Are Large Language Models All You Need for Temporal Knowledge Graph Forecasting?

Anonymous ACL submission

Abstract

While temporal knowledge graph forecasting (TKGF) approaches have traditionally relied heavily on complex graph neural network architectures, recent advances in large language models (LLMs) and in-context learning (ICL) have presented promising out-of-the-box alternatives. However, little is known about LLMs' limitations and generalization capabilities for TKGF. In this study, we conduct a comparative analysis of complexity (*e.g.*, more number of hops) and sparsity (*e.g.*, relation frequency) confounders between LLMs and supervised models using two weakly annotated TKGF benchmarks. Our experimental results showcase that while LLMs perform on par or outperform supervised models in low-complexity scenarios, their effectiveness diminishes in more complex settings (*e.g.*, multi-step, more number of hops, etc.) where supervised models maintain superior performance.

1 Introduction

Knowledge graphs (KGs) are commonly used structures that store relational information as a graph (Bollacker et al., 2008; Vrandečić and Krötzsch, 2014). While using KGs for keeping static facts is common, they are unsuitable for holding complex dynamic (*i.e.,* temporal) information. Temporal knowledge graphs (TKGs) are extensions of KGs that enable the storage of such information (Leetaru and Schrodt, 2013; García-Durán et al., 2018). Consequently, TKGs allow practitioners to do various predictive tasks on complex temporal data. One critical task that has been empowered by TKGs is temporal knowledge graph forecasting (TKGF) (Gastinger et al., 2023), where the objective is to predict future facts from a set of prior facts before a specific time in a TKG. A hypothetical real-world example of TKGF is to answer the question, *"Who will win the 2024 United States presidential election?"* based on previous political events. This scenario can be represented by the query quadruple $q = ($*General Election, Winner, ?, Nov 2024*$)$ and the time-constrained TKG $\mathcal{G}_t = \{($*Biden, Won, General Election, Nov 2020*$)$, $($*Jorgensen, Lost, General Election, Nov 2020*$), \ldots\}$.

Following the recent advancements in large language models (LLMs), the interest in employing them for temporal knowledge graph forecasting (TKGF) has increased. Recent studies have demonstrated LLMs' effectiveness as general estimators across various function classes (Garg et al., 2022; Mirchandani et al., 2023). Specifically, these models have shown immense potential for TKGF, surpassing state-of-the-art supervised models in some cases (Lee et al., 2023; Liao et al., 2023). These advancements present a cheap, fast, and ready-to-use alternative solution to the state-of-the-art methods, many of which use computationally heavy graph neural network (GNN) architectures. However, despite all their benefits, the broad applications of such solutions for forecasting problems and LLMs' "grey-box" nature give rise to concerns regarding their strengths, limitations, and generalizability.

In this study, we provide insights into the effect of various confounders – arising from relational and temporal patterns – on the effectiveness of LLM-based models for TKGF. To this end, first, we utilize a state-of-the-art rule-based model to generate reasoning rules for existing TKG datasets. Then, based on the generated rules, we create two weakly labeled datasets containing confounder annotations for the test sets. Finally, we use these datasets to compare state-of-the-art supervised models in single-step and multi-step settings (Gastinger et al., 2023) across complexity (*e.g.,* number of unique entities), and sparsity (*e.g.,* relation frequency) confounders (see Table 1 for more thorough examples).

Our experimental results on the annotated datasets derived from the well-known TKG benchmarks ICEWS14 and ICEWS18 (García-Durán

Temporal Rule	Complexity			Sparsity	
	# Unique Entities	# Unique Relations	# Hops	Relation Frequency	Time Interval
$(E_1, \textit{express intent to meet}^{-1}, E_2, T_1) \Rightarrow (E_1, \underbrace{\textit{share information}}_{R}, E_2, T_2)$	2	2	1	f_R	$T_2 - T_1$
$(E_1, \textit{provide military aid}, E_2, T_1) \wedge (E_2, \textit{intend to protect}^{-1}, E_3, T_2)$ $\Rightarrow (E_1, \underbrace{\textit{provide military aid}}_{R}, E_3, T_3)$	3	2	2	f_R	$T_3 - T_1$
$(E_1, \textit{riot}, E_2, T_1) \wedge (E_2, \textit{make statement}, E_1, T_2) \wedge (E_1, \textit{riot}, E_2, T_3)$ $\Rightarrow (E_1, \underbrace{\textit{demonstrate or rally}}_{R}, E_2, T_4)$	2	3	3	f_R	$T_4 - T_1$

Table 1: **Confounder values examples.** The samples are taken from Liu et al. (2022) with some small modifications. Note that f_R refers to the frequency of relation R among all quadruples in the dataset.

et al., 2018) reveal that: (1) LLMs outperform supervised models in scenarios with lower complexity, such as annotated samples with 1-hop patterns in single-step settings or samples involving only one unique relation, and (2) as the variability or complexity of the patterns increases, LLM-based models begin to underperform massively compared to supervised models. This phenomenon is particularly evident in multi-step settings, where LLMs lag behind supervised models in all scenarios.

2 Related Work

Supervised Models. Recent supervised models mostly utilize embedding-based GNNs to enhance their structural and sequential learning capabilities by introducing an autoregressive architecture to aggregate information both globally and locally in RE-Net (Jin et al., 2020), combining convolutional and recurrent architectures for modeling temporal sequences in RE-GCN (Li et al., 2021), introducing neural ordinary differential equations to model temporal sequences in TANGO (Han et al., 2021), and extending convolutional architectures to learn evolutionary patterns in CEN (Li et al., 2022). Moreover, in parallel to these models, other approaches have been introduced in prior works, such as using a copy-mechanism in CyGNet (Zhu et al., 2021), leveraging reinforcement learning on temporal paths in TiTer (Sun et al., 2021), and learning temporal logic rules via temporal random walks in TLogic (Liu et al., 2022).

LLM-based Models. Recent advances in LLMs have drastically improved their capabilities, leading to emergent behaviors such as in-context learning (ICL). ICL allows LLMs to perform tasks conditioned solely on the provided context without any parameter optimization. Utilizing ICL, Lee et al. (2023) introduced the first LLM-based TKGF model, which showed performance on par with state-of-the-art supervised models without any training. Liao et al. (2023); Xia et al. (2024) introduced an improved historical fact retriever and an alignment training procedure, posting better performances than the state-of-the-art supervised models. In parallel, Xia et al. (2024) introduced a fusion between LLM-based and supervised models, leading to performance improvements across the board. While these LLM-based models have shown stellar achievements toward the TKGF task, we still lack a proper understanding of their strengths and limitations, a gap that this work aims to bridge.

Dataset	$\vert\mathcal{E}\vert$	$\vert\mathcal{R}\vert$	# of Facts		Time Granularity
			Train/Valid/Test	Annotated	
ICEWS14	6,869	230	75k/8.5k/7.3k	11,625	1 day
ICEWS18	23,033	256	373k/46k/50k	65,003	1 day

Table 2: **Dataset statistics.** Each dataset consists of historical facts divided into three subsets based on time.

3 Experimental Setup

3.1 Datasets

Our experiments focused on two prominent TKGF datasets: ICEWS14 (García-Durán et al., 2018) and ICEWS18 (Jin et al., 2020) (see Table 2). We specifically chose these datasets because 1) they are commonly used by almost all the prior works in the literature and 2) they pose a much greater challenge to the forecasting models compared to other existing datasets such as WIKI (Leblay and Chekol, 2018) and YAGO (Rebele et al., 2016). Moreover, to keep our results consistent and comparable to previous works, we use the same train/valid/test splits as Gastinger et al. (2023).

3.2 Weak Labeling

One of the challenges we faced in our experiments was the absence of annotations for different confounders in the existing datasets. To overcome this issue, we used TLogic (Liu et al., 2022), a state-of-the-art rule-learning-based TKG model, to annotate test samples with temporal multi-hop patterns. To this end, first, we ran the rule-learning part of TLogic on the combination of all quadruples from the train, valid, and test sets with the number of hops $\in \{1, 2, 3\}$. Then, we annotated each test sample using the matching pattern with the highest score[1], if such a rule existed. Finally, for the annotated test quadruples, we extract various confounders from their associated patterns, including the number of unique entities and relations, the pattern's length denoted as "hop", the relation frequency of the test query, and the time interval (see Table 1 for examples of extracted values). Table 2 provides the annotation statistics.

3.3 Models

For our LLM-based model, we utilize the ICL-based model as described by Lee et al. (2023), which employs `gpt-neox-20b` (Black et al., 2022). This method is an inference-time approach that demonstrates performance comparable to that of supervised models. Moreover, we use GenTKG (Liao et al., 2023) as another LLM-based baseline. This method involves fine-tuning the base language model over a small portion of the training dataset. For both models, we utilize the implementation provided by the authors. Finally, for the TKG baselines we used the following state-of-the-art models with the hyperparameters and implementation as provided by (Gastinger et al., 2023): RE-Net (Jin et al., 2020), RE-GCN (Li et al., 2021), TANGO (Han et al., 2021), CyGNet (Zhu et al., 2021), and CEN (Li et al., 2022).

3.4 Implementation Details

To evaluate each prediction, we retain the top 100 entities with the highest scores (or the highest log probability). This is due to a limitation of the ICL-based models preventing them from predicting entities that do not appear in its context, which at most contains 100 historical facts, bounded by the context length of the underlying model (*i.e.,* `gpt-neox-20b`). This protocol allows us to evaluate and fairly compare the LLM-based and supervised models across our experiments. As for our metrics, we report the Hits@{1,3} based on the list of retained entities for each prediction. All baseline models, except for GenTKG (Liao et al., 2023), report both head and tail prediction performance by generating a head query $(?, r, o, t)$ and a tail query $(s, r, ?, t)$ for each test quadruple (s, r, o, t), following standard practices in the literature. However, GenTKG focuses solely on tail prediction. Our experiments are focused on combined head and tail predictions and include separate comparisons with GenTKG and other LLM-based models. The codebase utilizes PyTorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2020) libraries.

[1]For a reasoning path matched with a pattern, TLogic generates a score by combining rule confidence and temporal recency scores.

4 Experiments

ICL vs. GenTKG. Table 3 presents the single-step tail prediction performance of two prominent LLM-based approaches for TKGF, itemized by the *number of hops* confounder. As evident, the ICL-based approach outperforms GenTKG by a large margin across all scenarios. While this is somewhat surprising, given that GenTKG further fine-tunes LLMs for TKGF, we leave further investigations to future works and continue our experiments with the ICL method only.

LLMs vs. Supervised. We present our experimental results on both single-step (top) and multi-step (bottom) queries in Table 4, grouped by the *number of hops* as the confounder. As evident, LLM-based models only exhibit a better performance with 1-hop queries in the ICEWS14 dataset. Moreover, as the number of hops, an indicator of the pattern complexity, increases, LLMs' performance gap relative to the supervised models widens. Interestingly, this decline in performance is not monotonic in terms of complexity, making it even more challenging to predict the potential pitfalls. For example, LLMs' worst performance in ICEWS14 occurs in 2-hop queries, while the performance on 3-hop queries stays competitive. We observe the same trend when analyzing other confounders related to pattern complexity. For example, LLM-based models outperform the supervised models in patterns involving two unique entities on ICEWS14. However, as the *number of unique entities* increases, the performance of LLM-based models declines (see Table 5 in Appendix B). Similarly, this trend is evident when the samples are grouped

Single-step	Train	ICEWS14						ICEWS18					
		H@1			H@3			H@1			H@3		
		1-hop	2-hop	3-hop	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop
`gpt-neox-20b-entity`	✗	**0.467**	**0.132**	**0.459**	**0.676**	**0.224**	**0.605**	0.324	0.140	0.289	**0.517**	**0.273**	0.465
`gpt-neox-20b-pair`	✗	0.423	0.084	0.440	0.598	0.123	0.548	**0.332**	**0.141**	**0.324**	0.514	0.227	**0.496**
`GenTKG`	✓	0.406	0.128	0.375	0.557	0.207	0.497	0.088	0.066	0.113	0.129	0.089	0.143

Table 3: **Performance (Hits@K)** comparison between GentTKG and ICL methods for single-step tail prediction. The best performance is shown in bold.

Single-step	Train	ICEWS14						ICEWS18					
		H@1			H@3			H@1			H@3		
		1-hop	2-hop	3-hop	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop
`RE-GCN`	✓	0.426	0.152	0.387	0.636	0.322	0.561	0.345	0.195	0.319	0.547	0.355	0.502
`TANGO`	✓	0.364	0.120	0.362	0.545	0.248	0.502	0.297	0.163	0.283	0.488	0.310	0.455
`CEN`	✓	0.433	0.152	0.390	0.632	0.300	0.562	0.339	0.189	0.311	0.540	0.343	0.491
Average		0.408	0.141	0.380	0.604	0.290	0.542	0.327	0.183	0.304	0.525	0.336	0.483
Median		0.426	0.152	0.387	0.632	0.300	0.561	0.339	0.189	0.311	0.540	0.343	0.491
`gpt-neox-20b-entity`	✗	0.464	0.106	0.386	0.658	0.178	0.540	0.298	0.112	0.279	0.481	0.224	0.436
Δ Average		0.056	-0.035	0.007	0.054	-0.112	-0.002	-0.029	-0.071	-0.025	-0.044	-0.112	-0.046
Δ Median		0.037	-0.046	-0.001	0.027	-0.122	-0.021	-0.041	-0.078	-0.032	-0.059	-0.119	-0.055
`gpt-neox-20b-pair`	✗	0.416	0.068	0.379	0.583	0.098	0.514	0.310	0.109	0.307	0.487	0.179	0.471
Δ Average		0.009	-0.074	-0.001	-0.022	-0.192	-0.027	-0.017	-0.073	0.003	-0.038	-0.157	-0.012
Δ Median		-0.010	-0.085	-0.008	-0.049	-0.202	-0.046	-0.029	-0.080	-0.004	-0.053	-0.164	-0.020

Multi-step	Train	ICEWS14						ICEWS18					
		H@1			H@3			H@1			H@3		
		1-hop	2-hop	3-hop	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop
`RE-NET`	✓	0.373	0.133	0.360	0.541	0.259	0.513	0.288	0.160	0.278	0.480	0.314	0.450
`RE-GCN`	✓	0.366	0.157	0.349	0.554	0.300	0.490	0.295	0.182	0.289	0.483	0.330	0.458
`CyGNet`	✓	0.355	0.119	0.345	0.536	0.260	0.499	0.255	0.134	0.261	0.449	0.283	0.441
Average		0.364	0.136	0.351	0.543	0.273	0.501	0.279	0.159	0.276	0.471	0.309	0.450
Median		0.366	0.133	0.349	0.541	0.260	0.499	0.288	0.160	0.278	0.480	0.314	0.450
`gpt-neox-20b-entity`	✗	0.343	0.087	0.321	0.496	0.169	0.446	0.197	0.089	0.197	0.313	0.178	0.307
Δ Average		-0.021	-0.049	-0.030	-0.048	-0.104	-0.054	-0.082	-0.070	-0.079	-0.158	-0.131	-0.142
Δ Median		-0.023	-0.046	-0.028	-0.045	-0.091	-0.053	-0.090	-0.072	-0.081	-0.167	-0.136	-0.143
`gpt-neox-20b-pair`	✗	0.309	0.065	0.326	0.437	0.089	0.434	0.237	0.087	0.256	0.379	0.144	0.385
Δ Average		-0.055	-0.071	-0.025	-0.106	-0.183	-0.067	-0.042	-0.072	-0.020	-0.092	-0.164	-0.065
Δ Median		-0.056	-0.068	-0.023	-0.104	-0.170	-0.065	-0.050	-0.073	-0.022	-0.102	-0.170	-0.066

Table 4: **Performance (Hits@K)** comparison between supervised models and ICL for single-step (top) and multi-step (bottom) prediction, grouped by the **number of hops** as the confounder. The first group consists of supervised models, whereas the second group consists of ICL models, *i.e.,* `GPT-NeoX`. The green and red colors represent where LLM is outperforming and underperforming the average performance of the supervised models.

by *number of unique relations* (see Table 6 in Appendix B). When the samples are grouped by *relation frequency*, the LLM-based models perform on par or moderately outperform the supervised models only in the ICEWS14 single-step setting. In all other cases, the supervised models outperform the LLM-based models. However, the upward trend in Figure 1 (Appendix B) indicates that as relation frequency increases, the performance gap between the LLM-based and supervised models decreases. Moreover, when the samples are grouped by *time interval* (see Figure 2 in Appendix B), the supervised models consistently outperform the LLM-based models. We observe that LLM-based models perform worse in the multi-step setup across all confounders than their counterpart average supervised models. Finally, the performance gap is wider on the ICEWS18 dataset compared to the ICEWS14 dataset, which could be attributed to the fact that ICEWS18 is more dense and challenging.

5 Conclusion

In this paper, we presented an in-depth analysis of the effect of various confounders on the predictive power of LLM-based and supervised models for TKGF. Specifically, we created two annotated benchmarks for testing TKGF models across varied complexities and sparsity levels. Our experimental results indicate that while LLMs are effective in low-complexity scenarios, their performance rapidly deteriorates as the complexity of the patterns increases. These findings highlight the need for further development and optimization of LLMs for TKGF tasks.

Limitations

The potential problem with weakly annotating the existing datasets is the propagation of biases from both the annotator and the source dataset, potentially resulting in inflated or misleading performances. To overcome this issue, we need a new TKGF benchmarking framework for LLM-based models that focuses on controlling the confounders of the test samples. Specifically, this framework should generate pairs of historical context and query quadruple (*i.e.,* (G_t, q)) with controllable relational pattern distributions in historical context and known ground truth for the query quadruple. Such a benchmark allows us to examine the pure abilities of LLMs for reasoning over relational patterns and the effectiveness of context retrieval algorithms for gathering relevant historical facts.

References

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

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.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598. Curran Associates, Inc.

Julia Gastinger, Timo Sztyler, Lokesh Sharma, Anett Schuelke, and Heiner Stuckenschmidt. 2023. Comparing apples and oranges? on the evaluation of methods for temporal knowledge graph forecasting. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, pages 533–549, Cham. Springer Nature Switzerland.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8352–8364, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1771–1776, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

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, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022. Complex evolutional pattern learning for temporal knowledge graph reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 290–296, Dublin, Ireland. Association for Computational Linguistics.

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*, SIGIR '21, page 408–417, New York, NY, USA. Association for Computing Machinery.

Ruotong Liao, Xu Jia, Yunpu Ma, and Volker Tresp. 2023. Gentkg: Generative forecasting on temporal knowledge graph. *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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):4120–4127.

Suvir Mirchandani, Fei Xia, Pete Florence, brian ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large language models as general pattern machines. In *7th Annual Conference on Robot Learning*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II*, page 177–185, Berlin, Heidelberg. Springer-Verlag.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. TimeTraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8306–8319, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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 Xiaoyu Zhang. 2024. Enhancing temporal knowledge graph forecasting with large language models via chain-of-history reasoning. *arXiv preprint arXiv:2402.14382*.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4732–4740.

A Formal Definition of TKGF

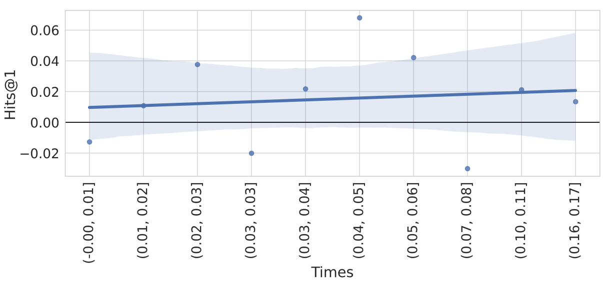
Formally, a TKG $\mathcal{G} = (\mathcal{Q}, \mathcal{E}, \mathcal{R}, \mathcal{T})$ comprises a set of quadruples $\mathcal{Q}$ in the form (s, r, o, t), where s and o are entities within $\mathcal{E}$, r is a relation within $\mathcal{R}$, and t is a timestamp from $\mathcal{T}$. The TKG forecasting task aims to predict a missing entity in future quadruples, either as $(s, r, ?, t)$ for tail prediction or $(?, r, o, t)$ for head prediction, using historical data from the graph. This involves scoring all entities such that the true entity receives a higher ranking than others.

B Full Experimental Results

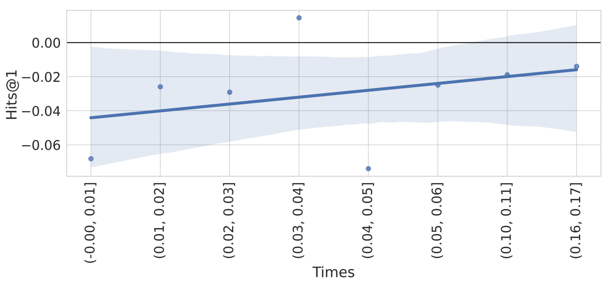
Single-step	Train	ICEWS14						ICEWS18					
		H@1			H@3			H@1			H@3		
		2	3	4	2	3	4	2	3	4	2	3	4
RE-GCN	✓	0.452	0.368	0.106	0.660	0.552	0.232	0.350	0.329	0.160	0.552	0.517	0.302
TANGO	✓	0.394	0.342	0.083	0.572	0.490	0.180	0.302	0.293	0.128	0.495	0.470	0.255
CEN	✓	0.461	0.369	0.098	0.658	0.546	0.230	0.344	0.323	0.147	0.546	0.509	0.283
Average		0.436	0.360	0.096	0.630	0.529	0.214	0.332	0.315	0.145	0.531	0.499	0.280
gpt-neox-20b-entity	✗	0.492	0.353	0.078	0.681	0.508	0.168	0.303	0.273	0.129	0.486	0.437	0.222
Δ Average		0.056	-0.007	-0.017	0.051	-0.021	-0.046	-0.029	-0.042	-0.016	-0.045	-0.061	-0.057
Δ Median		0.039	-0.015	-0.020	0.023	-0.038	-0.063	-0.041	-0.050	-0.018	-0.060	-0.071	-0.060
gpt-neox-20b-pair	✗	0.446	0.345	0.068	0.613	0.483	0.099	0.316	0.304	0.120	0.494	0.469	0.193
Δ Average		0.011	-0.014	-0.028	-0.016	-0.047	-0.115	-0.017	-0.011	-0.025	-0.038	-0.030	-0.087
Δ Median		-0.006	-0.022	-0.031	-0.044	-0.063	-0.131	-0.029	-0.019	-0.027	-0.053	-0.040	-0.090

Multi-step	Train	ICEWS14						ICEWS18					
		H@1			H@3			H@1			H@3		
		2	3	4	2	3	4	2	3	4	2	3	4
RE-NET	✓	0.400	0.341	0.099	0.573	0.492	0.202	0.293	0.287	0.130	0.487	0.466	0.256
RE-GCN	✓	0.382	0.350	0.106	0.564	0.505	0.208	0.301	0.301	0.137	0.490	0.475	0.267
CyGNet	✓	0.379	0.333	0.083	0.562	0.496	0.170	0.260	0.267	0.114	0.454	0.456	0.237
Average		0.387	0.341	0.096	0.566	0.498	0.193	0.285	0.285	0.127	0.477	0.466	0.253
gpt-neox-20b-entity	✗	0.371	0.294	0.064	0.525	0.419	0.136	0.202	0.197	0.082	0.319	0.312	0.156
Δ Average		-0.015	-0.047	-0.032	-0.041	-0.078	-0.057	-0.082	-0.087	-0.045	-0.158	-0.154	-0.097
Δ Median		-0.011	-0.047	-0.035	-0.039	-0.077	-0.066	-0.091	-0.089	-0.048	-0.168	-0.154	-0.100
gpt-neox-20b-pair	✗	0.341	0.302	0.058	0.467	0.414	0.084	0.244	0.254	0.088	0.387	0.384	0.144
Δ Average		-0.046	-0.040	-0.038	-0.099	-0.084	-0.109	-0.040	-0.031	-0.039	-0.091	-0.082	-0.109
Δ Median		-0.042	-0.039	-0.041	-0.096	-0.082	-0.118	-0.048	-0.033	-0.042	-0.100	-0.082	-0.112

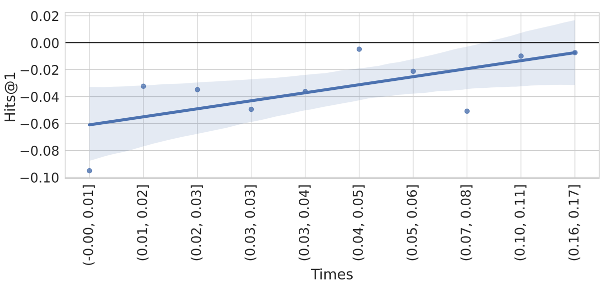
Table 5: **Performance (Hits@K)** comparison between supervised models and ICL for single-step (top) and multi-step (bottom) prediction, grouped by the number of **number of unique entities** as confounder. The first group consists of supervised models, whereas the second group consists of ICL models, *i.e.*, GPT-NeoX with a history length of 100. The green and red colors represent where LLM is outperforming and underperforming the average performance of the supervised models.



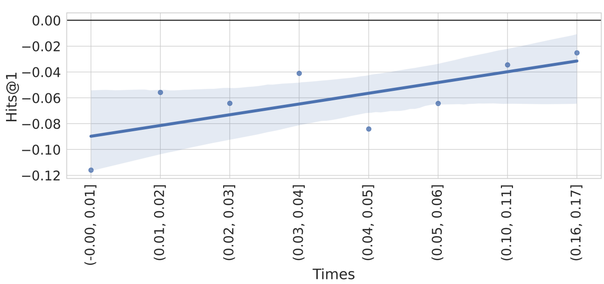
(a) ICEWS14 Single-step



(b) ICEWS18 Single-step



(a) ICEWS14 Multi-step



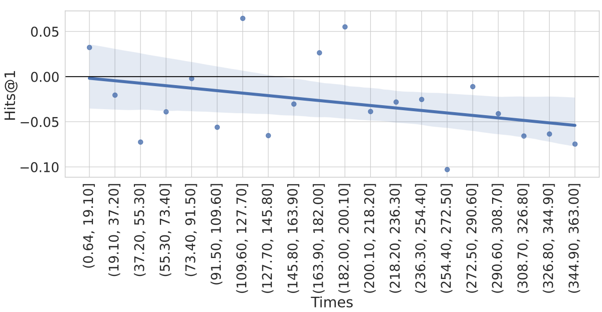
(b) ICEWS18 Multi-step

Figure 1: **Hits@1 difference** between the average performance of ICL and the average performance of supervised models,grouped by the **relation frequency** confounder, for single-step (top) and multi-step (bottom) prediction.

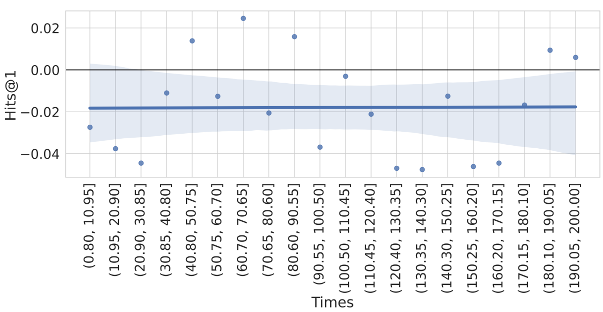
Single-step	Train	ICEWS14								ICEWS18							
		H@1				H@3				H@1				H@3			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
RE-GCN	✓	0.495	0.357	0.344	0.403	0.717	0.548	0.530	0.553	0.346	0.319	0.308	0.284	0.550	0.512	0.486	0.456
TANGO	✓	0.455	0.292	0.318	0.380	0.646	0.454	0.469	0.497	0.309	0.269	0.276	0.248	0.508	0.451	0.441	0.411
CEN	✓	0.501	0.366	0.345	0.403	0.702	0.549	0.524	0.563	0.338	0.313	0.301	0.279	0.539	0.504	0.477	0.449
Average		0.484	0.338	0.335	0.396	0.689	0.517	0.508	0.538	0.331	0.300	0.295	0.270	0.533	0.489	0.468	0.439
Median		0.495	0.357	0.344	0.403	0.702	0.548	0.524	0.553	0.338	0.313	0.301	0.279	0.539	0.504	0.477	0.449
gpt-neox-20b-entity	✗	0.570	0.363	0.351	0.358	0.770	0.531	0.500	0.512	0.301	0.282	0.242	0.224	0.482	0.456	0.391	0.357
Δ Average		0.086	0.024	0.016	-0.037	0.081	0.014	-0.008	-0.026	-0.030	-0.018	-0.052	-0.046	-0.051	-0.033	-0.077	-0.082
Δ Median		0.074	0.005	0.008	-0.045	0.067	-0.017	-0.024	-0.041	-0.037	-0.031	-0.058	-0.055	-0.058	-0.047	-0.086	-0.093
gpt-neox-20b-pair	✗	0.585	0.295	0.344	0.322	0.820	0.408	0.480	0.414	0.358	0.279	0.268	0.226	0.566	0.441	0.411	0.334
Δ Average		0.101	-0.043	0.009	-0.074	0.131	-0.109	-0.028	-0.124	0.027	-0.021	-0.027	-0.044	0.033	-0.048	-0.057	-0.105
Δ Median		0.090	-0.062	0.001	-0.081	0.118	-0.140	-0.044	-0.139	0.020	-0.033	-0.033	-0.052	0.026	-0.063	-0.066	-0.115

Multi-step	Train	ICEWS14								ICEWS18							
		H@1				H@3				H@1				H@3			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
RE-NET	✓	0.455	0.308	0.317	0.370	0.620	0.466	0.476	0.517	0.304	0.261	0.268	0.247	0.508	0.440	0.441	0.408
RE-GCN	✓	0.425	0.312	0.317	0.358	0.636	0.476	0.461	0.485	0.313	0.269	0.286	0.250	0.506	0.448	0.450	0.410
CyGNet	✓	0.466	0.271	0.314	0.342	0.662	0.433	0.469	0.494	0.287	0.227	0.252	0.216	0.490	0.411	0.428	0.378
Average		0.448	0.297	0.316	0.357	0.639	0.458	0.469	0.499	0.301	0.252	0.268	0.237	0.501	0.433	0.439	0.399
Median		0.455	0.308	0.317	0.358	0.636	0.466	0.469	0.494	0.304	0.261	0.268	0.247	0.506	0.440	0.441	0.408
gpt-neox-20b-entity	✗	0.415	0.280	0.287	0.311	0.569	0.414	0.413	0.440	0.219	0.181	0.179	0.155	0.330	0.299	0.284	0.246
Δ Average		-0.033	-0.017	-0.029	-0.046	-0.070	-0.044	-0.056	-0.058	-0.082	-0.072	-0.090	-0.082	-0.171	-0.134	-0.155	-0.153
Δ Median		-0.039	-0.028	-0.030	-0.047	-0.067	-0.052	-0.056	-0.054	-0.085	-0.080	-0.089	-0.091	-0.175	-0.141	-0.156	-0.162
gpt-neox-20b-pair	✗	0.448	0.223	0.295	0.286	0.628	0.312	0.403	0.358	0.290	0.210	0.230	0.185	0.467	0.333	0.341	0.275
Δ Average		-0.001	-0.074	-0.021	-0.070	-0.012	-0.147	-0.066	-0.141	-0.012	-0.043	-0.038	-0.052	-0.034	-0.100	-0.099	-0.123
Δ Median		-0.007	-0.085	-0.022	-0.072	-0.008	-0.155	-0.066	-0.137	-0.014	-0.051	-0.037	-0.061	-0.039	-0.106	-0.100	-0.132

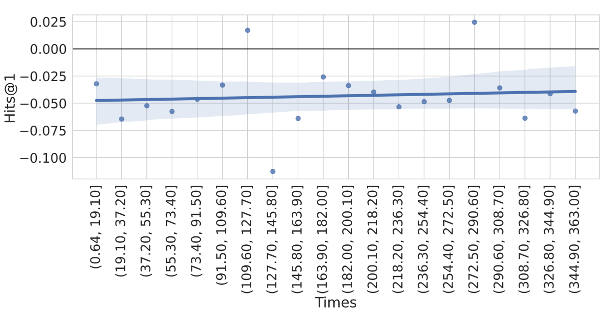
Table 6: **Performance (Hits@K)** comparison between supervised models and ICL for single-step (top) and multi-step (bottom) prediction, grouped by the number of **number of unique relations** as confounder. The first group consists of supervised models, whereas the second group consists of ICL models, *i.e.,* GPT-NeoX with a history length of 100. The green and red colors represent where LLM is outperforming and underperforming the average performance of the supervised models.



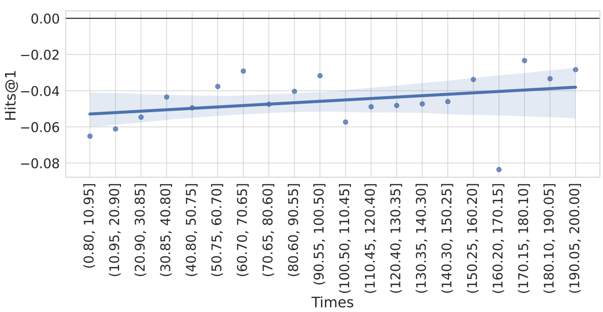
(a) ICEWS14 Single-step



(b) ICEWS18 Single-step



(a) ICEWS14 Multi-step



(b) ICEWS18 Multi-step

Figure 2: **Hits@1 difference** between the average performance of ICL and the average performance of supervised models, grouped by the **time interval** confounder, for single-step (top) and multi-step (bottom) prediction.