.8**	27.1
Shuffled (Entities)	True	True	True	-	-	-	-0.1	+1.0	+0.3	-0.7	+0.2	+1.0	-0.6	-0.2	-0.1
	True	True	False	-	-	-	-0.1	0.0	-0.7	-0.6	+1.7	+1.4	-0.7	+0.4	0.0
	True	False	True	-	-	-	+0.4	0.0	+0.3	+0.2	+0.2	+0.8	-0.5	-0.3	-0.1
	True	False	False	-	-	-	+0.7	-0.2	+0.1	-0.9	+1.2	+1.4	-0.7	-1.0	-0.5
	False	True	True	-	-	-	-0.1	+2.8	+1.5	+2.1	+1.0	+0.7	+0.6	+0.3	-0.1
	False	True	False	-	-	-	+1.1	+1.9	+0.2	+3.6	+0.9	+0.7	-1.4	+0.7	+0.4
	False	False	True	-	-	-	0.0	+2.7	+1.2	+1.4	+2.3	-0.1	+0.2	-1.0	-0.5
	False	False	False	-	-	-	+2.8	+2.3	-0.7	+5.3	+1.7	+0.7	-1.9	-1.7	-0.8
Shuffled (All)	True	True	True	-	-	-	+3.1	+2.4	+4.2	-8.4	+2.2	+9.6	-1.1	+1.3	+0.1
	True	True	False	-	-	-	+2.4	+0.5	+4.3	-10.5	+2.4	+6.8	-2.7	0.0	-0.1
	True	False	True	-	-	-	+1.9	+1.9	+4.2	-7.6	+2.3	+9.5	-0.3	+1.3	0.0
	True	False	False	-	-	-	+1.8	+0.4	+5.6	-11.3	+1.6	+5.8	-3.5	-0.5	-1.7
	False	True	True	-	-	-	-0.7	+0.3	+4.8	-5.4	+1.2	+7.8	-2.4	+1.4	+0.2
	False	True	False	-	-	-	-1.0	-4.0	+7.0	-6.8	-0.2	+4.6	-5.3	-0.5	-1.6
	False	False	True	-	-	-	-4.0	-0.6	+4.2	-4.4	+0.7	+5.3	-4.0	+1.1	-0.1
	False	False	False	-	-	-	-6.4	-9.5	+6.0	-6.6	-4.0	+3.2	-9.3	-1.3	-3.2

Table 1: Relation prediction accuracy (%) on `htkgh-polecat` variations. For each test sample, we include the most recent 100 facts (after filtering) as contextual information. In regular variation, bold values beat all the respective heuristics. In both shuffled variations, the numbers are reported as performance differences compared to their regular counterparts. **Legend:** $\mathcal{F}$ → `Frequency`, $\mathcal{R}$ → `Recency`, $\mathcal{C}$ → `Copy`, **L** → `Llama-3.1-8B-Instruct`, **Q4N** → `Qwen3-4B-Instruct-2507`, **Q8N** → `Qwen3-8B (non-thinking)`, **G4** → `gemma-3-4b-it`, **G12** → `gemma-3-12b-it`, **D** → `DeepSeek-R1-Distill-Qwen-7B`, **Q4T** → `Qwen3-4B-Thinking-2507`, **Q8T** → `Qwen3-8B (thinking)`, and **O** → `openai/gpt-oss-20b (medium reasoning effort)`.

instruction-tuned) and thinking. Finally, to better analyze their performance, we allow thinking models to generate up to 16384 tokens while non-thinking models are limited to 14 tokens (*i.e.,* max length of the relations, nine, plus five), all using their preferred decoding strategy and parameters.

6.2 Experimental Results

Q1. How good are LLMs at relation prediction? To better understand LLMs' performance, we compare them against three baselines that use simple heuristics that have been shown to either impose strong bias in LLMs (Lee et al., 2023) or have strong performance in TKG forecasting benchmarks (Gastinger et al., 2024). More specifically, given the historical context $\mathcal{H}$, we calculate 1) *Frequency* as the relation with the most occurrences, 2) *Recency* as the relation that comes last, and 3) *Copy* as the last relation that appears with the same elements as the query. Table 1 (Regular variation) presents our experimental results on nine different state-of-the-art non-thinking and thinking LLMs. Looking at the non-thinking category, only `gemma-3-12b-it` and `Qwen3-4B-Instruct-2507` relatively consistently beat the heuristics, showcasing critical improvements with size increases (`gemma-3-4b-it` vs. `gemma-3-12b-it`) and alignment improvements[3] (`Qwen3-8B (non-thinking)` vs. `Qwen3-4B-Instruct-2507`). Moreover, we see that with more rigorous filtering, which leads to more relevant facts being present in the history, LLMs tend to improve faster than the heuristics (*i.e.,* increasing their lead margin or closing the performance gap), showcasing the potential gains of improving history retrievers, similar to the findings of Xia et al. (2024). In the thinking category, we observe more cases where the models beat the heuristics, along with similar observations on the filtering; however, we notice two surprising results: 1) `DeepSeek-R1-Distill-Qwen-7B` does not perform well at all, which could par-

[3]Based on the release card, it seems that `Qwen3-*-2507` models have gone through an improved post-training phase (vs. `Qwen3-*` models) rather than a full pre-training phase.

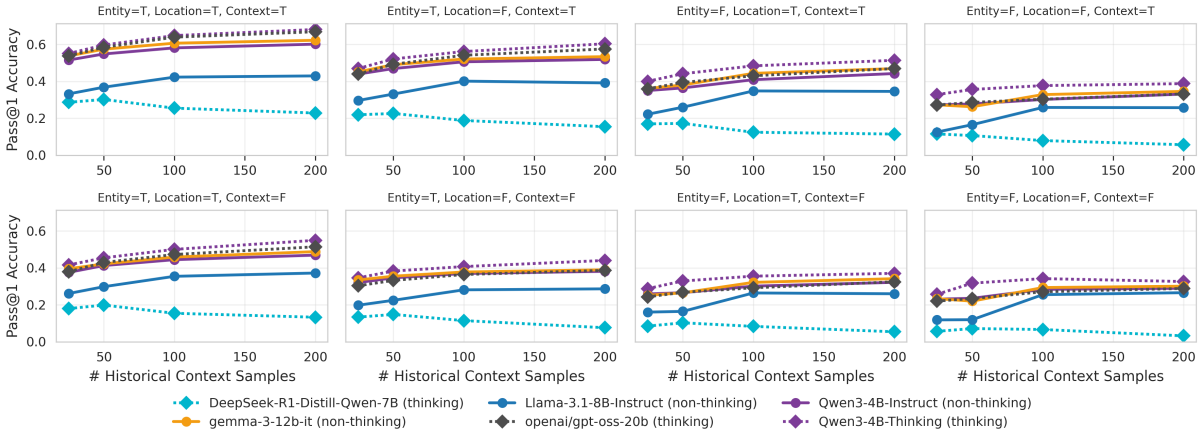
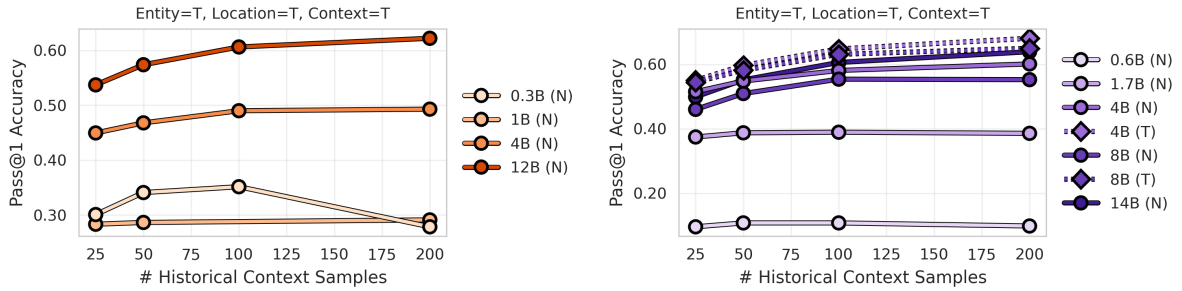


Figure 4: Relation prediction accuracy (%) on `htkgh-polecat` context variations over the number of retrieved contextual samples. From each model family, we only display the best-performing member.



(a) Gemma-3

(b) Qwen3

Figure 5: Relation prediction accuracy (%) on `htkgh-polecat` for different model sizes in the **(a)** Gemma-3 and **(b)** Qwen3 families. **Legend:** N → non-thinking, T → thinking.

tially be explained by misformatting rates (see Appendix I), and 2) `Qwen-3` models beat the significantly larger `gpt-oss` model by substantial margins in many cases, showcasing the power of smaller well-trained models. Moreover, we note a massive jump of **+6.0-7.6%** from non-thinking to thinking variation of the `Qwen3` models, which highlights the effectiveness of test-time scaling for the relation prediction task on TKGs.

Q2. How much do LLMs rely on memorization for their predictions? While the observed improvements using test-time scaling are a sign of LLMs' going beyond memorization, we further experiment with two shuffled variations of our dataset, where 1) only countries are shuffled, and 2) countries and relations are shuffled. Table 1 (Shuffled variations) presents our experimental results for this experiment. In the entity-only variation, we see that generally non-thinking models' accuracy slightly increases (up to **+5.3%**) while thinking models' accuracy slightly decreases (up to **-1.9%**), which showcases their resilience and context-dependence to a great degree, similar to the findings of Lee et al. (2023). However, when we also shuffle the relations, the behavior of many of the models becomes erratic, with some gaining up to **+9.6%** accuracy and some losing up to **-11.3%** accuracy. This phenomenon highlights the importance of testing LLMs under extreme settings where the provided information goes against the model's beliefs to different degrees.

Q3. What is the effect of the number of contextual samples? One of the critical considerations of using LLMs for TKG-related tasks is the number of facts that could be included in the context. Figure 4 illustrates our experimental results with respect to the number of contextual samples. As is evident, most models benefit from the inclusion of more context, `DeepSeek-R1-Distill-Qwen-7B` being the only model that degrades with more samples. Moreover, we observe that scenarios with stricter filtering benefit more from an increase in samples. These results illustrate a promising trend in context scalings for more complex TKG-related

Variation	Filters			Heuristics			GNN				LLM								
	Entity	Location	Context	$\mathcal{F}$	$\mathcal{R}$	$\mathcal{C}$	$\mathbf{B}_A$	$\mathbf{B}_M$	$\mathbf{Hy}_A$	$\mathbf{Hy}_M$	L	Q4N	Q8N	G4	G12	D	Q4T	Q8T	O
Regular	True	True	True	51.7	39.1	56.0	48.2	49.1	48.8	49.1	48.9	**63.3**	**59.4**	52.0	**65.5**	20.6	**70.2**	**65.8**	**69.3**
	True	True	False	47.0	33.9	33.3	**48.7**	**48.9**	**49.4**	**49.0**	41.9	**50.5**	46.2	44.2	**51.5**	13.8	**57.1**	**53.8**	**55.5**
	True	False	True	49.1	31.2	41.2	48.5	48.6	48.2	48.7	46.8	**56.8**	**54.2**	47.7	**58.4**	18.1	**62.3**	**60.0**	**61.2**
	True	False	False	44.9	29.0	12.8	**48.2**	**49.4**	**48.2**	**48.2**	33.6	43.5	39.7	39.3	43.7	11.8	**47.7**	**45.3**	43.7
	False	True	True	43.0	25.5	28.5	**48.2**	**49.1**	**49.3**	**48.7**	39.6	**46.4**	41.2	40.5	**49.0**	12.0	**53.0**	**48.4**	**49.5**
	False	True	False	33.7	14.7	8.6	**49.1**	**48.4**	**48.9**	**48.8**	30.9	**35.8**	27.6	30.5	**37.0**	7.3	**40.4**	**36.0**	**35.1**
	False	False	True	39.1	17.3	7.3	**49.1**	**49.2**	**49.5**	**49.1**	29.4	34.5	30.7	30.1	37.6	7.3	**41.2**	37.7	33.9
	False	False	False	26.4	4.3	4.8	**49.8**	**49.4**	**49.2**	**49.7**	**28.2**	**32.3**	23.8	23.8	**31.9**	4.9	**37.9**	**30.5**	**30.0**

Table 2: Relation prediction accuracy (%) on the last 24 months of the `htkgh-polecat` test set. For each test sample, an LLM predictor is provided with the most recent 100 facts after filtering, whereas a GNN predictor is provided with the last 4 windows. Bold values beat all the respective heuristics. **Legend:** $\mathcal{F} \rightarrow$ `Frequency`, $\mathcal{R} \rightarrow$ `Recency`, $\mathcal{C} \rightarrow$ `Copy`, $\mathbf{B}_A \rightarrow$ `Bagging aggregation with mean pooling`, $\mathbf{B}_M \rightarrow$ `Bagging aggregation with max pooling`, $\mathbf{Hy}_A \rightarrow$ `Hypergraph aggregation with mean pooling`, $\mathbf{Hy}_M \rightarrow$ `Hypergraph aggregation with max pooling`, **L** → `Llama-3.1-8B-Instruct`, **Q4N** → `Qwen3-4B-Instruct-2507`, **Q8N** → `Qwen3-8B (non-thinking)`, **G4** → `gemma-3-4b-it`, **G12** → `gemma-3-12b-it`, **D** → `DeepSeek-R1-Distill-Qwen-7B`, **Q4T** → `Qwen3-4B-Thinking-2507`, **Q8T** → `Qwen3-8B (thinking)`, and **O** → `openai/gpt-oss-20b (medium reasoning effort)`.

predictions. Note that some models allow us to scale the samples further, but due to resource constraints, we leave these experiments to future work.

Q4. What is the effect of the model size? While many training factors are often undisclosed, model size has always been one of the consistent predictors of performance. Figure 5 showcases a comparison with regard to model size within the Gemma-3 and Qwen3 families (see Appendix J for more filtering variations). In both families, the performance generally increases as model sizes increase, especially past the 1B size threshold. The most significant exceptions are the `Qwen3-4B-*-2507` models, which even beat the much larger `Qwen3-14B` model, underscoring the critical nature of other factors that must be studied for model selection.

Q5. How good are LLMs compared to supervised graph-based models? We experiment with two graph neural network (GNN) models, both based on first encoding fact-level information, then aggregating facts within a time window with either a permutation invariant *bagging* or a more expressive *hypergraph* approach, and finally using a transformer-based encoder to capture temporal trends (see Appendix K for more granular details on the GNN-based models). Since these supervised models need a training phase, we split the dataset into a train (*i.e.,* 2018-2022) and a test (*i.e.,* 2023-2024) set. Then, we randomly sample and hold out 10% of the training data as a validation set. Finally, we calculate the performance of the model on the validation set after each epoch and select the version with the best validation result for testing.

Table 2 presents our experimental results comparing GNN-based models with our baselines and selected LLMs. As is evident, in scenarios where we have weak filtering over contextual information, GNNs consistently beat LLMs, which we can potentially attribute to the information stored in embeddings and learned weights. However, as we tighten our filtering, LLMs start to gain momentum, beating GNN-based models, whose performance stays flat, by as much as **21%**. These results emphasize the importance of retrieving high-quality context for predictions again. Regarding our GNN-based models, we hypothesize that the current window encoding formulation introduces an information bottleneck by collapsing all fact representations, which leads to their flat performance across different filtered settings. Given the numerous different modeling choices and designs for GNNs, we leave further investigations to future work.

7 Conclusion

In this work, we presented HTKGH, an extension of HTKG that is designed to efficiently and accurately express complex facts involving many primary entities. Then, using the HTKGH's formalization, we introduced the `htkgh-polecast` dataset, built on top of the POLECAT event database. Moreover, we conducted a thorough investigation into using state-of-the-art LLMs on a relation prediction task defined on `htkgh-polecast`. Our results showcased the out-of-the-box flexibility of LLMs to adapt to reasoning over complex higher-order facts, paving the way for future applications on even more complex structures and underlying data.

Limitations

Domain Throughout this work, we focused on the geopolitical events due to their popularity; however, there are a plethora of other domains that could use such formalization using HTKGH, which we hope to see in future work.

Graph Neural Networks In this work, we focused on LLMs due to their flexibility and ease of use out of the box, but it is possible to investigate various existing and novel architectures based on the GNNs' framework. While Section 6 and Appendix K present a small study on a few GNN-based models, an in-depth and comprehensive experimentation with these models is warranted, which we leave to future work, as it is outside the scope of this work.

Knowledge Representation One of the often overlooked criteria for graph structures is the knowledge representation perspective, which concerns itself with querying time and complexity using technologies such as SPARQL. Ideally, we want structures that both benefit learning algorithms and querying technology, demanding studies similar to the work by Iglesias-Molina et al. (2023) for static knowledge graphs. Given the limited scope of this work, we leave such investigations to future work.

References

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Asmaa Chebba, Thouraya Bouabana-Tebibel, and Stuart H Rubin. 2018. Attributed and n-ary relations in owl for knowledge modeling. *Computer Languages, Systems & Structures*, 54:183–198.

Zifeng Ding, Heling Cai, Jingpei Wu, Yunpu Ma, Ruotong Liao, Bo Xiong, and Volker Tresp. 2023. zrllm: Zero-shot relational learning on temporal knowledge graphs with large language models. *arXiv preprint arXiv:2311.10112*.

Zifeng Ding, Jingcheng Wu, Jingpei Wu, Yan Xia, Bo Xiong, and Volker Tresp. 2024. Temporal fact reasoning over hyper-relational knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 355–373, Miami, Florida, USA. Association for Computational Linguistics.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium. Association for Computational Linguistics.

Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, Reihaneh Rabbany, and Guillaume Rabusseau. 2025. Tgb 2.0: a benchmark for learning on temporal knowledge graphs and heterogeneous graphs. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Julia Gastinger, Christian Meilicke, Federico Errica, Timo Sztyler, Anett Schuelke, and Heiner Stuckenschmidt. 2024. History repeats itself: a baseline for temporal knowledge graph forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Deborah J Gerner, Philip A Schrodt, Omür Yilmaz, and Rajaa Abu-Jabr. 2002. Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions. *International Studies Association, New Orleans*, 35.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3988–3995.

Andrew Halterman, Benjamin E Bagozzi, Andreas Beger, Phil Schrodt, and Grace Scraborough. 2023a. Plover and polecat: A new political event ontology and dataset. In *International Studies Association Conference Paper*.

Andrew Halterman, Philip A Schrodt, Andreas Beger, Benjamin E Bagozzi, and Grace I Scarborough. 2023b. Creating custom event data without dictionaries: A bag-of-tricks. *arXiv preprint arXiv:2304.01331*.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020. xerte: Explainable reasoning on temporal knowledge graphs for forecasting future links. *arXiv preprint arXiv:2012.15537*.

Zhongni Hou, Xiaolong Jin, Zixuan Li, Long Bai, Saiping Guan, Yutao Zeng, Jiafeng Guo, and Xueqi Cheng. 2023. Temporal knowledge graph reasoning based on n-tuple modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1090–1100, Singapore. Association for Computational Linguistics.

Ana Iglesias-Molina, Kian Ahrabian, Filip Ilievski, Jay Pujara, and Oscar Corcho. 2023. Comparison of knowledge graph representations for consumer scenarios. In *The Semantic Web – ISWC 2023*, pages 271–289, Cham. Springer Nature Switzerland.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6669–6683, Online. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. 2023. Temporal knowledge graph forecasting without knowledge using in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 544–557, Singapore. Association for Computational Linguistics.

Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. 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.

Ruotong Liao, Xu Jia, Yangzhe Li, Yunpu Ma, and Volker Tresp. 2023. Gentkg: Generative forecasting on temporal knowledge graph with large language models. *arXiv preprint arXiv:2310.07793*.

Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 4120–4127.

Zhiqiang Liu, Yin Hua, Mingyang Chen, Yichi Zhang, Zhuo Chen, Lei Liang, Huajun Chen, and Wen Zhang. 2024. Unihr: Hierarchical representation learning for unified knowledge graph link prediction. *arXiv preprint arXiv:2411.07019*.

Xiaohua Lu, Liubov Tupikina, and Mehwish Alam. 2025. Two-dimensional taxonomy for n-ary knowledge representation learning methods. *arXiv preprint arXiv:2506.05626*.

Natasha Noy, Alan Rector, Pat Hayes, and Chris Welty. 2006. Defining n-ary relations on the semantic web. *W3C working group note*, 12(4).

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Clionadh Raleigh, Rew Linke, Håvard Hegre, and Joakim Karlsen. 2010. Introducing acled: An armed conflict location and event dataset. *Journal of peace research*, 47(5):651–660.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Sina Semnani, Pingyue Zhang, Wanyue Zhai, Haozhuo Li, Ryan Beauchamp, Trey Billing, Katayoun Kishi, Manling Li, and Monica Lam. 2025. LEMONADE: A large multilingual expert-annotated abstractive event dataset for the real world. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25813–25852, Vienna, Austria. Association for Computational Linguistics.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. *arXiv preprint arXiv:2109.04101*.

Ralph Sundberg, Kristine Eck, and Joakim Kreutz. 2012. Introducing the ucdp non-state conflict dataset. *Journal of peace research*, 49(2):351–362.

Ralph Sundberg and Erik Melander. 2013. Introducing the ucdp georeferenced event dataset. *Journal of peace research*, 50(4):523–532.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yuwei Xia, Ding Wang, Qiang Liu, Liang Wang, Shu Wu, and Xiao-Yu Zhang. 2024. Chain-of-history reasoning for temporal knowledge graph forecasting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16144–16159, Bangkok, Thailand. Association for Computational Linguistics.

Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal knowledge graph reasoning with historical contrastive learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 4765–4773.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. *Advances in neural information processing systems*, 30.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4732–4740.

A Examples

Link Prediction Example Using a link prediction query, we can answer questions such as " *Who* is going to *win* the *Super Bowl* in *2026* at the *Levi's stadium*? " which is represented as

$$((?, \text{win}, \text{Super Bowl}, 2026), \{\text{stadium: Levi's}\})\ .$$

Relation Prediction Example Using a relation prediction query, we can answer questions such as " *What* is going to happen between *Russia* and *Ukraine* in *2026* at *Donbas*? " which is represented as

$$((\text{Russia}, ?, \text{Ukraine}, 2026), \{\text{location: Donbas}\})\ .$$

Hyper-Relational Temporal Knowledge Hypergraph Example Using an HTKH, we can express facts such as " *US*, *Canada*, and *Mexico* will *negotiate* a new *trade agreement* in *2026*. " in the form of

$$((\{\text{US}, \text{Canada}, \text{Mexico}\}, \text{negotiate}, 2026), \{\text{type: trade agreement}\})\ .$$

Hyper-Relational Temporal Knowledge Generalized Hypergraph Example Using an HTKGH, we can efficiently express facts such as " *US* and *UK* *sanction* *Russia* and *Belarus* in *2022* over *Russo-Ukrainian war*. " in the form of

$$((\{\text{US}, \text{UK}\}, \text{sanction}, \{\text{Russia}, \text{Belarus}\}, 2022), \{\text{cause: Russo-Ukrainian war}\})\ .$$

B Backward Compatibility

It is trivial to show that any HTKG can be converted into an HTKGH, as we can replace each entity in the primary quadruple with a set containing only that entity. Concretely, this entails the following conversion:

$$((s, r, o, t), Q) \rightarrow (\{s\}, r, \{o\}, t), Q)$$

Consequently, any model that can process HTKGHs will also be able to process HTKGs, which is crucial for maintaining backward compatibility with existing datasets.

C Dataset

C.1 Dataset Construction

Entity Construction To construct entities with more interactions (*i.e.,* dense), we use countries instead of individuals (*e.g.,* Vladimir Putin) as our entities. Moreover, to add more specificity and resolution to the entities, whenever available, we include the sector information (*e.g.,* judicial, government, or civilians). This design choice creates multiple entities per country, each representing a slightly more granular actor with varying duties or interests. For example, we do not include entities such as "Mark Carney" but instead use "Canada (GOV)" as the entity. Ultimately, we end up with *5268 entities*, built on top of 199 countries.

Relation Construction To construct the relations, whenever available, we combine the "event type" and " event mode" fields to provide more granularity regarding the actions in the facts. One of the examples of such a merger is "retreat (ceasefire)" where the event mode (*i.e.,* ceasefire) provides more resolution to the event type (*i.e.,* retreat). This results in *42 relations* built from the original 18 event types in the PLOVER ontology.

Qualifier Construction We use two fields in the original coded events to construct the qualifiers: country and contexts. When available, the former denotes the country where the event occurred. The latter, when available, denotes the context of the event from 37 categories, such as military or legislative. Since many events have multiple contexts, we add one qualifier for each. In the end, each fact has, on average, *1.37 qualifiers*.

C.2 Dataset Statistics

Figure 6 illustrates various statistics on the `htkgh-polecat` dataset, from the top frequent entities and relations to the distribution of the number of entities. Moreover, Figure 7 presents the frequency of different edge types in the `htkgh-polecat` dataset. As is evident, the two

Publisher	Model	#Params (B)	Type
Google	gemma-3-270m-it	0.3	N
	gemma-3-1b-it	1.0	N
	gemma-3-4b-it	4.0	N
	gemma-3-12b-it	12.0	N
Qwen	Qwen3-0.6B	0.8	H
	Qwen3-1.7B	2.0	H
	Qwen3-4B-Instruct-2507	4.0	N
	Qwen3-4B-Thinking-2507	4.0	N
	Qwen3-8B	8.0	H
	Qwen3-14B	15.0	H
Meta	Llama-3.1-8B-Instruct	8.0	N
DeepSeek	DeepSeek-R1-Distill-Qwen-7B	8.0	T
OpenAI	gpt-oss-20b	22.0	T

Table 3: We curate an inclusive list of thinking and non-thinking models from prominent model providers. **Legend:** **N** → Non-thinking, **T** → Thinking, and **H** → Hybrid.

types of facts discussed in Section 3.4 encompass many facts in real-world political events. Finally, Figure 8 showcases the distribution of events across different years, where we mostly see a uniform distribution across the years, except for 2024, where there is about a 46% drop in the frequency of events due to a data cutoff in July.

C.3 Test Set Construction

Figure 9 presents descriptive statistics over the constructed test set. As is evident, the most frequent entities are similar to those for the dataset overall (see Figure 6) with minor shuffling of the order outside the top-3. Moreover, the distribution of relations, which were stratified upon, is unchanged from the overall dataset. Similarly, the distribution over years is relatively unchanged, the only difference being that we drop 2018 from the test set to ensure the existence of a historical context for all the facts. We do see fewer events with high numbers of actors or recipients, but they have not been eliminated from the test set. Finally, as shown in Figure 7, 23.5% of the edges in the test set are of the newly included edge types, compared to 23.6% in the overall dataset.

D Comparison to `tkgl-polecat`

Compared to `tkgl-polecat`, we 1) preserve all the primary entities, instead of choosing one on each side to force a TKG format, 2) include location and context qualifiers as extra information, 3) encompass a more extended period of time (2018-2023 vs. 2018-2024), 4) use a scheme to construct entities and relations, leading to a higher density in the graph, and 5) employ rigorous filtering to ensure quality of all the facts in the dataset.

E Construction of `htkgh-polecat-anon`

Accounting for information leaks from the pre-training phase, we create anonymized versions of `tkgl-polecat` to investigate memorization versus pattern recognition abilities properly. To this end, we shuffle the entities and/or relations, practically transferring the facts to an imaginary world that the model has not seen before and is counterintuitive to its existing beliefs. During anonymization, we ensure consistency by 1) prohibiting multiple original words from being shuffled with the same new word and 2) synchronizing the countries in the location qualifiers and the primary entities while shuffling. Moreover, we consider the following three fields as potentially confounding information for anonymization: entities (actors and recipients), relations, and location qualifiers. Finally, we create two variations based on the anonymized fields: 1) entities and country qualifiers only, 2) entities, country qualifiers, and relations.

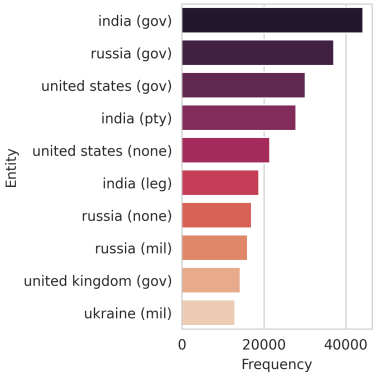
F Prompt Templates

F.1 Non-thinking

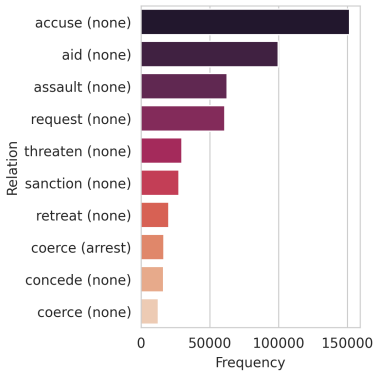
```
You are given a series of related geopolitical
events, each described by its "actors",
"recipients", "relation", and "qualifiers".
Analyze the historical events and the new event
carefully, considering the context provided by
the historical events.
Based on your analysis, determine the most
likely "relation" from a list of candidates for
the new event.
For your final answer, only output the chosen
candidate in the same exact format as the
candidates without any additional text.

Output format example:
protest (strike)

Here are the historical events to analyze:
{context_samples}

Here is the new event to classify:
{fact}

Here are the candidate relations:
{candidates}

The most likely relation is:
```

F.2 Thinking

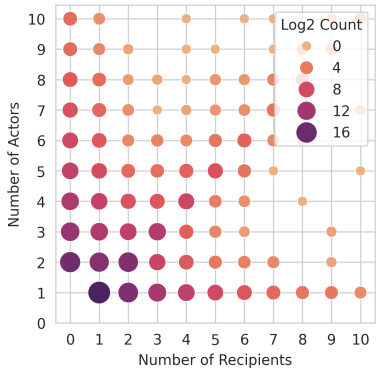
```
You are given a series of related geopolitical
events, each described by its "actors",
"recipients", "relation", and "qualifiers".
```



(a) Top 10 Frequent Entities



(b) Top 10 Frequent Relations



(c) Number of Entities Frequency (≤ 10)

Figure 6: Statistics of the `htkgh-polecat` dataset: **(a)** The top 10 frequent entities. Each entity is comprised of a country and a sector within that country. **(b)** The top 10 frequent relations. Each relation is comprised of a type and a mode for that type. **(c)** The frequency of the number of entities (*i.e.,* actors and recipients), up to 10 per group.

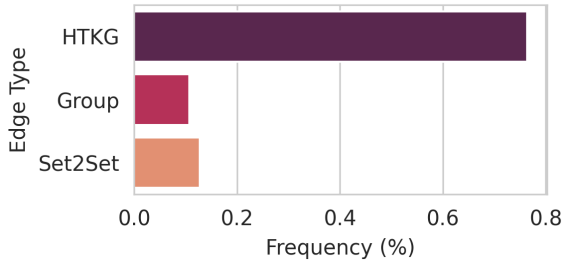


Figure 7: The frequency of different edge types in the `htkgh-polecat` dataset. The newly included edge types (*i.e.,* Group and Set2Set) compose roughly 23.6% of all edges, showcasing their prevalence in real-world data.

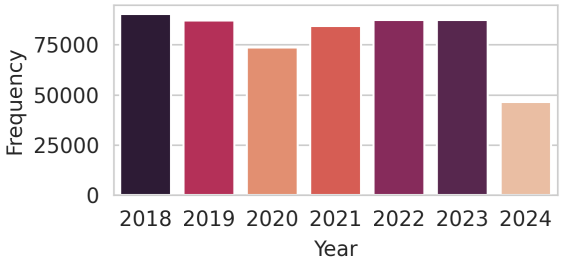


Figure 8: The frequency of events at different years in the `htkgh-polecat` dataset. The events are mostly distributed uniformly across years, except for 2024, where there is a roughly 46% drop, due to a July 22 cutoff.

```
Consider the new event and the provided context
carefully to determine the most likely
"relation" from a list of candidates for the
new event.
For your final answer, only output the chosen
candidate in the same exact format as the
candidates without any additional text.

Here are the historical events to analyze:
{context_samples}

Here is the new event to classify:
{fact}

Here are the candidate relations:
{candidates}
```

G Models

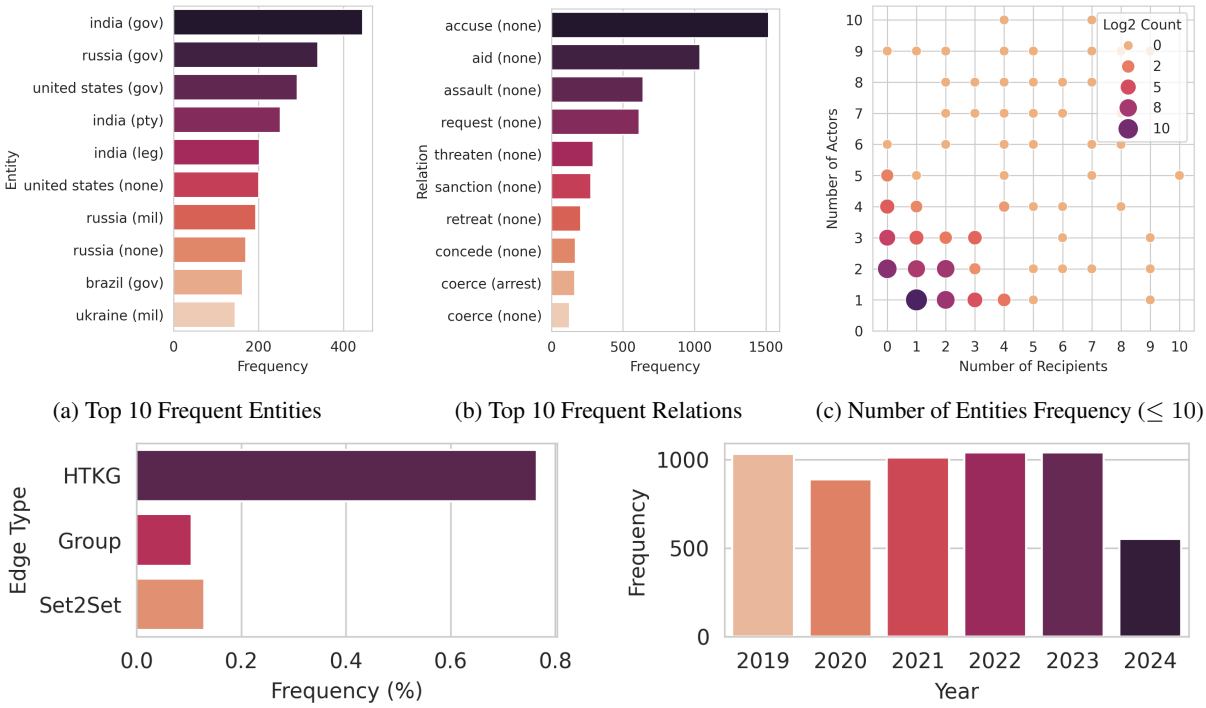
Table 3 showcases all 13 models used in our experiments.

H Implementation Details

We implemented our codebase using the vLLM (Kwon et al., 2023) and Transformers (Wolf et al., 2020) libraries. All our experiments are run on a server with 8×A6000 48GB GPUs and two servers with 8×A5000 24GB GPUs.

I Response Parser

To include as many responses as possible, we use a two-tier extractor that searches for matches on both the ground truth's raw text and index. To this end, we extract all the possible answers that are in the correct format from the response, taking the first one as the final answer and comparing it to the ground truth's raw text. If there are no matches, we extract all the numbers from the response, taking the first one as the final answer and comparing it to the ground truth's index. Figure 10 shows the ratio of misformatted outputs that were not parsed by our method. We can see that most models consistently exhibit a low rate of misformatted outputs, except for `Llama-3.1-8B-Instruct` and `DeepSeek-R1-Distill-Qwen-7B`. Specifically, `DeepSeek-R1-Distill-Qwen-7B` increas-



(a) Top 10 Frequent Entities

(b) Top 10 Frequent Relations

(c) Number of Entities Frequency (≤ 10)

(d) Edge Types

(e) Sample by Year

Figure 9: Statistics of the `htkgh-polecat` test set: **(a)** The top 10 frequent entities. **(b)** The top 10 frequent relations. **(c)** The frequency of the number of entities (*i.e.,* actors and recipients), up to 10 per group. **(d)** Edge type distribution. **(e)** Sample year distribution. Distributions are similar to those of the overall `htkgh-polecat` dataset, apart from having fewer facts with high numbers of actors or recipients, and the dropping of events from 2018 due to a lack of historical context in the dataset.

ingly struggles as we increase the number of contextual samples, while `Llama-3.1-8B-Instruct` specifically struggles when all filters are turned on.

J Effect of Model Size

Figure 11 and Figure 12 present the full results of our experiments on the effect of model size on relation prediction accuracy for the Gemma-3 and Qwen3 families. In both families, we observe that performance generally increases as models get larger, with some minor exceptions in the smaller $<$ 1B models. Notably, the exceptions happen in the scenarios with fewer contextual samples and looser filters, which showcases a potential lack of adequate information for reasoning, leading to a virtual upper bound in performance. Moreover, in the Qwen-3 family, we observe that the thinking variants outperform non-thinking ones consistently, with the 4B models (*i.e.,* `Qwen3-4B-*-2507`) outperforming the rest of the family.

K Graph Neural Network Models

In this section, we study GNN-based models.

K.1 Preliminaries

Let a fact be defined as

$$\psi = (A_\psi, r_\psi, R_\psi, t_\psi, Q_\psi), \tag{5}$$

consisting of the actors set, relation label, recipients set, timestamp, and qualifier key-value pairs, respectively. For a query fact q at time t_q, we form a query-specific window history sequence with fixed lookback duration Δ and history length $\mathcal{H}$ as

$$\mathcal{S}(q) = \{\mathcal{W}_0(q), \mathcal{W}_1(q), ..., \mathcal{W}_\mathcal{H}(q)\}\,. \tag{6}$$

For each window index $j \in \{1, ..., \mathcal{H}\}$, facts in that window are defined as

$$\begin{aligned}\Psi_j(q) = \{ \\ \psi : t_q - j\Delta \leq t_\psi < t_q - (j-1)\Delta \wedge \\ \kappa(\psi, q) = 1 \\ \},\end{aligned} \tag{7}$$

where $\mathcal{W}_j(q)$ is the window structure built from $\Psi_j(q)$, and κ is an indicator function to filter contextual facts based on zero or more criteria (*i.e.,* entities, locations, or context). The query window

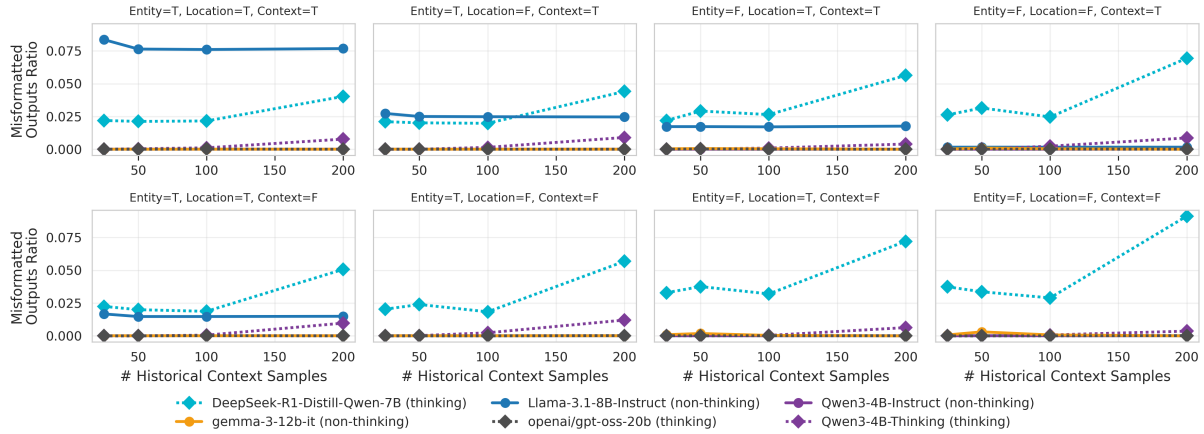


Figure 10: Proportion of misformatted outputs on `htkgh-polecat` variations. From each model family, we only display the best-performing member.

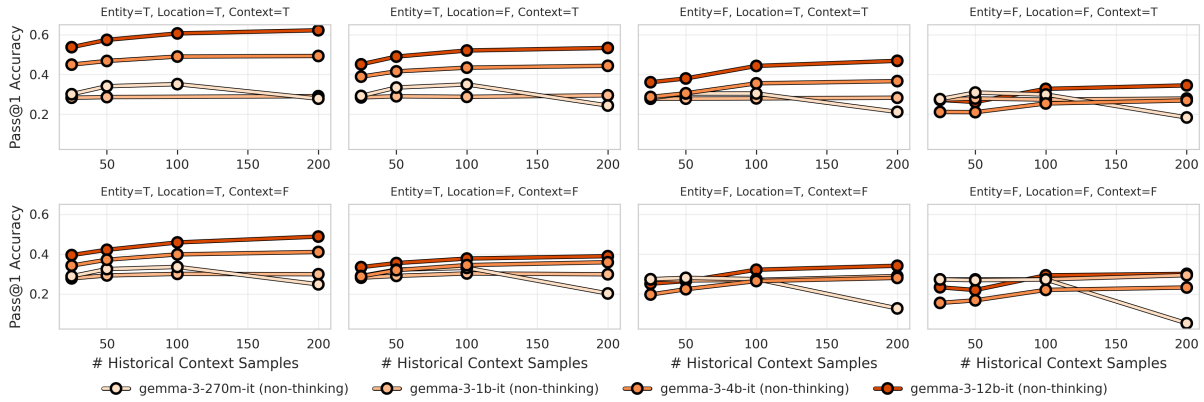


Figure 11: Relation prediction accuracy (%) on `htkgh-polecat` variations for the Gemma-3 family.

$\mathcal{W}_0(q)$ is constructed from only actors, recipients, and qualifiers of the query, and as such, it does not include relation information.

K.2 Fact Representation

For a given fact $\psi \in \Psi_i(q)$, we compute three role-specific DeepSet (Zaheer et al., 2017) representations for actors, recipients, and qualifiers as

$$\begin{aligned}
\mathbf{h}_\psi^A &= g_A\big(\text{Pool}(\{f_A(E_\mathcal{E}[e]) : e \in A_\psi\})\big),\\
\mathbf{h}_\psi^R &= g_R\big(\text{Pool}(\{f_R(E_\mathcal{E}[e]) : e \in R_\psi\})\big),\\
\mathbf{h}_\psi^Q &= g_Q\big(\text{Pool}\big(\{f_Q([E_\mathcal{R}[r]; E_\mathcal{E}[e]]) : r, e \in Q_\psi\}\big)\big),
\end{aligned} \tag{8}$$

where *Pool* is a permutation invariant pooling function, g_* and f_* are MLPs with ReLU activations, and $E_\mathcal{E} \in \mathbb{R}^{|\mathcal{E}|\times d}$ and $E_\mathcal{R} \in \mathbb{R}^{|\mathcal{R}|\times d}$ are entity and relation embedding lookup tables, respectively. While in the original DeepSet, *Pool* is an element-wise mean pooling, we also experiment with max pooling, motivated by its utility in preserving important characteristics of input embeddings. Given these role-specific representations, we construct the fact representation z_ψ as

$$\begin{aligned}
\tilde{z}_\psi &= f_\psi([h_\psi^A; h_\psi^R; h_\psi^Q]),\\
(\alpha_\psi, \beta_\psi) &= f_r(E_\mathcal{R}[r_\psi]),\\
z_\psi &= \alpha_\psi \odot \tilde{z}_\psi + \beta_\psi,
\end{aligned} \tag{9}$$

where f_ψ and f_r are MLPs, and α_ψ and β_ψ impose a relation-specific affine transformation (Perez et al., 2018). For query facts, where we do not have access to relation information, we use a placeholder "query" relation to avoid target leakage.

K.3 Window Representation

Given a window $\mathcal{W}_i$ in $\mathcal{S}(q)$ with the corresponding fact set $\Psi_i(q)$, we compute a window representation as

$$z_i = f_{\text{Pool}}(\mathcal{W}_i) \in \mathbb{R}^d, \tag{10}$$

where d is the hidden dimension of the encoder. Specifically, we consider two aggregation methods: *bag aggregation* and *hypergraph aggregation*.

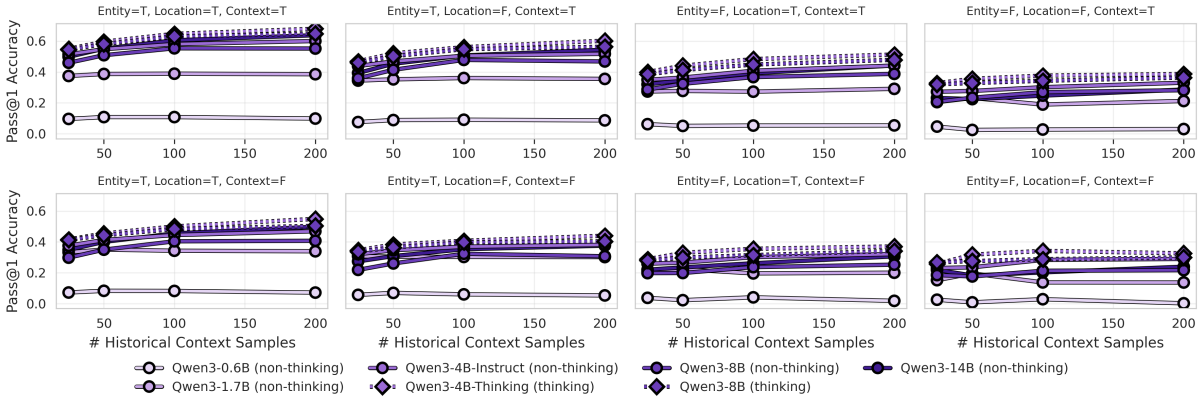


Figure 12: Relation prediction accuracy (%) on `htkgh-polecat` variations for the Qwen3 family.

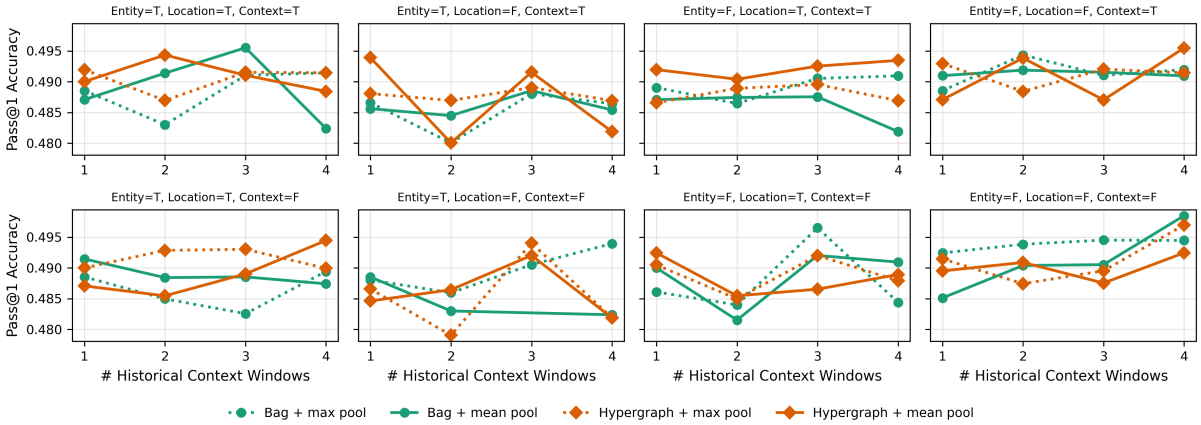


Figure 13: Relation prediction accuracy (%) of GNN models on the final 24 months of the `htkgh-polecat` test set over the number of historical context windows used.

K.3.1 Bag Aggregation

In this variation, we use a DeepSet aggregator to compute the window representation as

$$z_i = g_{\mathcal{W}}(\text{Pool}(\{f_{\mathcal{W}}(z_\psi) : \psi \in \Psi_i(q)\})), \quad (11)$$

where $g_{\mathcal{W}}$ and $f_{\mathcal{W}}$ are MLPs with ReLU activations. This variation is an extreme simplification of the HTKGH structure as we naively aggregate facts within a window without any attention to their relationships.

K.3.2 Hypergraph Aggregation

To utilize relational information, we model each window as a typed hypergraph with facts for nodes and hyperedges between facts that share elements. Specifically, for types $\tau \in \{A, R, Q\}$ we define three typed hyperedges as

$$\mathcal{G}_i^\tau(e) = \{\psi \in \Psi_i(q) : e \in \tau_\psi\}, \quad (12)$$

where τ_ψ is the set of type-τ entities in ψ.

To perform hypergraph message passing, we first initialize node representations as

$$x_\psi^{(0)} = z_\psi \,. \quad (13)$$

Then, we compute a representation for each hyperedge by pooling from its nodes. Specifically, the hyperedge representation at layer l is computed as

$$\begin{aligned} \bar{h}_{i,e}^{\tau,(l)} &= \frac{1}{|\mathcal{G}_i^\tau(e)|} \sum_{\psi \in \mathcal{G}_i^\tau(e)} x_\psi^{(l)}, \\ h_{i,e}^{\tau,(l)} &= f_e^\tau(\bar{h}_{i,e}^{\tau,(l)}), \end{aligned} \quad (14)$$

where f_e^τ is a type-specific MLP. Finally, we use an aggregation scheme similar to R-GCN (Schlichtkrull et al., 2018) to update the node representation for $\psi \in \Psi_i(q)$ as

$$\begin{aligned} m_\psi^{(l)} &= \sum_{\tau \in \{A,R,Q\}} \sum_{e \in \tau_\psi} W_\tau h_{i,e}^{\tau,(l)}, \\ x_\psi^{(l+1)} &= f_u([x_\psi^{(l)}; m_\psi^{(l)}]), \end{aligned} \quad (15)$$

where W_τ is a learnable matrix and f_u is an MLP.

After L layers of message passing, we use a DeepSet aggregator to compute the window representation as

$$z_i = g_{\mathcal{W}}(\text{Pool}(\{f_{\mathcal{W}}(x_\psi^{(L)}) : \psi \in \Psi_i(q)\})), \quad (16)$$

where $g_{\mathcal{W}}$ and $f_{\mathcal{W}}$ are MLPs with ReLU activations.

K.4 Temporal Encoder

To aggregate information over all the windows and construct a final representation, we first create a context sequence as

$$\mathcal{Z}(q) = [z_{\mathcal{H}}, ..., z_1, z_0] \in \mathbb{R}^{(\mathcal{H}+1)\times d}. \quad (17)$$

Then, we compute the final representation as

$$h_q = f_t(\mathcal{Z}(q)) \in \mathbb{R}^d, \quad (18)$$

where f_t is a transformer (Vaswani et al., 2017).

K.5 Linear Classifier

Given the final representation h_q, we use a linear layer to perform relation classification as

$$p(.|\mathcal{S}(q)) = \text{Softmax}(W_c h_q + b) \quad (19)$$

where W_c and b are learnable parameters.

K.6 Loss Functions

To train our models, we use the cross-entropy loss computed as

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|\mathcal{S}(q_i)) \quad (20)$$

where y_i denotes the ground truth relation for query i and N is the batch size.

K.7 Evaluation Setup

Historical Context To construct the contextual information, we use a combination of the query-dependent entity, location, and context filters, analogously to those used in the LLM predictors.

Hyperparameters For each variant, we sweep over $learning\_rate \in \{3e^{-5}, 1e^{-4}, 3e^{-4}\}$ using AdamW optimizer with a weight decay of 0.01. Training is done over eight epochs with a batch size of 256. For the temporal encoder, we use a single-layer encoder-only Transformer with $d\_model = 128$ and $nhead = 4$. For the hypergraph aggregator, we use $L = 2$. Finally, with each window covering one day, we temporally aggregate over 7 days before the query.

K.8 Window Encoder Bottleneck Discussion

While LLMs attend to individual facts in the context, our GNNs collapse all facts in a given window into one vector. Based on our experimental results, this approach seems to be too coarse when the historical context is highly filtered. Moreover, whether models use the bagging or hypergraph aggregation window encoder has little effect on performance. Finally, we observe that using mean or max pooling in set encoders has little effect on the performance.

K.9 Ablation on Number of Windows

Figure 13 illustrates our experimental results when we vary the number of windows of context for our GNN-based models. In these experiments, we do not observe any notable trends in model performance, which supports our information bottleneck hypothesis, where window representations contain a mix of signal and noise that is challenging for relation prediction.