On the Evaluation of Methods for Temporal Knowledge Graph Forecasting

Julia Gastinger^{1,2}, Timo Sztyler^{1}, Lokesh Sharma^{1}, Anett Schuelke^{1}
`firstname.lastname@neclab.eu`

^{1}NEC Laboratories Europe
Heidelberg, Germany

^{2}University of Mannheim
Mannheim, Germany

Abstract

Due to its ability to incorporate and leverage time information in relational data, Temporal Knowledge Graph (TKG) learning has become an increasingly studied research field. With the goal of predicting the future, researchers have presented innovative methods for what is called Temporal Knowledge Graph Forecasting. However, the experimental procedures in this line of work show inconsistencies that strongly influence empirical results and thus lead to distorted comparisons among models. This work focuses on the evaluation of TKG Forecasting models: we describe evaluation settings commonly used in this research area and shed light on its scholarship issues. Further, we provide a unified evaluation protocol and carry out a re-evaluation of state-of-the-art models on the most common datasets under such a setting. Finally, we show the difference in results caused by different evaluation settings. We believe that this work provides a solid foundation for future evaluations of TKG Forecasting models and can thus contribute to the development of this growing research area.

1 Introduction

Temporal Knowledge Graphs (TKG) are Knowledge Graphs (KG) where facts occur, recur or evolve over time [1]. TKG can accommodate time-evolving multi-relational data by extending facts with a timestamp to indicate that a triple is valid at this timestamp [2]. The research field of TKG Forecasting, or TKG Extrapolation, aims at predicting facts at future timesteps, based on the KG history [3]. Recently, various methods have been proposed to advance the field.

Unfortunately, and despite the progress made so far in TKG Forecasting, various reported experimental settings show discrepancies: first, the existing models are evaluated on scores computed with different filter settings; second, models for single-step prediction that predict one step to the future are lumped together with models for multi-step prediction that predict multiple steps to the future; third, multiple versions of the same datasets exist. Last but not least, some models do use the information from the validation set for testing, whereas others do not. These four issues can strongly influence the empirical results and significantly decrease comparability across works. Consequently, it is very difficult to understand existing methods' strengths or weaknesses or to identify the currently best-performing method.

In this paper, we address the aforementioned issues in the evaluation of TKG Forecasting models. We first provide an overview of existing models for TKG Forecasting (Section 2). We then describe common evaluation settings and compare those settings used in state-of-the-art models to highlight the inconsistencies (Section 3). In this context, we explain the problems we discovered for each setting. Because it is essential to evaluate models in a consistent way, we form a unified evaluation protocol using reasonable and sound evaluation settings (Section 4). We re-evaluate state-of-the-art

NeurIPS 2022 Temporal Graph Learning Workshop held in New Orleans, United States

models on this protocol and show results for seven state-of-the-art models on five commonly used datasets (Section 5). In addition, we provide insights into the influence of different setups on the result scores. Our hope is to set a new standard for rigorous evaluations of new models in this growing research field. Our contributions are:

1. A comprehensive discussion of evaluation settings and accompanying problems for TKG Forecasting.
2. The design of a unified evaluation protocol for TKG Forecasting from reasonable evaluation settings.
3. An extensive re-evaluation of state-of-the-art models on a consistent evaluation protocol, showing results and insights on the influence of different evaluation settings on these results.

Disclaimer: This work does not want to belittle the effort others have put into the development of methods for TKG Forecasting. Instead, we want to point out inconsistencies that do exist and how they may impact the final results. By suggesting a uniform evaluation protocol for TKG Forecasting, we provide a solid basis for a fair, objective, and consistent comparison of future contributions with existing methods.

2 Related Work and Terminology

Temporal Knowledge Graph Forecasting: In recent years (2019-2022), researchers have proposed various methods for TKG Forecasting. Some of them leverage Graph Neural Networks [4, 5] in combination with a sequential approach to integrate structural and sequential information in the learning process. This includes RE-Net [6], RE-GCN [7], TANGO [8], xERTE [2], and CEN [9]. Further, CluSTeR [10] and TimeTraveler [3], introduce Reinforcement Learning Approaches for TKG Forecasting. On the other hand, the model TLogic [11], is a rule-based approach. In addition, CyGNet [12] predicts based on the appearance and repetition of historical facts.

Please see the supplementary material for more detailed information on each method. In our work, we analyze the evaluation discrepancies of the introduced models and evaluate the models on a joint evaluation protocol.

Evaluation of Graph-based Machine Learning Models: When conducting empirical evaluations of machine learning algorithms, various issues can arise [13]. Such problems have been reported and partially addressed in various subfields, but in the following, we limit the discussion to works in the field of Graph Machine Learning. Shchur et al. [14] describe the shortcomings of evaluation strategies for Graph Neural Network models for node classification. Errica et al. [15] focus on graph classification, providing standard practices that should be avoided for a fair comparison. Further, Rossi et al. [16] describe shortcomings in the evaluation of KG link prediction. Han et al. [17] focus on the evaluation of models for TKG completion (not forecasting). Our work is the first to study evaluation problems for TKG Forecasting.

Terminology: A TKG is formalized as a sequence of timestamped Knowledge Graphs, $G = (G_1, G_2, ..., G_t, ...)$. A timestamped KG $G_t = \{\mathcal{V}, \mathcal{R}, \mathcal{E}_t\}$, or KG snapshot, describes the TKG at timestep t, with the set of entities $\mathcal{V}$, the set of relations $\mathcal{R}$, and the set of facts $\mathcal{E}_t$ at discrete timestamp t. Facts $\mathcal{E}_t$ are quadruples (s, r, o, t), with $s, o, \in \mathcal{V}$, and $r \in \mathcal{R}$, for example (Kamala Harris, visit, France, 2021-11-10). Entity prediction for TKG Forecasting is the task of predicting the missing object entity $(s, r, ?, t + k)$ and subject entity $(?, r, o, t + k)$ for a query, with $k \in \mathbb{N}^+$. [7]

3 Description of Evaluation Settings and Evaluation Problems

In this chapter, we subsequently focus on evaluation settings for TKG forecasting. In each subsection, we first describe a setting, and second, describe problems that we have encountered in that setting. In addition, Table 1, provides an overview, showing the settings each model uses by default. We refer to the respective parts of the table in each subsection. Further, the table contains links to the published code for each model, if available.

3.1 Filter Settings for link prediction metrics

Researchers in TKG Forecasting evaluate the models on metrics known from static link prediction, namely Mean Reciprocal Rank (MRR) and Hits@k, with $k = 1, 3, 10$. There are three settings which have been introduced subsequently, *raw*, *static filter*, and *time-aware filter*:

Raw: As introduced by Bordes et al. [18], for each test triple $(s_{test}, r_{test}, o_{test})$, remove the object $(s_{test}, r_{test}, ?)$, and compute the score that the model assigns for each entity $v \in \mathcal{V}$ to be the object in that triple, where the set of all possible triples (s_{test}, r_{test}, v) is termed corrupted triples. Sort the scores in descending order, and note the rank of the correct entity o_{test}. Repeat this by removing the subject $(?, r_{test}, o_{test})$. The MRR is the mean of the reciprocal of these ranks across all queries from the test set, and Hits@k is the proportion of correct entities ranked in the top k.

Static filter: To avoid counting higher ranks from other valid predictions as errors and thus having flaws in the metrics, Bordes et al. [19] propose to remove all triples (except the triple of interest) that appear in the train, valid, and test set from the list of corrupted triples.

Time-aware filter: Han et al. [20] note that the static filter setting is inappropriate for temporal link prediction because it filters out all triples that have ever appeared from the list of corrupted triples, ignoring the time validity of facts. As a consequence, it does not consider predictions of such triples as erroneous. For example, if there is a test query (Barack Obama, visit, India, 2015-01-25) and if the train set contains (Barack Obama, visit, Germany, 2013-01-18), the triple (Barack Obama, visit, Germany) is filtered out for the test query according to the static filter setting, even though it is not true for 2015-01-25 [2]. For this reason numerous works [2, 11, 8, 3, 9, 10] apply the *time-aware filter* setting which only filters out quadruples with the same timestamp as the test query. In the above example, (Barack Obama, visit, Germany, t) would only be filtered out for the given test query, if it had the timestamp $t =$ 2015-01-25, and otherwise stay in the list of corrupted triples.

Problem 1: Different Filter Settings. The works introduced in Section 2 do present result scores with MRR and Hits@k using the above-described filter settings. However, not all works report results on all filter settings, which is a problem, as it decreases comparability across works. Further, as mentioned above, the raw, and especially the static filter setting are not appropriate for TKG Forecasting. The first part of Table 1 illustrates the filter settings that each model reports.

3.2 Single-step and Multi-step prediction

Methods for forecasting run under two different prediction settings, single-step and multi-step prediction. Single-step (or one-step) prediction means that the model always forecasts the next timestep [21]. The ground truth facts are fed after every timestep before predicting the subsequent timestep. Multi-step prediction means that the model forecasts more than one future time step [21]. More specifically, in TKG Forecasting, the model predicts all timesteps from the test set, without seeing any ground truth information in between. As described by Brownlee [21], multi-step prediction is more challenging, as the model can only leverage information from its own forecasts, and uncertainty accumulates with an increasing number of forecasted timesteps.

Problem 2: Comparison of multi-step and single-step setting. The models described in Section 2 run in different settings. Some can do single-step prediction only, some can do multi-step prediction only, and some do both (see Table 1, second part). Still, single-step models are compared to multi-step models without drawing attention to the different setups. For example, TLogic [11] and TANGO [8] (single-step) are compared to RE-Net [6] (multi-step), and xERTE [2] to CyGNet [12]. The second part of Table 1 shows each model's prediction setting.

3.3 Datasets

Researchers in the domain of TKG Forecasting use the following datasets: Three instances of ICEWS [22]: ICEWS05-15 [23], ICEWS14 [23], and ICEWS18 [24], where the numbers mark the respective years; further, YAGO [25] and WIKI [26], preprocessed according to Jin et al. [24], as well as GDELT [27]. The supplementary material contains dataset statistics.

Problem 3: Multiple versions of the same dataset. The models described in Section 2 report results on different versions of the same dataset. For instance, three versions exist for ICEWS14. This hinders the comparability of results across works, causing confusion and potential errors. The third part of Table 1 shows an overview of different versions of each dataset, describing each version (marked with (a), (b), (c)) by the number of training triples. One version of the ICEWS14 dataset (see Table 1, version (c)) is especially problematic, as it does not contain a validation set. Instead, the test set is used for both validation and testing. Thus, with this setting, the test set is leaked during training.

3.4 Train, Validation, and Test Set

Researchers in TKG Forecasting split each dataset D into a training D_{train}, validation D_{valid}, and test set D_{test}. The model's training is conducted on D_{train}, not using information contained in D_{valid} or D_{test}. D_{valid} can be used for monitoring the training process, and selecting the best model (parameters) across epochs. There are different options to use the validation set during testing:

(a) The model can leverage all information from D_{train}, but not from D_{valid}, to predict D_{test}. This is consistent with the setting in link prediction for static knowledge graphs.
(b) The model can leverage all information from D_{train} and from D_{valid}, to predict D_{test}. This means, if a model has to answer the query $(s, r, ?, n)$ during testing, all quadruples from D_{train} and D_{valid} can be used. This is consistent with the setting used in time-series forecasting.

Problem 4: Usage of Validation set for Testing. For multi-step setting, during testing, some models (CygNet, TLogic) do not use the information from the validation set (option (a)), whereas others (RE-GCN, RE-Net) do use it (option (b)), see the fourth part of Table 1. Not using the information from the validation set leads to a significantly harder task, as the model needs to forecast more steps in the future: Instead of starting to predict the next unknown timestep $t+1$ for the first test set sample, the model needs to already predict the timestep $t + num_{valid} + 1$, with num_{valid} being the number of timesteps in the validation set, as an information gap between training and testing.

3.5 Problem Summary

When putting all four problems together, a dramatic picture emerges: results have been compared using different filter settings, prediction settings, dataset versions, and dataset splits. Table 1 illustrates the scattered landscape of evaluation settings, where no two models have ever been evaluated on identical settings. Without a uniform and standardized evaluation protocol, we will never be able to gauge true progress in the field. Still, in existing work, the methods are compared to each other, leading to confusion and inconsistencies.

4 A unified evaluation protocol

To tackle the problems introduced in Section 3, it is essential to evaluate TKG models in a consistent way. For this reason, we introduce a unified evaluation protocol with clear and reproducible choices.[1]

Filter settings: We report results on the time-aware filter setting. As explained in Section 3.1 this setting avoids counting higher ranks from other valid predictions as errors while taking into account time validity of facts.

Single-step and Multi-Step: While both settings are valid, the comparison of results for different settings is not fair (see Section 3.2). The setting to be used depends on the use case and on the methods' capabilities. If the method can predict in single- and multi-step, we re-evaluate it on both settings.

Datasets Versions: The same dataset versions should be used across works to ensure comparability. We suggest using version (a) for each dataset (see Table 1). We selected the dataset versions used by the authors of RE-GCN [7], mainly because these are (among) the most commonly used versions across all works. The supplementary material shows dataset statistics.

Train, Validation, and Test Set Usage: We use the train, validation, and test sets as described in Section 3.4, option (b), where the information from the validation set can be used for testing, to avoid time gaps between training and testing. In addition, we make sure that the test set is never used for model selection and the datasets are split based on ordered timestamps, whereas one timestamp should not belong to two different sets.

[1]The supplementary material also contains a checklist for benchmark experiments in this field.

Table 1: Methods and their experimental settings: Filter settings (Section 3.1), settings for single- and multi-step prediction (Section 3.2), dataset versions ((a), (b), (c)) used in papers (Section 3.3), and validation set usage (Section 3.4). We report dataset versions by the number of quadruples in the training set. An entry ✓ means that the model reported results on the respective setting, and an entry - that it does not. An entry *args* means, that the method provides the option to set this in the args of the code, but does not report the results in the paper. An entry *?* means that we cannot answer this question, as the code is not publicly available, or was published very recently at the time of writing.

Name	RE-GCN	RE-Net	xER-TE	CyG-Net	TLogic	TANGO	Time Trav-eler	CEN	CluS-TeR
Filter settings:									
raw	✓	✓	-	-	-	✓	-	-	✓
static	-	✓	-	✓	-	✓	-	-	-
time-aware	-	-	✓	-	✓	✓	✓	✓	✓
Prediction settings:									
single-step	args	partly[a]	✓	-	✓	✓	✓	?	?
multi-step	✓	✓	-	✓	args	-	-	?	?
Datasets:									
ICEWS14									
(a): 74845	✓	-	-	-	-	-	-	✓	✓
(b): 63685	-	-	✓	-	✓	-	✓	-	-
(c): 323895 w/o valid[b]	-	✓	-	✓	-	✓	-	-	-
ICEWS18									
(a): 373018	y	✓	✓	✓	✓	✓	✓	✓	✓
ICEWS05-15									
(a): 368868	✓	-	-	-	✓	-	-	-	✓
(b): 322958	-	-	✓	-	-	-	✓	-	-
(c): 369104	-	-	-	-	-	✓	-	-	-
GDELT									
(a): 1734399	✓	✓-	✓	-	-	-	-	✓	
YAGO									
(a): 161540	✓	✓	-	✓	-	✓	✓	-	-
(b): 51205	-	-	✓	-	-	-	-	-	-
WIKI									
(a): 539286	✓	✓	-	✓	-	✓	✓	✓	-
Validation Set for Testing:									
Use Valid	✓	✓	✓	-	-	✓	✓	?	?
Reference	[7]	[6]	[2]	[12]	[11]	[8]	[3]	[9]	[10]
Code Published	✓[c]	✓[d]	✓[e]	✓[f]	✓[g]	✓[h]	✓[i]	✓[j]	-

[a] RE-NET published results for the datasets ICEWS18 and GDELT ([6], Table 2, RE-Net w. GT). The published code does not provide the option to set this in the arguments.
[b] This specific version of ICEWS14 comes without validation set. Instead, the test set is used for validation.
[c] `https://github.com/Lee-zix/RE-GCN` [d] `https://github.com/INK-USC/RE-Net`
[e] `https://github.com/TemporalKGTeam/xERTE` [f] `https://github.com/CunchaoZ/CyGNet`
[g] `https://github.com/liu-yushan/TLogic` [h] `https://github.com/TemporalKGTeam/TANGO`
[i] `https://github.com/JHL-HUST/TITer/` [j] `https://github.com/Lee-zix/CEN`

5 Experiments

In the following, we show the results for seven models[2], and five datasets[3]. The supplementary material contains additional information on specific experimental settings. Please find the source code with scripts for experiments and evaluation at `https://github.com/nec-research/TKG-Forecasting-Evaluation`.

[2]Because CEN [9] was published recently and before the time of writing this paper, we could not integrate the results. However, thanks to the proposed evaluation protocol, it should not be difficult to re-evaluate it by following the instructions in our GitHub repository.

[3]Because of memory and runtime issues for multiple models due to its large amount of timestamps, and its similarity to the other ICEWS datasets, we excluded the dataset ICEWS05-15. By running the script as described in our GitHub repository, interested readers can include this dataset (provided they have enough compute power).

Table 2: Experimental results for multi-step and single-step prediction with datasets GDELT, YAGO, WIKI (top), and ICEWS14, ICEWS18 (bottom). Results for single-step prediction should not be compared to results for multi-step prediction. We report mean reciprocal rank (MRR), and Hits@k (H@k), with $k = 1, 3, 10$ in time-aware filter setting. The best results for each setting are marked in bold.

multi-step setting (time filter)	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	19.64	12.47	20.85	33.62	**75.40**	**71.75**	**77.67**	81.70	62.72	59.48	64.89	67.87
RE-Net	**19.71**	**12.48**	**20.90**	**33.93**	58.21	53.44	61.31	66.26	49.47	47.21	50.70	53.04
CyGNet	19.08	11.88	20.29	33.07	69.02	61.38	74.29	**83.42**	58.26	52.51	62.41	67.56
TLogic	17.68	11.26	18.90	30.29	66.93	63.14	70.63	71.58	**63.99**	**61.31**	**66.36**	**68.22**
single-step setting (time filter)	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	19.75	12.51	21.02	33.88	82.20	78.72	84.24	88.48	78.65	74.75	81.71	84.68
xERTE	18.89	12.73	21.09	31.96	87.31	84.20	90.28	**91.22**	74.52	70.30	78.58	80.13
TLogic	19.77	12.23	21.67	**35.62**	76.49	74.02	78.91	79.17	**82.29**	**78.62**	**86.04**	**87.01**
TANGO	19.22	12.19	20.42	32.81	62.39	59.04	64.69	67.75	50.08	48.30	51.41	52.76
Timetraveler	**20.23**	**14.14**	**22.18**	31.17	**87.72**	**84.55**	**90.87**	91.20	78.65	75.15	82.03	83.05

multi-step setting (time filter)	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	**37.82**	**27.86**	**42.14**	**57.50**	**29.03**	**19.52**	**32.66**	**47.50**
RE-Net	37.00	27.80	40.80	54.92	27.86	18.47	31.43	46.19
CyGNet	36.12	26.66	40.28	54.54	26.01	16.69	29.59	44.43
TLogic	35.48	26.54	39.59	53.11	24.01	15.59	27.23	41.20
single-step setting (time filter)	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	42.11	31.36	47.33	**62.66**	**32.58**	**22.37**	**36.78**	**52.56**
xERTE	40.91	33.03	45.48	57.07	29.23	20.92	33.50	46.26
TLogic	**42.53**	**33.20**	**47.61**	60.29	29.59	20.42	33.60	48.05
TANGO	36.77	27.29	40.84	55.09	28.35	19.10	31.88	46.27
Timetraveler	40.83	31.90	45.43	57.59	29.13	21.29	32.54	43.92

We run the experiments on a system with one Nvidia TITAN RTX (24 GB) GPU, 512 GB Memory, and an Intel Xeon Silver 4208 CPU with 16 cores (32 threads).

To eliminate the four problems described in Section 3, we follow the evaluation protocol from Section 4: We report results on time-aware filter settings for single-step and multi-step settings, use the dataset versions (a), and report the results with the validation set usage option (b). We show aggregated results (mean MRR and Hits@k across all test samples) for the seven models for the datasets GDELT, YAGO, WIKI, ICEWS14, and ICEWS18 in Table 2. The upper part for each dataset contains results in multi-step setting, and the lower part in single-step setting, where models with results for single-step prediction should not be benchmarked against methods with results of multi-step prediction. We mark the best result for each dataset for each setting in **bold**. For completeness and comparability to related work, the supplementary material reports results on raw and static filter settings. In addition, the supplementary material contains tables with information on the reproducibility of the results that have been reported by the original works [2, 3, 6, 7, 8, 11, 12]. Figure 1 shows the MRR for three selected datasets (ICEWS18, WIKI, and GDELT) over test timestamps (snapshots) for different evaluation settings. In the following, we will discuss important insights.

Single-step and Multi-step setting: Table 2 shows the difference in scores for single- vs. multi-step setting: Overall, scores for single-step setting are higher than for multi-step setting. This is especially visible for the two models (TLogic and RE-GCN) that run in both settings, but also true for the other results. Figure 1(a) - (d) shows the MRR (in %) over snapshots in multi-step setting (left)

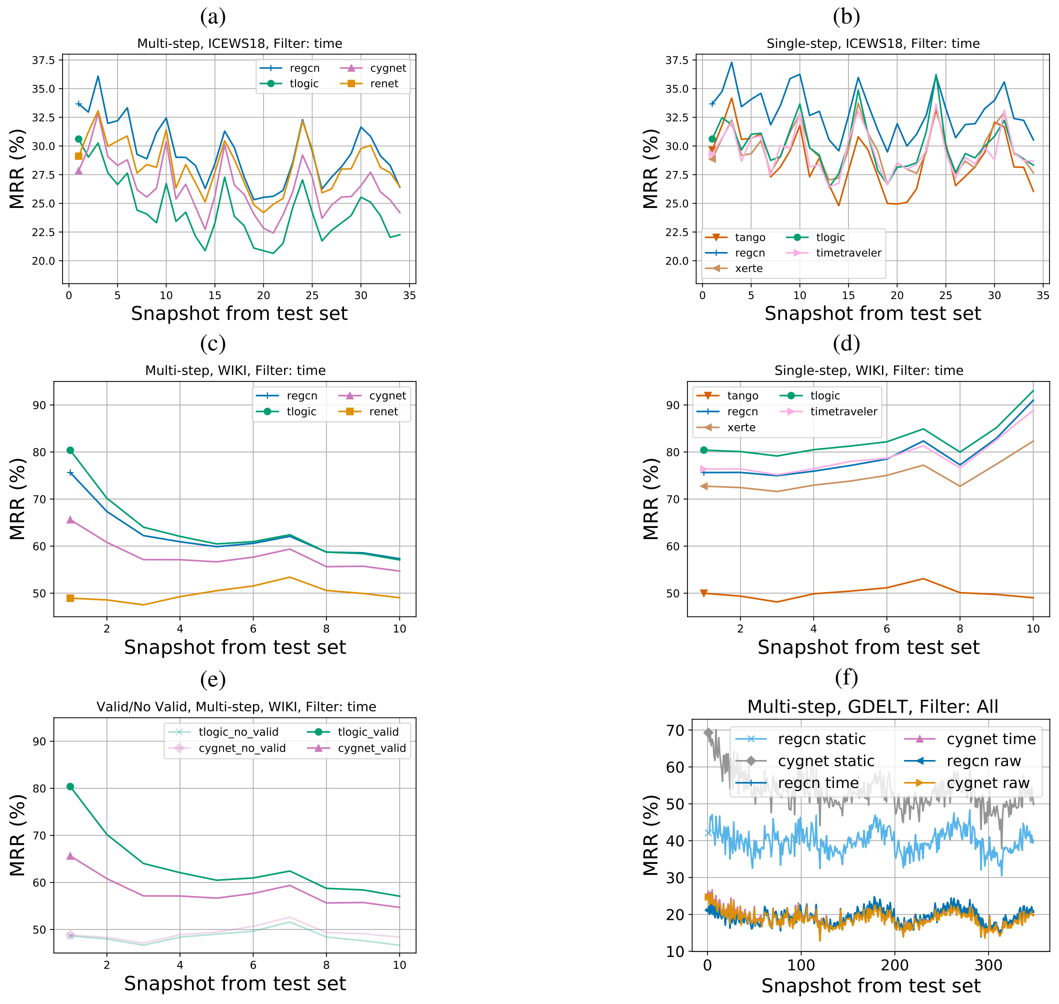

Figure 1: MRR (in %) over snapshots from test set (one snapshot is one timestamp) per method. (a)-(d): Datasets ICEWS18 (a),(b) and WIKI (c),(d) for multi-step prediction (left) and single-step prediction (right); (e): Comparison of using the validation set vs. not using it during testing for dataset WIKI; (f) Comparison of different filter settings for dataset GDELT.

and single-step setting (right).[4] The figure exhibits that the MRR for multi-step prediction has a decreasing trend with increasing timestamps, in contrast to single-step prediction, which shows no such decreasing trend. This is especially visible for the WIKI dataset in a single-step setting, which displays an increasing tendency for the MRR with increasing timestamps for the four best-performing methods. The results reflect the statement from Section 3.2, that multi-step prediction is more challenging, and uncertainty accumulates with increasing number of forecasted timesteps, as the models can only leverage information from their own forecasts. Thus, benchmarking models for multi-step prediction against single-step prediction is only fair for the first timestamp.

Validation Set Usage: In Figure 1(e), we show the MRR (in %) over snapshots in multi-step setting for TLogic and CyGNet[5], when using the validation set for testing (Section 3.4, option (b)) vs. not using the validation set for testing (option (a)) for the dataset WIKI.[6] The figure exhibits a difference in MRR between the two settings for each model, especially in the first two snapshots with a difference in MRR of > 30 for TLogic. This difference is caused by the information gap between the last training timestamp and the first testing timestamp. For the case of WIKI, the number

[4]The supplementary material shows results for ICEWS14, YAGO, and GDELT.

[5]The two models that run per default in multi-step setting, validation set option (a) from Section 3.4.

[6]The supplementary material shows results for YAGO, GDELT, ICEWS14, and ICEWS18.

of timestamps in the validation set is $num_{valid} = 11$. The difference decreases with increasing timestamps, because, due to the multi-step setting, there is also a rising information gap when feeding the validation set. Thus, using the information from the validation set for testing and avoiding the information gap is crucial for fair comparison among models.

Filter Settings: Figure 1(f) shows the MRR (in $\%$) over snapshots in multi-step setting, exemplary for CyGNet and RE-GCN for the dataset GDELT, computed with raw, static, and time-aware filter setting, as described in Section 3.1[7]. It exhibits a large difference in MRR for static filter setting, vs. raw setting or time-aware filter setting, especially for CyGNet. This is also visible for aggregated results: Where CyGNet does not have the highest MRR scores on any dataset for time-aware filter settings (see Table 2), it has the highest MRR scores on all five datasets in static filter setting (see supplementary material). The static filter setting filters out all triples that have ever appeared from the corrupted triples, ignoring the time validity, and does not count a prediction of these triples as error. Thus, for a given query, if a model predicts entities that have appeared in this triple at an earlier timestep, this will not be considered erroneous, even if the predicted fact is not true in the timestep of question. The model will potentially be assigned a higher static filter score than if it would predict previously unseen facts. Thus, the static filter setting favors models that predict repeated facts.

To summarize, we can see that no model shows the best results across all datasets. This evidence remarks the importance of fairly comparing models on different benchmarks. In this section, we stressed the clear differences in result scores for single-step and multi-step prediction. In addition, we pointed out that the usage of the validation set during testing does lead to substantially higher test scores. We also showed the significant influence of the filter setting used for score computation.

6 Conclusion

Summary: In this work, we examined the evaluation of TKG Forecasting models. We uncovered inconsistencies that strongly influence the experimental results and thus lead to distorted comparisons among models. We described these inconsistencies caused by different dataset versions, different filter settings, the absence of comparability of single-step and multi-step prediction, and the different practices of using the validation set during testing. To address these problems, we formed a unified evaluation protocol from reasonable evaluation settings and re-evaluated state-of-the-art methods. We illustrated the importance of a consistent evaluation by showing the effect of different evaluation settings on the results. Our work aims at establishing a unified evaluation protocol, stimulating discussions on the evaluation, and raising the community's awareness of experimental issues, with the goal of advancing the research field of TKG Forecasting.

Limitation of this study: Due to computational infeasibility, we could not conduct multiple repeats for each experiment run[8]. Even with one repetition per run, we experienced significant computation times for many models, e.g., multiple days to weeks for the dataset GDELT; thus, multiple repetitions per model and dataset were not possible. Adding multiple repetitions to the evaluation would have further improved the robustness of our results, which are nonetheless obtained under a unified and reproducible protocol.

Future Work: In future work, we aim to extend the proposed evaluation protocol to: First, evaluate the full predicted graph for methods that can predict full graphs (e.g., RE-Net), instead of exclusively focusing on link prediction. This could be based on graph similarity or computing a percentage of correctly predicted triples. Second, evaluate the change of the predicted graph snapshots over time to analyze if the predictions evolve and if they are able to capture time information. This could be done by comparing the predictions at different time steps. Third, include more fine-grained evaluation to answer what properties the models learned and what they did not. This could, for example, be done using the framework KGxBoard [28], which breaks down the performance measure (e.g., Hits@10) over individual data subsets.

Acknowledgments and Disclosure of Funding

We warmly thank Federico Errica for his time and very valuable feedback.

[7]The supplementary material shows results for YAGO, WIKI, ICEWS14, and ICEWS18.

[8]One experiment run: A one time training of a model with a given setting on a specific dataset.

References

[1] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 3462–3471. [Online]. Available: http://proceedings.mlr.press/v70/trivedi17a.html

[2] Z. Han, P. Chen, Y. Ma, and V. Tresp, "Explainable subgraph reasoning for forecasting on temporal knowledge graphs," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=pGIHq1m7PU

[3] H. Sun, J. Zhong, Y. Ma, Z. Han, and K. He, "Timetraveler: Reinforcement learning for temporal knowledge graph forecasting," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 8306–8319. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.655

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. [Online]. Available: https://doi.org/10.1109/TNN.2008.2005605

[5] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Trans. Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009. [Online]. Available: https://doi.org/10.1109/TNN.2008.2010350

[6] W. Jin, M. Qu, X. Jin, and X. Ren, "Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 6669–6683. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.541

[7] Z. Li, X. Jin, W. Li, S. Guan, J. Guo, H. Shen, Y. Wang, and X. Cheng, "Temporal knowledge graph reasoning based on evolutional representation learning," in *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, and T. Sakai, Eds. ACM, 2021, pp. 408–417. [Online]. Available: https://doi.org/10.1145/3404835.3462963

[8] Z. Han, Z. Ding, Y. Ma, Y. Gu, and V. Tresp, "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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 8352–8364. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.658

[9] Z. Li, S. Guan, X. Jin, W. Peng, Y. Lyu, Y. Zhu, L. Bai, W. Li, J. Guo, and X. Cheng, "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)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 290–296. [Online]. Available: https://aclanthology.org/2022.acl-short.32

[10] Z. Li, X. Jin, S. Guan, W. Li, J. Guo, Y. Wang, and X. Cheng, "Search from history and reason for future: Two-stage reasoning on temporal knowledge graphs," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 4732–4743. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.365

[11] Y. Liu, Y. Ma, M. Hildebrandt, M. Joblin, and V. Tresp, "Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium*

on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022. AAAI Press, 2022, pp. 4120–4127. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20330

[12] C. Zhu, M. Chen, C. Fan, G. Cheng, and Y. Zhang, "Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 4732–4740. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16604

[13] T. Liao, R. Taori, I. D. Raji, and L. Schmidt, "Are we learning yet? a meta review of evaluation failures across machine learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=mPducS1MsEK

[14] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Relational Representation Learning Workshop (R2L 2018), NeurIPS, Montréal, Canada*, 2018. [Online]. Available: http://arxiv.org/abs/1811.05868

[15] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=HygDF6NFPB

[16] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, "Knowledge graph embedding for link prediction: A comparative analysis," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 2, pp. 14:1–14:49, 2021. [Online]. Available: https://doi.org/10.1145/3424672

[17] Z. Han, G. Zhang, Y. Ma, and V. Tresp, "Time-dependent entity embedding is not all you need: A re-evaluation of temporal knowledge graph completion models under a unified framework," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 8104–8118. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.639

[18] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, W. Burgard and D. Roth, Eds. AAAI Press, 2011. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3659

[19] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 2787–2795. [Online]. Available: https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

[20] Z. Han, Y. Ma, Y. Wang, S. Günnemann, and V. Tresp, "Graph hawkes neural network for forecasting on temporal knowledge graphs," in *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*, D. Das, H. Hajishirzi, A. McCallum, and S. Singh, Eds., 2020. [Online]. Available: https://doi.org/10.24432/C50018

[21] J. Brownlee, *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[22] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, and M. Ward, "ICEWS Coded Event Data," 2015. [Online]. Available: https://doi.org/10.7910/DVN/28075

[23] A. García-Durán, S. Dumančić, and M. Niepert, "Learning sequence encoders for temporal knowledge graph completion," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 4816–4821. [Online]. Available: https://aclanthology.org/D18-1516

[24] W. Jin, M. Qu, X. Jin, and X. Ren, "Recurrent event network: Autoregressive structure inference over temporal knowledge graphs," *arXiv preprint arXiv:1904.05530*, 2019, preprint version.

[25] F. Mahdisoltani, J. A. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *CIDR*, 2015.

[26] J. Leblay and M. W. Chekol, "Deriving validity time in knowledge graph," in *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds. ACM, 2018, pp. 1771–1776. [Online]. Available: https://doi.org/10.1145/3184558.3191639

[27] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone, 1979–2012," in *ISA annual convention*. Citeseer, 2013, pp. 1–49.

[28] H. Widjaja, K. Gashteovski, W. B. Rim, P. Liu, C. Malon, D. Ruffinelli, C. Lawrence, and G. Neubig, "Kgxboard: Explainable and interactive leaderboard for evaluation of knowledge graph completion models," *CoRR*, vol. abs/2208.11024, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.11024

A Supplementary Material

A.1 Additional Information on Experimental Settings

In the following, we describe experimental settings in general, and subsequently special settings for each model, if they are different from the default settings or otherwise noteworthy.

For each model, we log the predicted scores for each quadruple from the test set D_{test}, in both directions (subject and object prediction) in a python dictionary. The dictionary contains as keys the query $(s, r, ?, t)$ or $(?, r, o, t)$, and as values the scores predicted by the model for each entity $v \in \mathcal{V}$ to belong to that query. After running all experiments on all datasets, settings, and methods, we compute the ranks (MRR, Hits@k, (with $k = 1, 3, 10$)) as described in Section 3.1 of the main paper based on these dictionaries.

If not stated otherwise, we use the hyperparameter settings (including the number of training epochs) reported in the respective papers. For each model, if the published source code provides the option to manually set a random seed, we did set the same seed. If the option was not explicitly provided in the published source code or could be reached with small (≤ 5 lines) modifications, during training we did not validate the models on the same filter setting as we did test them on. Instead, we validated on the default filter setting (please see details for each model below). The reason is that, in a preliminary experiment on selected models, we found that the filter setting has only a small influence on the best validation epoch (i.e., different settings for validation lead more often than not to the same best epoch). Thus, we are confident that this did not significantly influence the final results. Due to issues regarding memory consumption and very high computing time, we were not able to conduct the experiments for the dataset ICEWS05-15 for the better part of models and thus excluded this dataset from our experiments.

RE-Net [6] We run RE-Net in multi-step setting only, because the published source code does not provide the option to set the single-step option in the arguments. Also, when we asked via e-mail and GitHub issue about how to conduct the implementation of the single-step option, unfortunately, we did not receive a concrete reply. We train the models on the static filtered MRR, following the training procedure provided in the source code. Due to GPU memory issues with the dataset GDELT, we run the model for this specific dataset on CPU, which leads to a very long runtime (> 50 days of training). For the source code to be able to run on CPU, we have to conduct modifications to the source code. We run all other experiments for RE-Net on GPU.

RE-GCN [7] We train the models on the raw MRR, following the training procedure provided in the source code. While RE-GCN is originally also evaluated on relation prediction, we exclude this setting in our study, as the other models do not support relation prediction. We run RE-GCN in both settings, single-step and multi-step.

CyGNet [12] We run CyGNet on multi-step setting only, because non-trivial modifications in the source code would be necessary to run in single-step setting. Unfortunately, the authors did not reply to our question on the concrete implementation of multi-step setting. We train one model for each setting, raw, static, and time-aware filter. For testing, instead of allowing the model to only use the information from triples in the train set, we allow to also take into account the triples in the validation set. For this, we slightly adjust the original source code: In our version, the historical vocabulary (copy-sequences) now includes all timesteps from train and validation set, instead of only the timesteps from the train set. Please see Figure 2 for an overview of the change in testing scores for time-aware filter setting, when using the validation set (our modification) versus not using the validation set (original) during testing.

TLogic [11] For the datasets ICEWS14 and ICEWS18 we use the hyperparameters as described in the paper. The hyperparameter that changes across datasets is the window size w. According to the authors, the higher the window size, the better the performance, but also the higher the memory need. This means, that generally, the smaller (and less dense) the graph, the higher the window size can be in regard to memory usage. The datasets YAGO, WIKI, and GDELT have not been evaluated in the original paper. For the small dataset YAGO we set the window size to $w = 0$, which includes all past timesteps. For WIKI and GDELT we experience memory issues when using the machines described in Section 5 (main paper), and even when using a machine with 2 TB Memory. Integrating instructions kindly provided by the authors, for the datasets WIKI and GDELT we can circumvent these memory issues by decreasing the rule length to $l = \{1, 2\}$, instead of $l = \{1, 2, 3\}$. In addition, for WIKI and GDELT we set the window size to $w = 200$, the value reported by Liu et al. [11] for the larger dataset ICEWS18.

For the multi-step setting, we modify the published source code. Instead of allowing the model to only apply the rules based on occurrences of quadruples in the train set, we allow to also take into account the quadruples in the validation set. We modify the highest timestep for the rule application to be the highest timestep from the validation set, instead of the highest timestep from the training set. In addition, for datasets ICEWS18, WIKI and GDELT, we implement the option to set the window size of $w = 200$ also for multi-step prediction (instead of using all quadruples from training and validation set). Please see Figure 2 for an overview of the impact of using the validation set (our modification) versus not using the validation set (original) during testing (rule application) on testing scores for time-aware filter setting.

TimeTraveler [3] We run TimeTraveler only in single-step setting, because, as kindly confirmed by the authors, non-trivial modifications in the source code would have been necessary to run in multi-step setting. For GDELT, no hyperparameters were specified in the original paper. We use the same hyperparameters as for WIKI, because this dataset is the most similar in size. TimeTraveler is capable of doing inductive link prediction for future timesteps, i.e., prediction of triples with previously unseen nodes. We do not specifically evaluate this capability, as it is not in the scope of our study.

xERTE [2] We run xERTE only in single-step setting, because, as kindly confirmed by the authors, non-trivial modifications in the source code would have been necessary to run in multi-step setting. In the paper, hyperparameters are not specified for the datasets WIKI and GDELT. We use the hyperparameters as specified for the ICEWS18 dataset because ICEWS18 is most similar in size. For each epoch during training, we log the validation results for raw, static, and time-aware filter settings. We run separate testing for the three filter settings, where we select the trained model from the best training epoch for the respective setting. Please note, that in most cases, the best epoch was the same across settings[9]. We experienced a very long training time ($>$ 30 days) for xERTE on the GDELT dataset.

TANGO [8] We only run TANGO in single-step setting, because non-trivial modifications in the source code would be necessary to run in multi-step setting. Unfortunately, the authors did not reply to our question on the possibility of implementing the multi-step setting, nor on the question of how to realize the long-horizontal forecasting experiment they report in their paper. For GDELT, no hyperparameters were specified in the original paper. We use the same hyperparameters as for WIKI, because it is the most similar in size. We train one model for each setting, raw, static, and time-aware filter. TANGO is capable of doing inductive link prediction for future timesteps, i.e., prediction of

[9]The best epoch e_{best} was $e_{best} = 8$ for 9 out of 12 cases (3 settings across 4 experiment runs), with $variance_{bestepoch} = 0.52$.

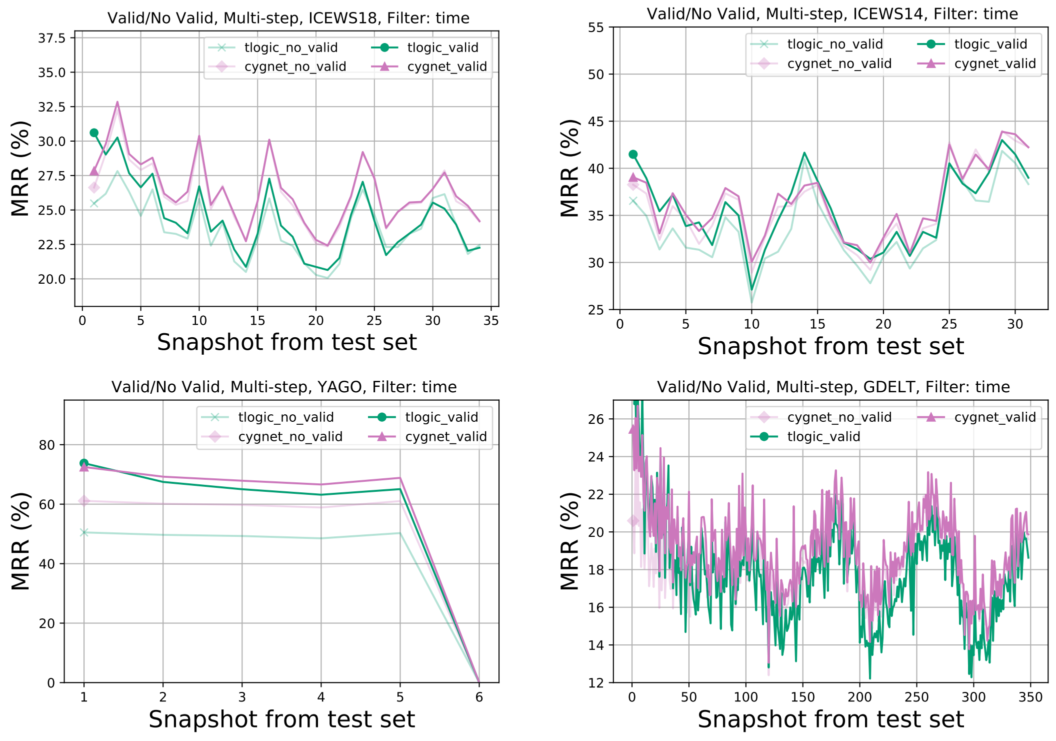

Figure 2: MRR (in $\%$) over snapshots from test set per method for datasets ICEWS18 (top left), ICEWS14 (top right), YAGO (bottom left), and GDELT (bottom right) for methods CyGNet and TLogic for multi-step prediction in time-aware filter setting. Each Subfigure shows the MRR when leveraging the information from the validation set during testing, vs. when not using it. Figures for static and raw setting are available upon request.

triples with previously unseen nodes. We do not specifically evaluate this capability, as it is not in the scope of our study.

A.2 Additional Experiment Results

A.2.1 Usage of the Validation Set

Figure 2 shows the performance of the methods TLogic and CyGNet on all datasets in multi-step setting, when not leveraging the information from the validation set D_{valid} (option a), versus leveraging D_{valid} (option b) (see Section 3 in main paper) during testing. Please note that the drop in scores for the sixth timestep on the YAGO dataset is due to the dataset only having two samples in this snapshot, and all models performing bad on these two samples.

Table 3: Experiment results for multi-step and single-step prediction with datasets GDELT, YAGO, WIKI, ICEWS14, and ICEWS18. Results for single-step prediction should not be compared to results for multi-step prediction. We report mean reciprocal rank (MRR), and Hits@k (H@k), with $k = 1, 2, 3$ in static filter setting (static filter).

multi-step setting (static filter)	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	39.90	32.38	43.12	53.27	77.81	75.36	78.95	82.30	66.16	64.73	66.80	68.49
RE-Net	41.45	34.68	43.84	54.04	64.99	63.44	65.31	67.73	52.18	51.27	52.31	53.83
CyGNet	**53.01**	**46.52**	**56.62**	**64.14**	**84.57**	**83.93**	**84.76**	**85.52**	**69.00**	68.38	**69.26**	**70.02**
TLogic	35.77	30.00	37.80	46.68	71.39	71.10	71.27	71.87	68.54	**68.52**	68.54	68.55

single-step setting (static filter)	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	40.26	32.84	**43.35**	**53.55**	83.81	81.05	85.18	88.99	81.83	79.96	82.92	85.05
xERTE	29.38	24.62	32.05	39.00	**90.44**	**89.90**	90.82	**91.28**	78.73	77.65	79.58	80.42
TLogic	37.62	30.47	41.11	51.78	79.10	78.97	79.06	79.28	**87.18**	**87.16**	**87.19**	**87.20**
TANGO	**41.03**	**35.12**	42.88	52.25	67.88	66.95	67.85	69.47	52.46	52.12	52.58	53.06
Timetraveler	28.62	23.90	29.29	37.33	90.26	89.37	**90.99**	91.24	82.60	82.19	82.73	83.27

multi-step setting (static filter)	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	48.14	40.12	51.78	**63.51**	41.23	33.58	44.28	55.81
RE-Net	48.21	41.52	50.81	61.13	42.88	36.19	45.36	55.97
CyGNet	**53.10**	**47.83**	**55.47**	62.85	**47.97**	**42.70**	**50.05**	**57.81**
TLogic	51.15	46.37	53.28	60.66	43.10	38.37	45.22	52.20

single-step setting (static filter)	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	52.91	44.97	56.91	**67.91**	45.57	37.49	49.04	**61.23**
xERTE	46.22	40.12	50.09	58.61	37.30	31.14	40.65	49.91
TLogic	**57.76**	**52.86**	**60.55**	66.86	**47.66**	**42.07**	**50.62**	58.27
TANGO	50.71	45.20	52.90	61.58	41.88	34.68	44.90	55.32
Timetraveler	48.09	41.62	51.00	60.17	36.09	30.10	38.37	47.25

A.2.2 Static and Raw Filters

Tables 3 and 4 report results on static filter setting and raw filter setting, for the five datasets GDELT, YAGO, WIKI, ICEWS14, and ICEWS18. Although we do not encourage evaluation on these settings, we have added the results for reasons of completeness and comparability. Figure 3 shows the MRR over test timestamps (snapshots), for multi-step and single-step prediction for the three remaining datasets (ICEWS14, YAGO, and GDELT) that have not been shown in the main paper. Please note that the drop in scores for the sixth timestep on the YAGO dataset is due to the dataset only having two samples in this snapshot, and all models performing badly on these two samples.

Table 4: Experiment results for multi-step and single-step prediction with datasets GDELT, YAGO, WIKI, ICEWS14, and ICEWS18. Results for single-step prediction should not be compared to results for multi-step prediction. We report mean reciprocal rank (MRR), and Hits@k (H@k), with $k = 1, 2, 3$ in raw setting (raw).

multi-step setting (raw)	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	19.21	11.96	20.47	33.34	**58.20**	**47.94**	**65.47**	75.73	39.96	31.74	44.57	54.01
RE-Net	**19.27**	**11.97**	**20.51**	**33.63**	46.49	37.74	52.13	61.55	31.00	25.12	33.76	41.29
CyGNet	18.68	11.41	19.90	32.81	54.88	43.52	61.54	**77.77**	37.58	28.37	42.72	54.08
TLogic	17.35	10.88	18.57	30.05	52.36	42.22	60.46	69.90	**40.58**	**32.49**	**45.67**	**54.72**
single-step setting (raw)	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	19.31	11.99	20.62	33.56	63.07	51.96	71.00	82.24	51.51	40.90	58.21	69.49
xERTE	18.51	12.26	20.76	31.75	64.70	52.07	74.48	**87.31**	53.15	42.07	61.15	71.93
TLogic	19.30	11.69	21.23	**35.31**	57.88	46.56	67.10	77.52	53.38	42.18	**61.41**	**72.09**
TANGO	18.80	11.69	20.04	32.52	49.02	40.61	55.04	63.01	30.72	25.07	33.74	40.42
Timetraveler	**19.77**	**13.52**	**21.84**	31.02	**64.83**	**52.23**	**74.57**	**87.31**	**53.41**	**42.27**	61.35	71.98

multi-step setting (raw)	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	**37.23**	**27.14**	**41.59**	**57.31**	**27.77**	**17.94**	**31.46**	**46.99**
RE-Net	36.27	26.75	40.31	54.68	26.59	16.87	30.29	45.67
CyGNet	35.41	25.86	39.72	54.42	25.05	15.53	28.66	43.83
TLogic	34.85	25.70	39.05	52.92	23.09	14.51	26.30	40.76
single-step setting (raw)	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-GCN	41.29	30.22	46.66	**62.45**	**31.02**	**20.38**	**35.41**	**51.94**
xERTE	40.06	31.74	45.01	56.91	27.95	19.23	32.46	45.84
TLogic	**41.56**	**31.81**	**46.95**	60.08	28.16	18.59	32.35	47.50
TANGO	36.04	26.29	40.34	54.88	27.03	17.42	30.79	45.74
Timetraveler	40.08	30.88	44.89	57.44	28.04	19.85	31.63	43.59

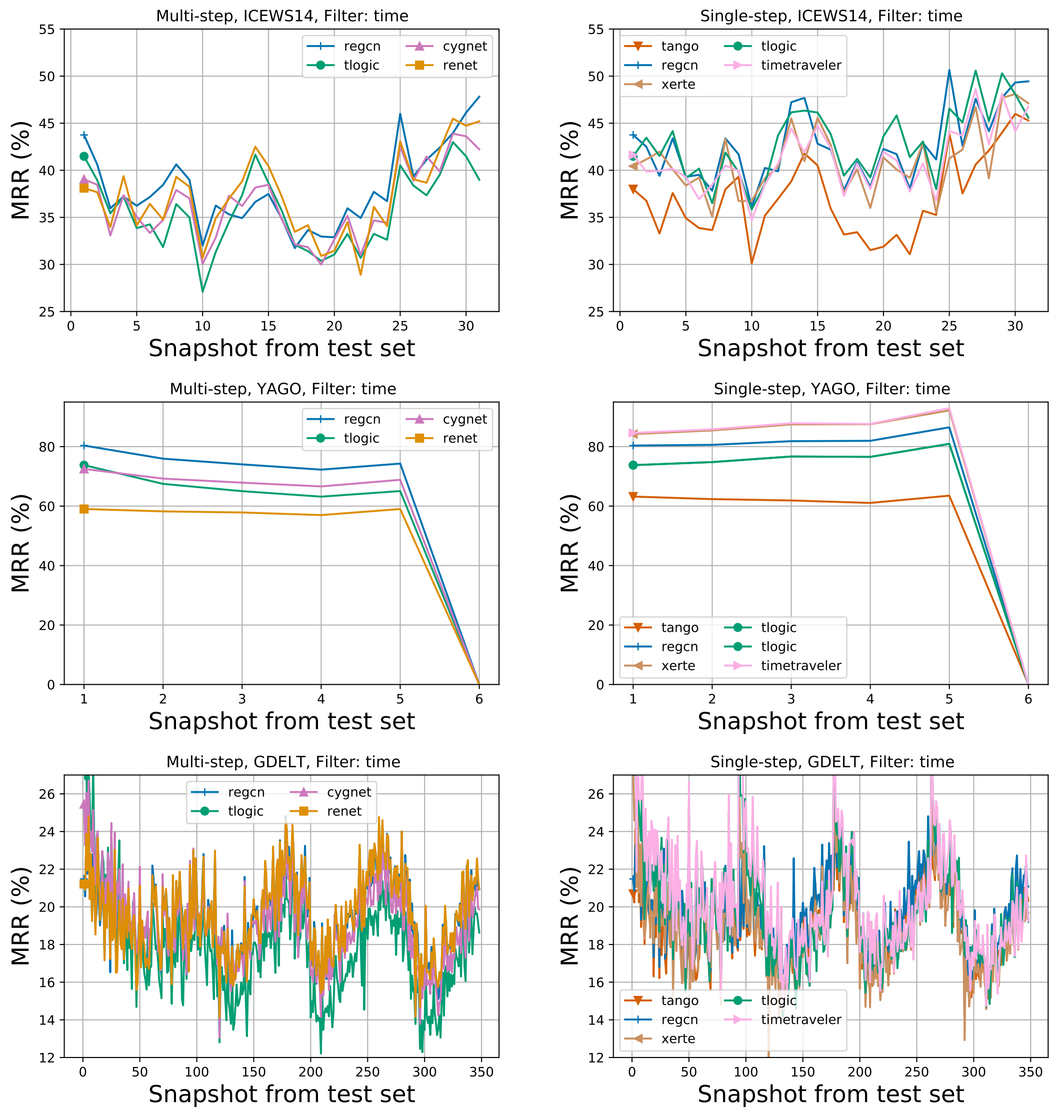

Figure 3: MRR (in %) over snapshots from test set (one snapshot is one timestamp) per method for datasets ICEWS14 (top), YAGO (middle) and GDELT (bottom) for multi-step prediction (left) and single-step prediction (right). Figures for static and raw setting are available upon request.

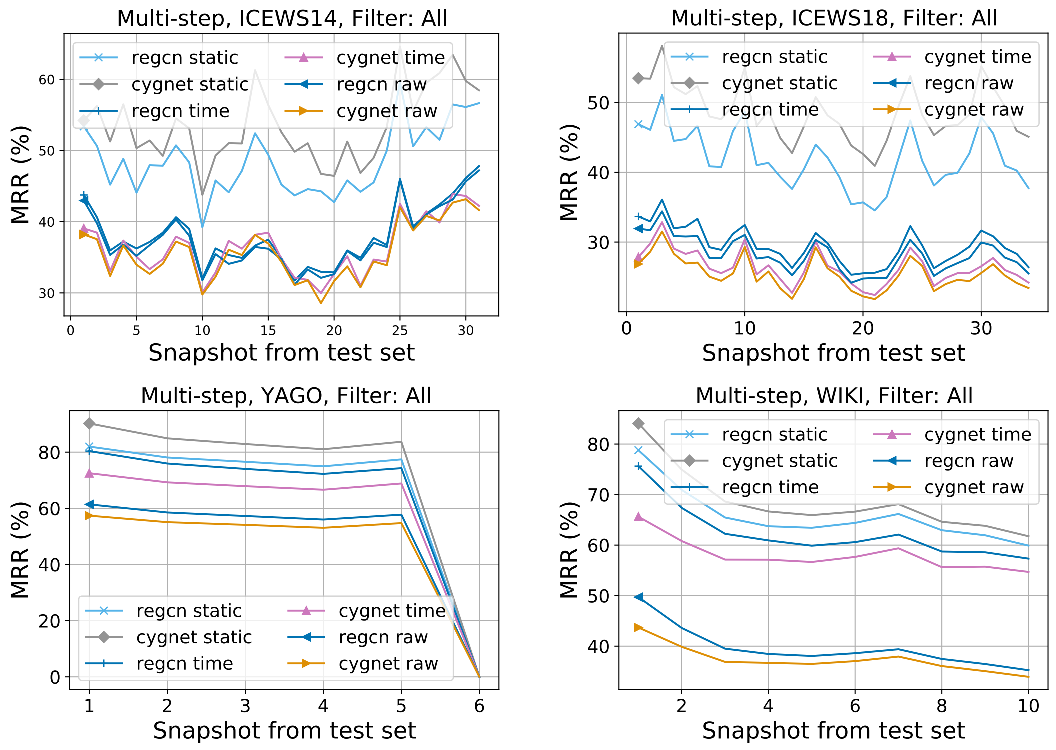

Figure 4: MRR (in %) over snapshots from test set (one snapshot is one timestamp) for methods CyGNet and RE-GCN on all filter settings (raw, static, time-aware filter) for datasets ICEWS14 (top left), ICEWS18 (top right), YAGO (bottom left) and WIKI (bottom right) for multi-step prediction. Figures for other methods and single-step prediction are available upon request.

Table 5: Experiment result Consistency: Difference Δ in reported scores (MRR and Hits) on multi-step and single-step setting, for time-aware filter, static filter, and raw setting on the datasets ICEWS14, ICEWS18, with $\Delta_{\text{Score}} = \text{Score}_{\text{Original Paper}} - \text{Score}_{\text{This Work}}$. An entry **n.r.** means that the result was **n**ot **r**eported by the original paper in this setting. An entry **d.v.** means that a different **d**ataset **v**ersion was used in the original paper, and thus results cannot be compared.

RE-GCN	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step								
time-aware	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	0.55	0.03	0.94	1.53	-0.26	-0.12	-0.29	-0.44
single-step								
time-aware	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	0.21	0.64	-0.06	0.02	-0.47	-0.38	-0.68	-0.48

RE-Net	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step								
time-aware	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
static	d.v.	d.v.	d.v.	d.v.	0.05	n.r.	0.11	-0.17
raw	d.v.	d.v.	d.v.	d.v.	0.03	n.r.	-0.02	-0.1
single-step	not computed							

CyGNet	Different usage of validation set: option (a) as described in section 3.4 in main paper. Thus, results are not comparable.
Tlogic	Different usage of validation set: option (a) as described in section 3.4 in main paper. Thus, results are not comparable.

xERTE	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step	not reported							
single-step								
time-aware	d.v.	d.v.	d.v.	d.v.	0.08	0.11	0.01	0.22
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.

TANGO	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step	not reported							
single-step								
time-aware	d.v.	d.v.	d.v.	d.v.	0.62	0.41	0.73	1.24
static	d.v.	d.v.	d.v.	d.v.	2.68	3.19	2.56	1.74
raw	d.v.	d.v.	d.v.	d.v.	0.56	0.35	0.61	1.18

TimeTraveler	ICEWS14				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step	not reported							
single-step								
time-aware	d.v.	d.v.	d.v.	d.v.	0.85	0.76	0.92	0.91
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.

A.2.3 Result Consistency

Tables 5 and 6 show the difference Δ of scores (MRR and Hits) reported by the authors of the original papers to the results from our experiments (as reported in Table 2, 3, and 4), if computable. As we show in Table 1, various differences in evaluation settings exist, and not all papers report results on all datasets, thus it is not possible to compute the differences for all datasets and settings for each method.

Table 6: Experiment result Consistency: Difference Δ in reported scores (MRR and Hits) on multi-step and single-step setting, for time-aware filter, static filter, and raw setting on the datasets GDELT, YAGO, and WIKI, with $\Delta_{\text{Score}} = \text{Score}_{\text{Original Paper}} - \text{Score}_{\text{This Work}}$. An entry **n.r.** means that the result was **n**ot **r**eported by the original paper in this setting. An entry **d.v.** means that a different **d**ataset **v**ersion was used in the original paper, and thus results cannot be compared.

RE-GCN	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step												
time-aware	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	-0.06	-0.04	-0.07	-0.15	0.07	n.r.	0.15	0.21	-0.12	n.r.	-0.14	-0.13
single-step												
time-aware	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	0	0	-0.01	0.03	0	n.r.	0.17	-0.17	0.02	n.r.	0.08	0.04
RE-Net	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step												
time-aware	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
static	-1.03	n.r.	-0.44	-0.34	0.17	n.r.	0.32	0.35	12.98[a]	n.r.	13.32[b]	14.25[c]
raw	0.33	n.r.	0.05	0.26	0.32	n.r.	0.58	0.38	-0.13	n.r.	1.79	-0.02
single-step	not computed											
CyGNet	Different usage of validation set: option (a) as described in section 3.4 in main paper. Thus, results are not comparable.											
Tlogic	Different usage of validation set: option (a) as described in section 3.4 in main paper. Thus, results are not comparable.											
xERTE	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step												
single-step												
time-aware	n.r.	n.r.	n.r.	n.r.	d.v.	d.v.	d.v.	d.v.	n.r.	n.r.	n.r.	n.r.
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
TANGO	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step												
single-step												
time-aware	n.r.	n.r.	n.r.	n.r.	0.95	1	0.5	1.04	2.96	3.22	2.43	2.7
static	n.r.	n.r.	n.r.	n.r.	0.46	0.1	0.54	1.23	1.59	-0.6	1.26	2.4
raw	n.r.	n.r.	n.r.	n.r.	0.47	0.29	0.38	0.73	1.81	1.26	2.01	2.75
TimeTraveler	GDELT				YAGO				WIKI			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
multi-step												
single-step												
time-aware	n.r.	n.r.	n.r.	n.r.	-0.26	0.34	-0.91	-0.93	-3.15	-2.19	-4.54	-4.03
static	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.
raw	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.	n.r.

[a] The results we find for RE-Net on the Wiki dataset in static filter setting are not consistent with the results reported by the authors of the original paper. However, our results are consistent with the results that the authors of CyGNet [12] report for RE-Net in this setting on this dataset, where we have $\Delta_{\text{MRR}} = \text{MRR}_{\text{CygNet reports for ReNet}} - \text{MRR}_{\text{This Work}} = -0.21$, $\Delta_{\text{H@3}} = -0.24$, and $\Delta_{\text{H@10}} = 0.08$.

[b] see footnote *a*.

[c] see footnote *a*.

A.3 Related Work, Additional Information

Following up on the short introduction in Section 2 of the main paper, here we provide more details on each TKG Forecasting model.

Graph Neural Networks (GNNs): A large group of models leverages a GNN [4, 5] in combination with a sequential approach to integrate the structural and sequential information. RE-Net [6] applies an autoregressive architecture. It builds on a two-step approach to learn the temporal dependency from a sequence of graphs and the local structural dependency from the neighborhood. The occurrence of a fact is modeled as a probability distribution conditioned on the temporal sequence of past snapshots. RE-Net can predict full graphs. RE-GCN [7] also models the sequence of the Knowledge Graph snapshots recurrently. For this, it combines a convolutional graph neural network with a sequential Neural Network model. Further, RE-GCN introduces a static graph constraint to take into account additional information like entity types. TANGO [8] bases on neural ordinary differential equations to model the temporal sequences combined with a GNN to capture the structural information. In addition, the authors introduce a stochastic jump method to incorporate stochastic events, i.e., triples appearing or disappearing over time. TANGO can learn continuous representations of entities and relations. xERTE [2] bases on so-called temporal relational attention mechanisms. To answer a query, it extracts query-relevant subgraphs. Further, it computes and propagates attention scores to identify the relevant evidence in the subgraphs, using a modified time-aware version of a message passing. CEN [9] integrates a Convolutional Neural Network which can handle evolutional patterns of different lengths via an easy-to-difficult curriculum learning strategy, which learns these evolutional patterns from short to long. The model can learn in an online setting, and thus can adapt to changes in evolutional patterns over time.

Reinforcement Learning: CluSTeR [10] introduces a two-step process: First, a Reinforcement learning agent, working with randomized beam strategy, searches and induces clue paths related to a given query. Second, an adopted GNN and sequence method models temporal information among the clues to find answers to a query. TimeTraveler [3] leverages a reinforcement learning model based on temporal paths. Starting from the query's subject node, the agent traverses outgoing edges across graph snapshots. For this, TimeTraveler samples actions according to transition probabilities, which are based on dynamic embeddings of the query, the path history, and the candidate actions. TimeTraveler uses a time-shaped reward based on Dirichlet distribution. The model is able to predict in the inductive setting by integrating a newly introduced Inductive Mean representation mechanism.

Rule-Based Approaches: TLogic [11], a symbolic framework, learns so-called temporal logic rules via temporal random walks, traversing edges through the graph backward in time. Rules contain nodes, edges, and timesteps. TLogic applies the rules to events that happened prior to the query. For scoring the answer candidates, it takes into account the rules' confidence as well as time differences.

Other: CyGNet [12] predicts future facts purely based on the appearance of historical facts. For this, to answer a query, it first computes each entity's embedding vector. Further, using these embeddings, it computes entity probabilities by combining predictions from a so-called "copy mode" that computes probabilities for historical events based on the repetition of facts in history and a "generation mode" that computes probabilities for every entity.

A.4 Dataset Statistics

Following up on the description in Section 3 (main paper), please see table A.4 for dataset statistics for dataset version (a), as reported by Li et al. [7].

Table 7: Dataset Statistics for dataset versions a, as reported by Li et al. [7].

Dataset	#Nodes	#Rels	#Train	#Valid	#Test	Time Interval
ICEWS14	6869	230	74845	8514	7371	24 hours
ICEWS18	23033	256	373018	45995	49995	24 hours
ICEWS0515	10094	251	368868	46302	46159	24 hours
GDELT	7691	240	1734399	238765	305241	15 minutes
YAGO	10623	10	161540	19523	20026	1 year
WIKI	12554	24	539286	67538	63110	1 year

A.5 Checklist for Benchmark Experiments on TKG Forecasting

In the following, we provide a checklist for benchmark experiments on TKG Forecasting.

- Are the datasets used the same version for all models? Check e.g., number of triples in train, validation, test set.
- Are the hyperparameters set as reported in the papers?
- Is the single-step/ multi-step setting consistent across models?
- Is the validation set used during testing?
- Are you sure that the test set is not leaked during training?
- Does the model predict in both directions, $(s, r, ?, t)$ and $(?, r, o, t)$?
- Are evaluation scores computed based on time-aware filtered setting? Is the implementation to compute the evaluation scores consistent across all models?