Towards Predicting Future Time Intervals on Temporal Knowledge Graphs

Roxana Pop
University of Oslo
roxanap@uio.no

Egor V. Kostylev
University of Oslo
egork@uio.no

Abstract

Temporal Knowledge Graphs (TKGs), a temporal extension of Knowledge Graphs where facts are contextualized by time information, have received increasing attention in the temporal graph learning community. In this short paper we focus on TKGs where the temporal contexts are time intervals, and address the time prediction problem in the forecasting setting. We propose both a system architecture for addressing the task and a benchmarking methodology.

1 Introduction

Temporal Knowledge Graphs (TKGs) are a means of representing temporally contextualized knowledge. While (static) Knowledge Graphs are a set of facts showing relations between entities, e.g. $(Alice, Employed, CompanyA)$, TKGs are a set of facts associated with the time context in which they are true, e.g. $(Alice, Employed, CompanyA)@[2018, 2023]$. The time context can be time points (Point-based TKGs) or time intervals (Interval-based TKGs, ITKGs in short) [1].

One of the main machine learning tasks on TKGs is time prediction, where the goal is to predict the temporal context for a given triple (a time interval in the case of ITKGs). A setting commonly considered in TKG prediction tasks is the forecasting setting [2], where the model takes as input historical data and makes predictions about the future. As we will highlight in the next section, systems have been developed for time prediction on ITKGs, but to the best of our knowledge not in the forecasting setting. Our main contributions are as follows.

- We propose a system architecture, TKGMixer, for addressing the time prediction task on ITKGs in the forecasting setting. Our architecture is inspired by GraphMixer [3]—a simple yet effective architecture introduced for graph learning.
- We describe a methodology for constructing relevant benchmarks, by specifying the evaluation metric and how to construct a training ITKG and test examples from a given ITKG.

This paper discusses work in progress and does not cover experimental results.

2 Related Work

Many systems have been introduced for prediction tasks on TKGs as well as temporal graphs in general, addressing a series of tasks and settings[4]. Most of the TKG learning systems operate on point-based TKGs[1], with notable exeptions being TA-DISTMULT [5], HyTE [6], TIMEPLEX [7], TIME2BOX [8], and TILP [9]. Of these methods, TIMEPLEX [7] and TIME2BOX [8] are the most relevant systems to our current work, as they address the time prediction task on ITKGs, i.e. they can predict time intervals for given facts. Yet, they do not do so in the forecasting setting[2]. They are designed to output time intervals, and they both use the greedy coalescing method [7] to construct such intervals. This method works as follows. First, assuming the model can provide a score for

Temporal Graph Learning Workshop @ NeurIPS 2023, New Orleans.

a temporal fact $(s, r, o)@t$, the probability $P(t \mid s, r, o)$ is computed from the scores across the set of candidate timepoints by using $softmax$. Then, the predicted interval is constructed iteratively starting from the timepoint with the highest probability and greedily adding timepoints to the left or to the right until the total probability of the interval exceeds a given treshold.

While not introduced for TKGs but for other types of temporal graphs, GraphMixer[3] is a relevant architecture to this work as we draw inspiration from some of its components. GraphMixer is a conceptually simple yet effective architecture which consists of mainly three modules: a link encoder that creates node representations based on the nodes' recent history of interactions, a node encoder which aggregates neighborhood information, and a link classifier which uses the output from the encoders. It also employs a simple time encoding function which was shown empirically[3] to lead to more stability in training as opposed to trainable time encoding alternatives.

3 Problem Formalisation

In this section, we formalise the problem we want to address: prediction of the next time interval for a given triple. We generally adopt the formalisation of this prediction function $f_{\mathsf{next\text{-}int}}$ from Pop and Kostylev [1], and simplify it by concentrating on the setting with integer timeline $\mathbb{Z}$ and no type triples. Additionally to the existing formalisation[1] we also to deal with unbounded intervals.

Let $\mathcal{R}$ be a finite set of *relations*, and let $\mathcal{E}$ be an infinite set of *entities*, also known as *constants*. We are interested in (non-empty and possibly unbounded) intervals over *timepoints* $\mathbb{Z}$ of one of the following forms, for timepoints t_1, t_2: $[t_1, t_2]$ with $t_1 \leq t_2$, $[t_1, \infty)$, $(-\infty, t_2]$, and $(-\infty, \infty)$. When the exact form is not important, we may write '$\langle s_1$' instead of '$[t_1$' and '$(-\infty$' and '$s_2\rangle$' instead of '$t_2]$' and '$\infty)$'. A *fact* is a triple of the form (e_1, r, e_2), where $e_1, e_2 \in \mathcal{E}$ and $r \in \mathcal{R}$, and a *temporal fact* is $\lambda@\rho$, where λ is a fact and ρ is an interval. An *interval-based temporal knowledge graph* (*ITKG*) is a set of temporal facts. An ITKG G is in *normal form* if there are no $\lambda@\rho_1$ and $\lambda@\rho_2$ in G with $\rho_1 \cap \rho_2 \neq \emptyset$; in what follows, we silently concentrate on ITKGs in normal form. For an ITKG G, let $\mathsf{Rels}(G)$ and $\mathsf{Consts}(G)$ denote the relations and entities appearing in G, respectively, and let $\mathsf{Sig}(G) = \mathsf{Rels}(G) \cup \mathsf{Const}(G)$. The *past subgraph* $G_{\leq t}$ of an ITKG G for a timepoint t contains every fact of the form $\lambda@\langle s_1, \min(t, s_2)]$ for which there is a fact $\lambda@\langle s_1, s_2\rangle \in G$. In this case, we call G a *temporal completion* of $G_{\leq t}$.

Given an ITKG $G_{\leq t}$ and a triple λ over $\mathsf{Sig}(G_{\leq t})$, and assuming existence of the most probable temporal completion G of $G_{\leq t}$, the *next interval function* $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda)$ returns

- $\lambda@[t+1, s_2\rangle$ if there is $\lambda@\langle s_1, s_2\rangle \in G$ with $s_1 \leq t < s_2$,
- $\lambda@[t_1, s_2\rangle$ if $\lambda@[t_1, s_2\rangle \in G$, $t_1 > t$, and t_1 is the smallest timepoint with this property,
- $\emptyset$ otherwise.

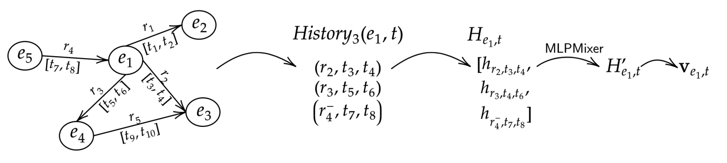
4 Method

We want to learn an approximation of function $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda)$. To do so, we first model the probability $P(t^+ \mid G_{\leq t}, \lambda, t)$ for each future time point $t^+ > t$, given the graph up to the current time point t and a triple of interest λ, as a soft boolean classifier trained on appropriately prepared examples extracted from a training ITKG, and then use this probability to construct the intervals.

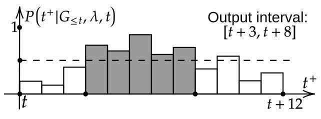
4.1 Classifier Architecture

Our goal is to design a simple architecture. In order to model the probability $P(t^+ \mid G_{\leq t}, \lambda, t)$, we construct representations for the fact λ (entities and relation) and for the time of interest t^+, concatenate these representations and pass them as input to an MLP g with $sigmoid$ activation. We do this for both (e_1, r, e_2) and the fact with the inverse role r^-, i.e. (e_2, r^-, e_1), and compute the average. Thus, assuming that we have $\mathbf{v}_{e_1,t}$, $\mathbf{v}_{e_2,t}$, $\mathbf{v}_r$ and $\mathbf{v}_{e_1,t}$ be vectorial representations of e_1, e_2, r and r^-, and that z is a function which encodes time points into vectors, we have $P(t^+|G_{\leq t}, (e_1, r, e_2), t) = (g([\mathbf{v}_{e_1,t}||\mathbf{v}_r||\mathbf{v}_{e_2,t}||z(t - t^+)]) + g([\mathbf{v}_{e_2,t}||\mathbf{v}_{r^-}||\mathbf{v}_{e_1,t}||z(t - t^+)]))/2$.

For the time encoding function $z : \mathbb{R} \to \mathbb{R}^{d_{\mathsf{time}}}$ we use $z(t) = cos(t\boldsymbol{\omega})$ for the vector $\boldsymbol{\omega} = [\alpha^{-(i-1)/\alpha}]_{i=1}^{d_{\mathsf{time}}}$ with $\alpha = \sqrt{d_{\mathsf{time}}}$. The definition of the time encoding function z follows from



(a) History-based entity representation



(b) Time interval construction

Figure 1: *a)* The representation $\mathbf{v}_{e_1,t}$ is constructed based on the history of interactions of e_1 formed of the most recent 3 facts (assume $t_1 \leq t_2 < t_3 \leq ...t_{10} \leq t$, hence why $(e_1, r_1, e_2)@[t_1, t_2]$ is not considered). *b)* The time interval $[t+3, t+8]$ is constructed when $M = 12$ (the probabilities $P(t^+|G_{\leq t}, \lambda, t)$ are considered for $t^+ \in [t, t+12]$). The dashed line symbolizes the thresholding.

GraphMixer[3] which we chose due to its empirical training stability[3]. Our entity representation computation is also based on $GraphMixer$[3], on the link-encoder module to be more exact, which we chose due to its simplicity. In order to adapt it for *ITKGs* we encoded two time points instead of one and also encoded the relations. We cover the details of our adaptation in the next paragraphs. For relations, since we consider the set of relations fixed, we simply learn embeddings, that is, for relation r, $\mathbf{v}_r$ is a learnable (for the whole ITKG) embedding of a fixed size d_{rel}.

For a given entity e and time point t, we construct their vector representation $\mathbf{v}_{e,t}$ using the history of interactions, as shown in Figure 1a. We concentrate on the temporal facts that e has been recently part of, as the facts most relevant for predictions about e after t. Let, for a hyper-parameter $k \in \mathbb{N}$, $G^k_{e,t}$ be the maximal set of at most k most recent temporal facts in $G_{\leq t}$ that mention e; here, a temporal fact $\lambda@\langle s_1, t_2] \in G$ is more recent than a fact $\lambda'@\langle s'_1, t'_2]$ if $t_2 > t'_2$. Then, let *k-bounded history* $\mathsf{History}_k(e, t)$ *of e relative to t* be the set

$$\{(r, s^*_1, t_2) \mid (e, r, e')@\langle s_1, t_2] \in G^k_{e,t}\} \cup \{(r^-, s^*_1, t_2) \mid (e', r, e)@\langle s_1, t_2] \in G^k_{e,t}\},$$

where r^- is the inverse of a role r, and s^*_1 is s_1 if $s_1 \in \mathbb{Z}$ and the smallest integer mentioned in $G_{\leq t}$ otherwise. Then, we encode each $(r, t_1, t_2) \in \mathsf{History}_k(e, t)$ by first encoding relation or inverse r, and timepoints t_1 and t_2 into vectors $\mathbf{v}_r$, $\mathbf{v}_{t_1}$ and $\mathbf{v}_{t_2}$ of fixed sizes, and then concatenating them to a vector $\mathbf{h}_{r,t_1,t_2} = [\mathbf{v}_r || \mathbf{v}_{t_1} || \mathbf{v}_{t_2}]$, where $\mathbf{v}_{t_1} = z(t - t_1)$ and $\mathbf{v}_{t_2} = z(t - t_2)$. We then stack the representations $\mathbf{h}_{r,t_1,t_2}$ together, in the descending order of t_2, t_1, r (using an arbitrary but fixed order for r), to obtain a matrix $\mathbf{H}_{e,t} \in \mathbb{R}^{d \times k}$, where $d = d_{\mathsf{rel}} + 2d_{\mathsf{time}}$; if $|\mathsf{History}_k(e, t)| < k$ then the missing values in $\mathbf{H}_{e,t}$ are set to 0. Once we have $\mathbf{H}_{e,t}$, we process it with an $\mathsf{MLPMixer}$ [10] transformation (as in GraphMixer [3]) to obtain matrix $\mathbf{H}'_{e,t}$ of the same dimensions.[1] More concretely, for *layer normalisation* function $\mathsf{LayerNorm}$ [11] and activation function GELU [12], we first apply applying a 2-layer MLP to each column of $\mathbf{X} = \mathsf{LayerNorm}(\mathbf{H}_{e,t})$ to obtain matrix $\mathbf{U}$ by setting $\mathbf{U}_{*,j} = \mathbf{X}_{*,j} + \mathbf{W}_2\mathsf{GELU}(\mathbf{W}_1\mathbf{X}_{*,j})$ for each $j = 1, \ldots, k$. Then we apply a 2-layer MLP to each row of $\mathbf{Y} = \mathsf{LayerNorm}(\mathbf{U})$ by setting $(\mathbf{H}'_{e,t})_{i,*} = \mathbf{Y}_{i,*} + \mathbf{W}_4\mathsf{GELU}(\mathbf{W}_3\mathbf{Y}_{i,*})$ for each $i = 1, \ldots, d$. Matrices $\mathbf{W}_1 \in \mathbb{R}^{d_{\mathsf{col}} \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{1 \times d_{\mathsf{col}}}$, $\mathbf{W}_3 \in \mathbb{R}^{d_{\mathsf{row}} \times d}$ and $\mathbf{W}_4 \in \mathbb{R}^{1 \times d_{\mathsf{row}}}$ are learnable matrices with fixed d_{col} and d_{row}. Finally, we compute the representation $\mathbf{v}_{e,t}$ of e at time t by averaging $\mathbf{H}'_{e,t}$ along the rows.

4.2 Classifier Training

To train our classifier, we assume that it receives a training ITKG G_{train} as input, from which we generate n positive and n negative examples $((G_{\leq t_i}, \lambda_i, t_i, t^+_i), y_i)$, for $i = 1, \ldots 2n$, where $G_{\leq t_i}$ is the past subgraph of G_{train} for a timepoint t_i, λ_i and $t^+_i > t_i$ are a fact and a timepoint to predict for, and $y_i \in \{0, 1\}$ is the golden truth. We propose to sample positive and negative examples in the following way.

To generate positive examples, we first sample several timepoints between the minimal and maximal integer timepoints mentioned in G_{train}. Then, for each such t, we construct the past graph $G_{\leq t}$ and sample several facts λ such that $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda) \neq \emptyset$. Finally, we sample several timepoints t^+ from $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda)$, thus arriving to a set of positive examples $((G_{\leq t}, \lambda, t, t^+), 1)$. We select the numbers of samples to ensure that the overall number of generated examples is n.

[1]We also plan to experiment with other ways to get $\mathbf{H}'_{e,t}$, such as using $\mathbf{H}_{e,t}$ directly or applying an MLP to it.

As for negative examples, we generate two types of such examples: examples with facts that are represented among positive ones, and examples with facts not mentioned in G_{train}. For the first type, we sample $n/2$ positive examples from the constructed set and, for each $((G_{\leq t}, \lambda, t, t^+), 1)$ of these samples, take $((G_{\leq t}, \lambda, t, t^-), 0)$, where $t^- > t$ is sampled from the outside of all ρ with $\lambda@\rho \in G_{\mathsf{train}}$, with higher probability closer to the endpoints of $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda)$; if $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda) = [t+1, \infty)$ (i.e., when such a t^- does not exist) we repeat with another sample of a positive example. For the second type, we again sample $n/2$ positive examples. We sample triples λ over $Sig(G)$ but which do not appear in G, and we sample time points t^+ between the minimal and maximal time points in the training set.

Having these examples, we can train our classifier in a standard supervised manner using the log loss $\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(P(t_i^+ \mid G_{\leq t_i}, \lambda_i, t_i)) + (1-y_i)\log(1 - P(t_i^+ \mid G_{\leq t_i}, \lambda_i, t_i))\right]$.

4.3 Time Interval Construction

Our approach to construct our model of $f_{\mathsf{next\text{-}int}}(G_{\leq t}, \lambda)$ from the predicted probabilities $\bar{P}(t^+ \mid G_{\leq t}, \lambda, t)$ with $t^+ > t$ is inspired by the greedy coalescing method [7]. However, contrary to this method, we do not use *softmax* for candidate timepoints, thus avoiding an assumption that the timepoints are mutually exclusive; moreover, our method is able to output $\emptyset$.

Formally, we proceed as follows. First, we convert the soft binary classification $\bar{P}(t^+ \mid G_{\leq t}, \lambda, t)$ for each individual t^+ to a hard binary classification using a threshold, which is a hyperparameter. Then we compute the prediction of the resulting classifier for each $t^+ \in [t+1, M]$, where $M \in \mathbb{N}$ is deduced from the training data. Finally, we set the prediction as $\emptyset$ if all the predictions are negative; otherwise as the interval $[t_1^+, s_2^+\rangle$, where t_1^+ is the smallest t^+ with positive prediction, and s_2^+ is either the largest timepoint t' in $[t_1^+, t + t_{\max}]$ such that the prediction for all timepoints in $[t_1^+, t']$ is positive if $t' < t_{\max}$, or ∞ otherwise. Figure 1b exemplifies our method for $M = 12$.

5 Benchmarking

To the best of our knowledge, there are no benchmarks addressing our problem. Hence, in this section we describe a method how to construct such a benchmark from a given ITKG G (eg. YAGO11k, Wikidata12k[6]). Such a benchmark is a triple $(G_{\mathsf{train}}, D_{\mathsf{test}}, f_{\mathsf{eval}})$ that consists of a training ITKG G_{train} (which can be processed by a model as desired for training), a test dataset D_{test} of examples of the form (x, y) with input x and golden-truth output y, and an evaluation metric f_{eval} which computes a single score for the benchmark by comparing the golden-truth outputs y from D_{test} with the predicted outputs $\bar{y}$ when the model of interest is applied on the corresponding inputs x.

We construct G_{train}, D_{test}, and f_{eval} from G as follows. First, we pick a time point t_{split} so that $|G_{\leq t_{\mathsf{split}}}|$ is roughly 70% of $|G|$ and let G_{train} be $G_{\leq t_{\mathsf{split}}}$. Then, we let D_{test} consist of two types of examples, both based on the ITKG $G' = G \setminus G_{\mathsf{train}}$ (such split is in spirit of other benchmarks for the forecasting setting [13]). First, it has a larger proportion of examples of the form of an input-output pair $(G'_{\leq t}, \lambda), f_{\mathsf{next\text{-}int}}(G'_{\leq t}, \lambda))$, where t is random timepoint sampled between t_{split} and the maximal integer mentioned in G', and λ is a random fact mentioned in G'. Second, a smaller proportion of examples are of the form $(G'_{\leq t}, \lambda'), \emptyset)$, where t is taken from a random example of the first type, and λ' is a random fact not mentioned in G' over $\mathsf{Sig}(G)$.

Finally, as f_{eval} we average $gaeIOU$ [8] scores for the golden-truth y and predicted $\bar{y}$ intervals for input x across all examples (x, y) in D_{test}. Before applying this metric we assume a transformation such that $+\infty$ is replaced with a very large timepoint (ensuring the metric is always defined).

6 Conclusion and Future Work

In this paper we introduced a system for future time interval prediction on ITKGs by extensively adapting the GraphMixer[3] architecture and providing an alternative method to the greedy coalescing methodology for the addressed task. Yet, we mainly discussed the transductive setting. We believe the introduced model should have inductive capabilities (as the representation does not depend on the constants, only on their interractions). We therefore plan to also design an inductive benchmark for this task. We will then empirically evaluate our system in both the transductive and inductive setting.

References

[1] Roxana Pop and Egor V. Kostylev. Inductive future time prediction on temporal knowledge graphs with interval time. In *The International Workshop on Neural-Symbolic Learning and Reasoning*, volume 3432, pages 233–240. CEUR-WS.org, 2023.

[2] Julia Gastinger, Timo Sztyler, Lokesh Sharma, Anett Schuelke, and Heiner Stuckenschmidt. Comparing apples and oranges? on the evaluation of methods for temporal knowledge graph forecasting. In Danai Koutra, Claudia Plant, Manuel Gomez Rodriguez, Elena Baralis, and Francesco Bonchi, editors, *Machine Learning and Knowledge Discovery in Databases: Research Track*, pages 533–549, Cham, 2023. Springer Nature Switzerland.

[3] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The International Conference on Learning Representations (ICLR)*, 2023.

[4] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21(1), jan 2020.

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

[6] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *The 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011. Association for Computational Linguistics, 2018.

[7] Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. Temporal Knowledge Base completion: New algorithms and evaluation protocols. In *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3733–3747, 2020.

[8] Ling Cai, Krzysztof Janowicz, Bo Yan, Rui Zhu, and Gengchen Mai. Time in a Box: Advancing Knowledge Graph Completion with Temporal Scopes. K-CAP '21, pages 121–128, New York, NY, USA, December 2021. Association for Computing Machinery.

[9] Siheng Xiong, Yuan Yang, Faramarz Fekri, and James Clayton Kerce. TILP: Differentiable learning of temporal logical rules on knowledge graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP Architecture for Vision. In *The Advances in Neural Information Processing Systems (NeurIPS)*, pages 24261–24272, 2021.

[11] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[12] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[13] Namyong Park, Fuchen Liu, Purvanshi Mehta, Dana Cristofor, Christos Faloutsos, and Yuxiao Dong. Evokg: Jointly modeling event time and network structure for reasoning over temporal knowledge graphs. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 794–803, New York, NY, USA, 2022. Association for Computing Machinery.