Leveraging Pre-trained Language Models for Time Interval Prediction in Text-Enhanced Temporal Knowledge Graphs

Duygu Sezen Islakoglu[1][0000−0002−9441−6458], Melisachew Wudage Chekol[1][0000−0002−5286−8587], and Yannis Velegrakis[2,1][0000−0001−6332−0296]

[1] Utrecht University, The Netherlands
{d.s.islakoglu, m.w.chekol, i.velegrakis}@uu.nl
[2] University of Trento, Italy

Abstract. Most knowledge graph completion (KGC) methods rely solely on structural information, even though a large number of publicly available KGs contain additional temporal (validity time intervals) and textual data (entity descriptions). While recent temporal KGC methods utilize time information to enhance link prediction, they do not leverage textual descriptions or support inductive inference (prediction for entities that have not been seen during training).

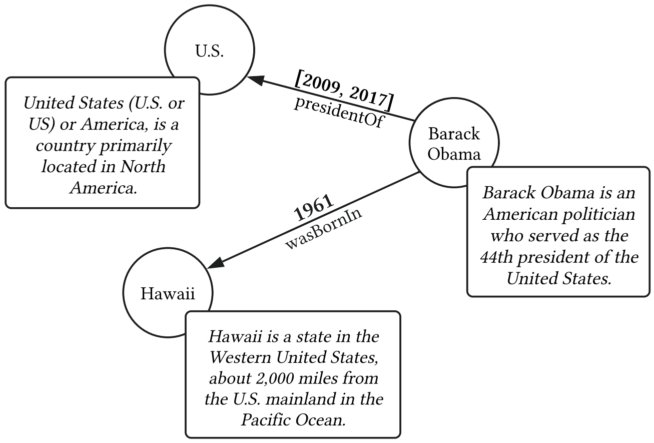
In this work, we propose a novel framework called TEMT that exploits the power of pre-trained language models (PLMs) for temporal KGC. TEMT predicts time intervals of facts by fusing their textual and temporal information. It also supports inductive inference by utilizing PLMs. In order to showcase the power of TEMT, we carry out several experiments including time interval prediction, both in transductive and inductive settings, and triple classification. The experimental results demonstrate that TEMT is competitive with the state-of-the-art, while also supporting inductiveness.

Keywords: Knowledge Graph Completion · Temporal Knowledge Graphs · Pre-trained Language Models

1 Introduction

Knowledge graphs are often incomplete, meaning that some elements of the facts are not available. To this end, knowledge graph completion (KGC) methods aim at finding missing links between the entities. Most of these studies focus on static knowledge graphs where the graph remains unchanged over time [15]. However, in real life, facts are not always valid throughout time, but only in specific time periods. For instance, presidents of a country are valid only throughout their term.

To model this, triples in a knowledge graph may have validity time intervals associated with them. This additional information converts a triple into a quadruple form ⟨subject, relation, object, time interval⟩, e.g. ⟨Obama, presidentOf, U.S., [2009, 2017]⟩. A graph consisting of a set of such temporal facts -quadruples- is referred to as a *Temporal Knowledge Graph* (TKG).



United States (U.S. or US) or America, is a country primarily located in North America.

Barack Obama is an American politician who served as the 44th president of the United States.

Hawaii is a state in the Western United States, about 2,000 miles from the U.S. mainland in the Pacific Ocean.

Fig. 1. An example of a text-enhanced temporal knowledge graph.

Alike traditional KGs, temporal knowledge graphs suffer inherently from incompleteness due to their dynamic nature. In particular, the temporal information (validity time interval) is often missing after an automatic knowledge graph construction. To this end, the temporal knowledge graph completion (TKGC) task aims to predict a missing quadruple element, such as time interval prediction ⟨s, r, o, ?⟩. The time interval prediction task is useful for temporal question answering, automatic TKG construction, and verification of TKGs that have temporal constraints [35].

(Temporal) knowledge graphs completion methods [36] can benefit from textual descriptions of entities and relations since they contain valuable information regarding semantic relationships across entities. For instance, in a description of an entity, there may be a reference to some other entity, despite the absence of any type of relationship (edge) in the knowledge graph among them. Furthermore, by considering the semantics of the descriptions, one may gain insight into the validity time of the facts. As an example, if an entity description contains elements from a certain century or a period like Renaissance, facts involving that entity may be valid for that period. Figure 1 illustrates a temporal knowledge graph with entities that have textual descriptions associated with them.

The availability of textual descriptions in knowledge graphs provides an excellent opportunity for exploiting the benefits that both knowledge graphs and language models can offer. Recent works has shown that language models store real-world knowledge in their parameters and can potentially be used as knowledge graphs [27, 3], and that textual information can improve link prediction for static knowledge graphs [20, 38]. Moreover, it has been shown that entity descriptions and pre-trained language models can model facts that involve unseen entities (inductiveness) [9, 34]. Supporting inductive reasoning is crucial since most real-world knowledge graphs are often continuously extended with new entities. Unfortunately, most temporal knowledge graph completion mechanisms are transductive, which means that they can only perform predictions on the entities they have already seen during training, i.e., are part of the training set.

We provide a novel temporal knowledge graph completion framework called TEMT (**T**ext **E**ncoder **M**eets **T**ime)[3] that combines the available textual and temporal information for **time interval prediction**. TEMT is able to predict time intervals for unseen entities by leveraging a pre-trained language model. We extend the two commonly used datasets, i.e., YAGO11k and Wikidata12k [8], with textual descriptions. In addition, we create new train/valid/test splits for experiments on time interval prediction in inductive setting. Our experiments show that TEMT is competitive with state-of-the-art and able to reason on unseen entities (even when both the subject and object of a quadruple are unseen in training).

2 Preliminaries

A temporal knowledge graph (TKG) is a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$ where $\mathcal{E}, \mathcal{R}, \mathcal{T}$ are sets of entities, relations, time points and $\mathcal{Q}$ represents the set of quadruples (or temporal facts) in the format $\langle$subject s, relation r, object o, time interval $t_I\rangle$ where $s, o \in \mathcal{E}$, $r \in \mathcal{R}$, $t_I = [t_{start}, t_{end}]$ and t_{start}, $t_{end} \in \mathcal{T}$. Note that t_I can also be a single time point if t_{start}=t_{end}.

Temporal knowledge graphs can be grouped into two types: event TKGs and interval-based TKGs. The former refers to TKGs in which $t_{start} = t_{end}$ for every quadruple. On the other hand, interval-based TKGs, which are the focus of this paper, represent TKGs where each quadruple has a validity time interval $t_I = [t_{start}, t_{end}]$. An interval is called left-open if t_{start} is unknown, right-open if t_{end} is unknown, and closed interval if both the start and end points are known. The format of time points depends on the chosen time granularity, such as years, months, or days.

Text-enhanced temporal knowledge graphs are TKGs where each entity and relation is associated with a name and each entity has some natural language text that describes its meaning. This additional information provides a context and attaches a semantic meaning to the facts, which can be informative for predicting the validity time interval of the fact. More formally, a text-enhanced temporal knowledge graph is a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}, \mathcal{N}, \mathcal{D})$ where $\mathcal{E}, \mathcal{R}, \mathcal{T}$ are sets of entities, relations, time points and $\mathcal{Q}$ represents the set of quadruples, $\mathcal{N}$ denotes the set of entity and relation names, and $\mathcal{D}$ denotes the set of entity descriptions. An example for text-enhanced TKG is given in Figure 1.

We can split $\mathcal{Q}$ into three disjoint sets as train, validation, and test sets. Formally, $\mathcal{Q} = \mathcal{Q}_{train} \cup \mathcal{Q}_{val} \cup \mathcal{Q}_{test}$ where $\mathcal{Q}_{train}$ represents the set of train quadruples, $\mathcal{Q}_{val}$ represents the set of validation quadruples and $\mathcal{Q}_{test}$ represents the set of test quadruples. Similarly, we can specify the set of entities as $\mathcal{E}_{train}$, $\mathcal{E}_{val}$ and $\mathcal{E}_{test}$.

[3] The datasets and the source code are available at https://github.com/duyguislakoglu/TEMT.

Problem Statement Time interval prediction is the task of predicting the validity time interval of a triple. More formally, given a quadruple ⟨s, r, o, ?⟩ with unknown validity time interval, the objective is to predict a time interval t_I. We can reformulate the question as follows: given training and validation sets, $\mathcal{Q}_{train}$ and $\mathcal{Q}_{val}$, the time interval prediction is to output t_I for each test quadruple ⟨s, r, o, ?⟩ $\in \mathcal{Q}_{test}$.

In this paper, we focus on two variants of this problem: transductive time interval prediction and inductive time interval prediction. The transductive case aims to predict t_I for each test quadruple in $\mathcal{Q}_{test}$ that does not contain any new entities, i.e., $\mathcal{Q}_{test} = \{\langle s, r, o, ?\rangle | s, o \in \mathcal{E}_{train}\}$. In other words, all entities in the test set are included in the training set, i.e., $\mathcal{E}_{test} \subseteq \mathcal{E}_{train}$. Furthermore, inductive time interval prediction aims to predict the time interval of facts where the test set contains previously unseen entities [9]. For inductive inference, we create splits as follows: for each quadruple in the test set, the subject or the object (or both) does not appear in $\mathcal{E}_{train}$. Therefore, the test quadruples $\mathcal{Q}_{test} = \{\langle s, r, o, ?\rangle | s \notin \mathcal{E}_{train}\} \cup \{\langle s, r, o, ?\rangle | o \notin \mathcal{E}_{train}\}$.

Pre-trained Language Models We exploit the representational power of pre-trained language models to capture the semantics of facts and to deal with unseen entities. Language models assign a probability to a word by taking into account the other words in a sentence and can predict the next word given a sequence of words. This can be done by learning a latent representation of words in a vector space. Moreover, models such as bidirectional encoder representations from transformers (BERT) [11] do not only consider the previous words but also take the subsequent words into account. BERT generates a contextual word embedding where the representation of a word depends on the whole context.

A pre-trained language model (PLMs) is a language model that is trained on a large text corpora including books, encyclopedias, and web data. PLMs can be used for many downstream tasks such as question-answering and text summarization. A pre-trained model, such as pre-trained BERT, can be further fine-tuned for a specific task or can be used for feature extraction of a sentence. However, since BERT is designed for word-level tasks and not optimized for sentence-level tasks, it performs poorly in semantic textual similarity tasks [28]. On the other hand, Sentence-BERT [28], a language model built on top of BERT, is explicitly trained to generate sentence embeddings where semantically similar sentences are closer in the embedding space. In the next section, we explain how Sentence-BERT can be used to generate an embedding for a triple.

3 The Framework

We materialize the idea of using pre-trained language models for temporal knowledge graph completion into a framework called TEMT (**T**ext **E**ncoder **M**eets **T**ime). It learns a scoring function that takes a quadruple ⟨s, r, o, t⟩ where t is a single time point and outputs a plausibility score. In the inference time, we

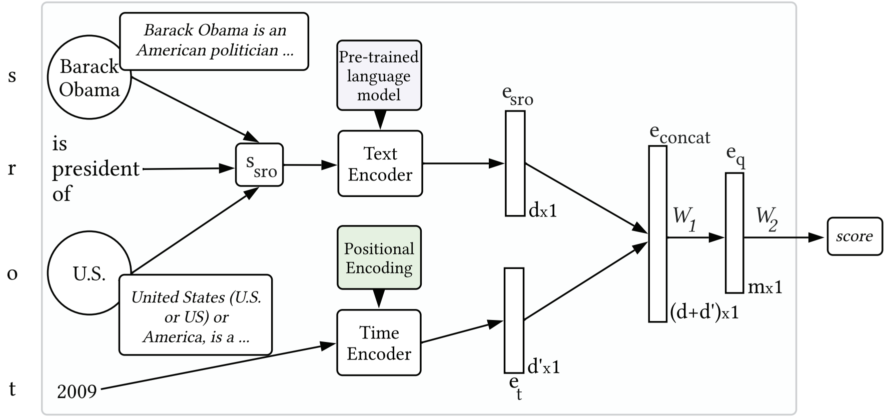


Fig. 2. Overview of TEMT's architecture for quadruple scoring: the TextEncoder produces *triple embedding* e_{sro}, and the TimeEncoder generates *time point embedding* e_t. These representations are fused to compute a plausibility score.

utilize the learned scoring function to output a time interval for a given triple ⟨s, r, o, ?⟩. Figure 2 gives an overview of the TEMT framework.

Embedding Triples with a Text Encoder The text encoder packs the names and descriptions of triple elements as a single sentence and returns a vector. As our text encoder, we leverage a pre-trained language model Sentence-BERT [28][4] to benefit from its representation power. Inspired by [38], we form a single textual sentence S_{sro} for a triple to feed Sentence-BERT.

$$S_{sro} = \mathcal{N}_s + \mathcal{N}_r + \mathcal{N}_o + (\mathcal{D}_s + \mathcal{D}_o), \tag{1}$$

where S_{sro} is a string concatenation of the names and descriptions of the entities and relations, $\mathcal{N}$ refers to names and $\mathcal{D}$ refers to entity descriptions. The text encoder then outputs $e_{sro} \in \mathbb{R}^d$ which we call *triple embedding*:

$$e_{sro} = \text{TextEncoder}(S_{sro}) \tag{2}$$

Our main motivation to leverage a language model as a text encoder is twofold. Firstly, the language model captures the interactions between the subject, relation, and the object and outputs a semantically rich contextualized embedding of the fact. Secondly, language models can model unobserved entities and therefore support inductive reasoning. Moreover, not only Sentence-BERT, but also any encoder-only model (e.g. BERT) can be used as our text encoder. This is possible by adding a special [CLS] token to the beginning of the S_{sro}. In this way, the token embedding for [CLS] captures the possible interactions between the subject, relation and object [38] and functions as a sentence embedding. Thus, TEMT is not dependent on Sentence-BERT; instead, it is model-agnostic.

[4] The name of the model used is all-mpnet-base-v2.

Embedding Time Points with a Time Encoder We use positional encoding method [33] to produce vector representations of time points. This method is widely used in (knowledge graph) question answering systems [23, 16, 39] in order to embed time. Given a time point t and a reference time point t_{min} (which is the earliest time point in the dataset), the j-th component of a *time point embedding* for t is defined as follows:

$$\text{TimeEncoder}(t, t_{min})[j] = \begin{cases} \sin\left(\frac{t-t_{min}}{10000^{i/d'}}\right) & \text{if } j = 2i \\ \cos\left(\frac{t-t_{min}}{10000^{i/d'}}\right) & \text{if } j = 2i+1 \end{cases} \tag{3}$$

where the term $t-t_{min}$ refers to the position of t relative to the earliest time point t_{min} in $\mathcal{T}$ and d' is the dimension of the *time point embedding*. Intuitively, the *time point embedding* can be thought of as a position in time. The time encoder requires a first time point t_{min} as a reference point, then the other time points will be positioned relative to this reference point. For the sake of brevity, we omit t_{min} from the time encoder function and simply write as TimeEncoder(t). The time encoder outputs $e_t \in \mathbb{R}^{d'}$ which we call *time point embedding.*

$$e_t = \text{TimeEncoder}(t) \tag{4}$$

We emphasize mainly the two properties of positional encodings: each time point corresponds to a unique vector and the vectors of close time points are closer in the vector space [16]. This enables us to model the dependencies across different time points and the notion of ordering. On the contrary, many previous works [14, 19] learn the time vectors within the same space as entities and relations, and they cannot model the time point dependencies. For instance, they learn the consecutive years independently which may not capture ordering.

Moreover, TEMT can be adapted to different time granularities. For instance, one can set months as granularity by converting the years into corresponding number of months. Lastly, in contrast to previous work, the time encoder can represent unobserved time points. Although we do not focus on temporal inductiveness in this paper, the time encoder can potentially be used for performing predictions on future or unseen time points.

Fusing Triple and Time Point Embeddings In the previous sections, we introduced two functions, namely TextEncoder and TimeEncoder, that allow us to produce embeddings of triples and time points respectively. We are now ready to discuss how these embeddings from different spaces can be combined (or fused) together. Similar to [25, 12], by treating the textual and temporal features as different modalities, TEMT combines triple and time embeddings using a multi-layer perceptron (MLP). The fusion of these two embeddings produces a representation of a quadruple. Formally, given a quadruple q, the time-aware quadruple representation e_q is obtained as follows:

$$e_q = (W_1 v_{concat} + b_1) \in \mathbb{R}^m \tag{5}$$

$$v_{concat} = [e_{sro}; e_t] \in \mathbb{R}^{d+d'} \tag{6}$$

where [;] denotes concatenation operation, $W_1 \in \mathbb{R}^{m\times(d+d')}$ and $b_1 \in \mathbb{R}^m$ denote the learnable parameters and m is the dimension of e_q where $m < (d + d')$.

Another approach for embedding a quadruple is to append the time point to the textual sentence S_{sro} in Equation (1) and feed pre-trained language model with this new sentence. This would represent all quadruple elements in a single space, which is the text space. However, it is shown that PLMs are not good at number representations [31]. Our preliminary analysis also demonstrated that language models are not good at temporal reasoning such as ordering events and interval arithmetic. This motivates the need for an external time encoder.

Parametric Quadruple Scoring Function Although most methods use a fixed distance function for scoring triples or quadruples, there are some methods such as ConvE [10] and ConvKB [24] that learn the parameters for a scoring function. Similarly, we employ a parametric scoring function to output a plausibility score for a quadruple of a given TKG:

$$f(s, r, o, t) = W_2 e_q + b_2 \tag{7}$$

where $W_2 \in \mathbb{R}^{1\times m}$ and $b_2 \in \mathbb{R}$ are learnable parameters of the final layer of the neural network. Before feeding the input to this final layer, we use ReLU [1] as an activation function.

Negative Sampling The model learns by distinguishing valid quadruples from incorrect quadruples. To this end, TEMT employs two different types of negative sampling. The first type is called *entity-corrupted negative sampling*. In this approach, the set of negative quadruples $D^-_{\langle s,r,o,t\rangle}$ is created by corrupting the subject or the object of a given quadruple $\langle s, r, o, t\rangle$ as shown below:

$$D^-_{\langle s,r,o,t\rangle} = \{\langle s', r, o, t\rangle \notin D^+ | s' \in \mathcal{E}\} \cup \{\langle s, r, o', t\rangle \notin D^+ | o' \in \mathcal{E}\}. \tag{8}$$

where $D^+ = \mathcal{Q}_{train}$ denotes the set of positive quadruples.

The second one is called *time-corrupted negative sampling* [6]. In this approach, the set of negative quadruples $D^-_{\langle s,r,o,t\rangle}$ is created by corrupting the time point of a given quadruple $\langle s, r, o, t\rangle$ as the following:

$$D^-_{\langle s,r,o,t\rangle} = \{\langle s, r, o, t'\rangle \notin D^+ | t' \in \mathcal{T}\}. \tag{9}$$

t' is sampled based on the validity time interval t_I of the given quadruple. If t_I is a right-open interval, then $t' < t_{start}$ is sampled (e.g. t'= 2006 for t_I=[2008, unknown]); if t_I is left-open interval, then $t' > t_{end}$ is sampled (e.g. t'= 1950 for t_I=[unknown, 1930]); if t_I is a closed interval, then $t' \notin [t_{start}, t_{end}]$ is randomly chosen (e.g. t'=1750 for t_I=[1800, 1810]).

Training Similar to [4], we use the following margin-based ranking loss for training:

$$\mathcal{L} = \sum_{q_p \in D^+} \sum_{q_n \in D^-_{q_p}} \max(0, f(q_n) - f(q_p) + \gamma). \tag{10}$$

where q_p is a positive quadruple, q_n is a negative quadruple, γ is the margin value and f is the scoring function from Equation (7). We train the model to give higher scores to positive quadruples (with a given margin γ) than negative quadruples. In this way, we learn a quadruple scoring function that will be used for predicting time intervals.

Inference In the previous sections, we discuss how TEMT learns a scoring function that gives plausibility score to an individual quadruple. Now, we will discuss how we go from the individual quadruple scores to a time interval. As discussed in Section 2, our main goal is to predict time interval for a given $\langle s, r, o, ?\rangle$. Given the earliest ($t^{test}_{min}$) and the latest time point (t^{test}_{max}) in the test set, we compute the plausibility score of quadruple with each time point in the interval $[t^{test}_{min}, t^{test}_{max}]$. So the list of test scores are defined as the following:

$$\mathbf{S} = \left[f(s, p, o, t) | t \in [t^{test}_{min}, t^{test}_{max}]\right] \tag{11}$$

We turn these scores into probabilities by using the softmax function.

$$\mathbf{P} = \left[P(t|s, r, o) | t \in [t^{test}_{min}, t^{test}_{max}]\right] \tag{12}$$

where

$$P(t_i|s, r, o) = \frac{exp(f(s, p, o, t_i))}{\sum_{s_j \in \mathbf{S}} exp(s_j)}. \tag{13}$$

Lastly, we use greedy-coalescing algorithm from [6] that takes probabilities **P** and outputs k time intervals as our predictions.

4 Experiments

4.1 The Datasets

We perform our experiments on two interval-based TKGs: YAGO11k and Wikidata12k [8]. YAGO11k is created from YAGO3 knowledge graph [22] by the meta-facts in the form of (#factID, occurSince, t_{start}) and (#factID, occurUntil, t_{end}) that are available in some of the facts. Wikidata12k is a subgraph of a preprocessed version of Wikidata [19] that contains facts with temporal annotations (e.g. point-in-time, start time, end time) or properties (e.g. "start time", "inception", "demolition time"). In both datasets, each fact has a time interval attached to it and each entity has at least two edges. Furthermore, ind-YAGO11k datasets and ind-Wikidata12k are inductive splits that we generate from YAGO11k and Wikidata12k. The details of the four datasets are given in Table 1.

Table 1. Dataset statistics.

Dataset	Entity	Relation	Train	Valid	Test
YAGO11k	10,623	10	16,408	2,050	2,051
Wikidata12k	12,554	24	32,497	4,062	4,062
ind-YAGO11k	10,623	10	12,330	3,726	4,453
ind-Wikidata12k	12,554	24	27,330	6,354	6,937

We enhance the datasets with the names and descriptions of entities and relations. For Wikidata12k, the entity names and descriptions are extracted from their corresponding Wikipedia pages. For YAGO11k, the entity and relation names are already available in the dataset. We extract the entity descriptions for YAGO11k from Wikipedia pages as well. For both datasets, the entity descriptions are limited to one sentence. Similar to [14, 6], we fix the time granularity as "year" and drop the months and days. For each quadruple in the training set that has closed-interval, i.e., ⟨s, r, o, $[t_{start}, t_{end}]$⟩, we get two training data points ⟨s, r, o, t_{start}⟩ and ⟨s, r, o, t_{end}⟩. An alternative would be to get all intermediate time points between t_{start} and t_{end}. However, this approach would result in oversampling for relations with long duration [14]. Lastly, if either t_{start} and t_{end} is unknown, we only consider the known time point.

To test TEMT's ability to generalize on unseen entities, we design new splits based on YAGO11k and Wikidata12k and refer to them as ind-YAGO11k and ind-Wikidata12k, respectively. For inductive reasoning, the validation and test sets should have some entities that are not in the training set. We employ the algorithm from [9] to create the new splits. The algorithm samples an entity and removes this entity from the graph $\mathcal{G}$ if this removal does not result in any isolated node or any relation type with less than 100 edges in the graph. The removed entity and its edges are then added either to the validation set and or to the test set. Thus, each triple in the test set has either a new subject or a new object. The test set has 1062 and 1255 unseen entities for YAGO11k and Wikidata12k, respectively. The algorithm works in triple level therefore assumes that the underlying graph is static by ignoring the validity time intervals. Therefore, each split has different triples, not quadruples. As a last step, we attach the corresponding time information to each triple to convert it to a quadruple.

4.2 Evaluation Metrics

For time interval prediction, we use three interval metrics that compare the predicted interval $I_p = [t^p_{start}, t^p_{end}]$ and the ground-truth interval $I_g = [t^g_{start}, t^g_{end}]$. Ideally, I_p is completely the same as I_g or they have some overlap. If there is no overlap, at least I_p and I_g should be close to each other. The first metric gIOU (generalized intersection over union) [30] is defined as follows:

$$gIOU\,(I_p, I_g) = IOU\,(I_p, I_g) - \frac{|gap(I_p, I_g)|}{|hull(I_p, I_g)|} \tag{14}$$

where

$$IOU\left(I_p, I_g\right) = \frac{|overlap(I_p, I_g)|}{|I_p| + |I_g| - |overlap(I_p, I_g)|} \tag{15}$$

$|gap(I_p, I_g)|$ is the distance between I_p and I_g in non-overlapping case (otherwise 0) and the hull is the shortest interval that covers both I_p and I_g. The length of an interval is defined as $|[t_{start}, t_{end}]| = t_{end} - t_{start} + 1$. In Section 7.2, we provide a visual illustration demonstrating the terminology used.

The second metric is aeIOU [14], affinity enhanced intersection over union. It is defined as follows:

$$aeIOU\left(I_p, I_g\right) = \begin{cases} \frac{|overlap(I_p,I_g)|}{|hull(I_p,I_g)|} & |overlap(I_p, I_g)| > 0, \\ \frac{1}{|hull(I_p,I_g)|} & \text{otherwise.} \end{cases} \tag{16}$$

The drawback of aeIOU that it outputs the same scores for both I_p and I_{p^*} if hull(I_p, I_g) = hull(I_{p^*}, I_g), ignoring the fact that one of them can be closer to I_g. In order to address this drawback, the study in [6] introduces a new metric called gaeIOU (generalized aeIOU).

$$gaeIOU\left(I_p, I_g\right) = \begin{cases} \frac{|overlap(I_p,I_g)|}{|hull(I_p,I_g)|} & |overlap(I_p, I_g)| > 0, \\ \frac{|gap(I_p,I_g)|^{-1}}{|hull(I_p,I_g)|} & \text{otherwise.} \end{cases} \tag{17}$$

Using the three metrics, given in Equation 14, 16 and 17, we report time interval prediction results based on gIOU@k, aeIOU@k, and gaeIOU@k. Given the predicted intervals for a test triple $\langle s, r, o, ?\rangle$ and its ground-truth interval I_g, gIOU@k is defined as the following:

$$gIOU@k = \max_{1 \leq i \leq k} gIOU(I_{p_i}, I_g) \tag{18}$$

where I_{p_i} denotes ith predicted interval for that particular triple. aeIOU@k and gaeIOU@k are defined analogously. We report the results averaged over all the test triples. We present the variance results for each metric in Section 7.3. The range for gIOU is [-1,1]. We follow the same procedure as our baselines and report the scaled gIOU values ((gIOU+1)/2). The range for aeIOU and gaeIOU is [0,1].

4.3 Experimental Setup

For all experiments, the dimension d for the triple embedding e_{sro} is 768 and the dimension d' for time point embedding is 64. We set m in Equation (5) to 64. We use 128 time-corrupted negative samples. A further analysis on the effect of the number of negative samples can be found in Section 7.1. The results show that time-corrupted negative sampling strategy is more suitable for our problem. We train our model with Adam optimizer [17] for 50 epochs with a learning rate of 0.001 and margin value $\gamma = 2$. We set the threshold for greedy-coalescing to 0.65. We report the effect of hyperparameters in Section 4.6.

Model Variants We use two different variants of TEMT, namely, TEMT$_N$ and TEMT$_{ND}$. The variant TEMT$_N$ is trained without the entity descriptions but only with the entity and relation names. In order to reflect this change, we modify Equation (1) as $S_{sro} = N_s + N_r + N_o$. The variant TEMT$_{ND}$ is trained with both entity descriptions and names. Regarding the sequence length for the language model, we keep the default Sentence-BERT setting which is 128 tokens.

Baselines In order to compare TEMT variants against the state-of-the-art, we identify four different TKGC methods as baselines: HyTE [8], TNT-ComplEx [18], TIMEPLEX-base [14] and TIME2BOX-TNS [6]. To the best of our knowledge, these are the only methods that perform time interval prediction. TIMEPLEX has two variants: TIMEPLEX-base and TIMEPLEX. Unlike the other baselines, TIMEPLEX relies on temporal constraints to improve its performance. However, TIMEPLEX-base does not follow the same. Here, we report the results of the latter. All of the baselines are transductive and do not use pre-trained language models for learning entity and relation embeddings. To the best of our knowledge, TEMT is the only method that supports inductive reasoning for time interval prediction. Following our baselines, we only report on the test instances that contain known time points which is compatible with our metrics so we do not report results on test quadruples that has unknown start or end time points.

4.4 Transductive Time Interval Prediction

In this experiment, the task is to predict the validity time intervals of facts in TKGs, namely predicting ⟨s, r, o, ?⟩. We compare the TEMT variants with the baselines and report the transductive time interval prediction results for YAGO11k and Wikidata12k in Table 2. On the YAGO11k dataset, TEMT outperforms the baselines in all the metrics but aeIOU@10. Notably, in the gIOU@1 metric, TEMT achieves 16 points more than the next best competitor TIME2BOX-TNS. For Wikidata12k, we observe that TEMT variants show improvements in the gIOU@1 metric in comparison with the baselines. For aeIOU@1 and gaeIOU@1, which are more stringent metrics as discussed in 4.2, TEMT variants are outperformed by the baselines. However, we also observe that there is not a significant performance difference across the datasets unlike our baselines. This may indicate that TEMT is not very sensitive to dataset size since YAGO11k is half the size of Wikidata12k. Moreover, TEMT is competitive with the state-of-the-art on the metrics gIOU@10, aeIOU@10 and gaeIOU@10.

Comparing the variants, we observe that TEMT$_{ND}$ performs better than TEMT$_N$ in most cases. This observation supports the claim that entity descriptions improve the context and, therefore, help to create more meaningful semantic triple embeddings. We also observe that TEMT variants are better at capturing the start and the end years compared to intermediate years, which possibly hurts the time interval prediction performance. The possible explanation is that the text corpora that the language model is trained on generally contain either the starting date or end date. In addition, the textual descriptions of entities may contain temporally irrelevant descriptions. For instance, for time point

Table 2. Transductive time interval prediction experiment results on YAGO11k and Wikidata12k datasets. The values are expressed in percentages. Results marked (*) are taken from [6], results marked (†) are reproduced by us, and the others are taken from [14]. "-" denotes unavailable results.

Methods	gIOU@1	aeIOU@1	gaeIOU@1	gIOU@10	aeIOU@10	gaeIOU@10
			YAGO11k			
HyTE	15.96	5.41	-	-	-	-
TNT-ComplEx	20.78	8.40	-	-	-	-
TIMEPLEX-base†	23.77	12.62	6.92	48.30	**34.63**	26.63
TEMT$_N$	**39.85**	13.05	**10.05**	58.78	32.89	29.24
TEMT$_{ND}$	38.60	**13.48**	9.61	**60.65**	34.33	**30.34**
			Wikidata12k			
HyTE	14.55	5.41	-	-	-	-
TNT-ComplEx	36.63	23.35	-	-	-	-
TIMEPLEX-base†	39.44	**26.14**	17.23	69.00	46.82	42.98
TIME2BOX-TNS*	42.30	25.78	**17.41**	**70.16**	**50.04**	**47.54**
TEMT$_N$	39.35	12.90	8.81	61.68	34.97	30.71
TEMT$_{ND}$	**43.52**	17.13	12.58	65.84	42.00	38.43

Table 3. Inductive time interval prediction experiment results on ind-YAGO11k and ind-Wikidata12k datasets. The values are expressed in percentages.

Methods	gIOU@1	aeIOU@1	gaeIOU@1	gIOU@10	aeIOU@10	gaeIOU@10
			ind-YAGO11k			
TEMT$_N$	**39.07**	14.23	**10.32**	**61.53**	35.90	32.79
. TEMT$_{ND}$	37.20	**14.81**	10.15	60.07	**36.73**	**33.32**
			ind-Wikidata12k			
TEMT$_N$	**39.78**	12.94	9.18	60.88	34.92	31.07
TEMT$_{ND}$	38.43	**16.43**	**11.01**	**64.50**	**40.06**	**36.63**

2000, we may get an entity description from Wikipedia that is updated in 2020. Note that the results of TIME2BOX-TNS are taken from the paper [6]. We could not reproduce the results for Wikidata12k and test the method on YAGO11k as neither the source code nor the details for pre-processing the datasets is available. In addition, TIME2BOX-TNS does not provide results for the YAGO11k dataset.

4.5 Inductive Time Interval Prediction

In this experiment, we perform inductive time interval prediction on newly generated inductive datasets ind-YAGO11k and ind-Wikidata12k. Since all our baselines only support transductive reasoning, they cannot be compared with TEMT.

Hence, we exclude them from this experiment. The inductive time prediction results are reported in Table 3. ind-YAGO11k and ind-Wikidata12k have 1062 and 1255 unseen entities in test set, respectively. The results show that TEMT's performance on inductive datasets is quite close to the transductive setting (Table 2). This demonstrates the generalization power of TEMT on unseen entities by the usage of pre-trained language models.

Another observation is that we do not see any significant drop in performance although the models are trained on $\sim 4{,}000$ fewer training points than YAGO11k and $\sim 5{,}000$ fewer training points than Wikidata12k. Moreover, similar to Section 4.4, TEMT$_{ND}$ variant performs better in most cases. This indicates that the context that entity descriptions provide helps the model to capture the semantics of a triple better. TEMT is also applicable to a fully inductive setting where there is no overlap between train and test set entities, which we leave as future work.

4.6 Fine-grained Analysis

Triple Classification We investigate the representation power of triple embeddings by performing triple classification experiment. The motivation is to make sure that our text space is also meaningful like our time space. With this experiment, we predict whether a triple is correct or not. To this end, we train an MLP classifier [26] with *triple embeddings* (e_{sro}). We first convert the train and test quadruples into triples by removing time intervals. For the training set, we corrupt the subject or object randomly and create one negative example to avoid class imbalance. For the test set, we remove the triples that exist in training or validation set to prevent information leakage. For each test triple, we create one negative example that does not appear in training, validation or test set. We create the sentences by applying Equation (1) and extract the features using our text encoder. The sizes of the training sets are as follows: 32,690 for YAGO11k, 64,980 for Wikidata12k, 24,558 for ind-YAGO11k, and 54,646 for ind-Wikidata12k. Similarly, the sizes of the test sets are: 4,100 for YAGO11k, 5,530 for Wikidata12k, 8,880 for ind-YAGO11k, and 13,872 for ind-Wikidata12k.

We set L2 regularization term alpha to 0.05 and perform maximum 1000 iterations. We keep the default values of the MLP classifier in [26] for the other settings. TEMT achieves an accuracy of 89.12% on YAGO11k, 91.55% on Wikidata12k, 88.64% on ind-YAGO11k and 89.82% on ind-Wikidata12k respectively. It illustrates the effectiveness of the text encoder thus supporting the claim that the triple embeddings are semantically meaningful and potentially capture structural information.

Time Prediction Diagnosis Table 4 illustrates some examples for time interval prediction experiment on both YAGO11k and Wikidata12k datasets. "Triple" column represents some triples from the test set and the "Gold answer" column represents their correct validity time interval. The table covers the triples that occurred in different centuries and that have varying durations. The next columns

Table 4. Example predictions from Yago11k and Wikidata12k. The entity descriptions are not displayed although utilized. (T1: "Kaká member of sports team Hertha BSC", T2: "Ippling located in the administrative territorial entity Bezirk Lothringen", T3: "Paulo Lopes (footballer) plays for S.L. Benfica", T4: "Jeff Morrow is married to Anna Karen Morrow", T5: "Henry Clay is affiliated to Whig Party (United States)").

Triple	Gold answer	1st prediction	2nd prediction	3rd prediction
T1	**[2008, 2012]**	[2008, 2014]	[2011, 2012]	[2004, 2011]
T2	**[1871, 1920]**	[1860, 1920]	[1871, 2018]	[1919, 1920]
T3	**[1997, 2002]**	[1997, 2004]	[1999, 2000]	[2000, 2008]
T4	**[1947, 1993]**	[1956, 1992]	[1959, 1960]	[1960, 2013]
T5	**[1833, 1852]**	[1830, 1862]	[1817, 1847]	[1818, 1829]

Table 5. Results of different hyperparameters on the validation set of Wikidata12k.

d′	gaeIOU@1	learning rate	gaeIOU@1	margin	gaeIOU@1	threshold	gaeIOU@1
32	11.76	0.001	11.86	1	12.26	0.4	10.29
64	11.94	0.002	11.52	2	12.33	0.5	11.56
128	11.77	0.003	11.74	5	11.18	0.65	11.87
256	11.78	0.01	11.36	7	10.97	0.7	11.66

report the TEMT$_{ND}$ predictions for the corresponding triple. This experiment shows that TEMT$_{ND}$ is able to predict intervals that are close to the ground-truth. In the first row, TEMT successfully predicts the starting point but output a longer interval than the ground-truth. In the second row, the ending time point is predicted correctly with an earlier starting point from the gold answer. The predictions are usually a subset of the gold interval so it shows that the textual information helps to predict the time period of facts.

Ablation Study We explore the effect of various hyperparameters on the performance of TEMT with a number of experiments. This also allows us to choose the optimal parameters that are discussed in Section 4.3. The results are shown in Table 5 and the setting where d′=64, learning rate=0.001, margin=2, and threshold=0.65 obtains the best results.

5 Related Work

Static KGC methods [15] can be roughly divided into two: knowledge graph embedding (KGE) methods and text-based methods. KGE methods represent entities and relations with low-dimensional vectors. They can be broadly classified into three different types: translational [4], semantic matching [37], and deep learning methods [10]. A common approach for KGE methods is to learn a function to score the plausibility of a triple. These methods perform well on many downstream tasks such as link prediction. However, they only utilize the structure of a graph and cannot easily be adapted to use additional information such as the textual descriptions of entities and relations. Text-based methods, which we will discuss at the end of this section, utilize textual information available in knowledge graphs to infer missing links between entities.

Although the majority of prior research has focused on static KGs, there has been a growing interest in exploring evolving knowledge graphs [5]. In this section, we focus on interval-based TKG completion methods. A common approach is to incorporate time into the scoring functions of static KGE methods. For instance, HyTE [8] learns to assign a hyperplane for each time point. For each hyperplane, it learns the temporal embeddings of entities and relations using the TransE scoring function [4]. Since the hyperplanes are learned independently, it is not able to model the dependencies between the time points. Moreover, both TNT-ComplEx [18] and TIMEPLEX [14] are based on ComplEx [32] and learn complex-valued embeddings for entity, relation and time points. TNT-ComplEx extends ComplEx by adding a new factor and solve a tensor completion problem. TIMEPLEX adds multiple time-dependent components to the scoring function and also takes into account additional learned features such as temporal constraints. TIME2BOX [6] extends the box embedding idea [29] by time-aware boxes and allows atemporal and temporal facts. Unlike TEMT, these models do not benefit from external information such as textual descriptions of entity and relations. Furthermore, these models are transductive so they cannot predict on unseen entities.

Recent works on text-enhanced static KGs employ pre-trained language models for static KGC [38, 9, 20, 34, 2]. The textual descriptions are fed into pre-trained language models (PLMs), that store real-world knowledge in their parameters, to obtain rich contextual entity/relation representations. However, they ignore the dynamics in which the relations between entities hold in a time interval. Only a few studies combines LMs and TKGs. ECOLA [13] jointly optimizes the LM and TKG embedding objectives via combining their loss functions. It retrieves textual information from news articles that correspond to specific dates. Moreover, it does not focus on time interval prediction. By contrast, TEMT leverages entity/relation names and descriptions from Wikipedia pages for time interval prediction. Similar to TEMT, SST-BERT [7] combines the textual information of entities/relations with time to get a plausibility score of a temporal fact. However, it utilizes relation paths with a primary focus on relation prediction whereas TEMT focuses on time interval prediction.

6 Conclusion

We propose TEMT, a model for text-enhanced temporal knowledge graph completion. TEMT outperforms state-of-the-art methods on the YAGO11k dataset and achieves competitive results on the Wikidata12k dataset. To the best of our knowledge, TEMT is the first method that is capable of performing time interval prediction on unseen entities. As a future work, we plan to investigate other pre-trained language models e.g. RoBERTa [21] and time encoding methods. We also plan to incorporate structural information into TEMT's fusing function.

7 Appendix

7.1 Effect of Number of Negative Samples

Table 6. Time prediction performance with respect to the number of negative samples.

	gIOU@1	aeIOU@1	gaeIOU@1	gIOU@10	aeIOU@10	gaeIOU@10
			YAGO11k			
#Entity-corrupted						
16	14.67	1.37	0.34	42.82	4.73	2
32	21.29	0.65	0.24	44.72	3.14	1.78
64	22.61	5.35	2.49	39.29	11.55	7.34
128	4.77	0.46	0.1	35.09	1.52	0.68
#Time-corrupted						
16	44.24	11.36	9.05	58.71	32.45	28.38
32	41.17	12.78	9.77	59.54	33.38	29.39
64	39.50	13.14	9.59	60	34.02	30.03
128	38.60	13.48	9.61	60.65	34.33	30.34
			ind-YAGO11k			
#Entity-corrupted						
16	26.96	4.8	2.22	40.52	10.81	6.69
32	32.75	1.96	1.08	47.55	6.99	4.73
64	11.99	1.29	0.33	33.01	2.11	1.11
128	3.28	0.22	0	20.72	0.39	0.06
#Time-corrupted						
16	47.47	14.39	12.19	61.64	36.23	33
32	41.89	15.31	11.51	62.08	37.50	34.17
64	41.79	14.12	10.71	63.70	37.61	34.6
128	37.20	14.81	10.15	60.07	36.73	33.32

We conduct an empirical study to see how sampling types discussed in Section 3 affect the performance of TEMT$_{ND}$. We analyze different number of entity-corrupted and time-corrupted negative samples on YAGO11k and ind-YAGO11k datasets. The results are reported in Table 6. We perform the same experiments on Wikidata12k and ind-Wikidata12k as well, however, we do not include their results here, for the sake of brevity.

The results in Table 6 show that the entity-corrupted negative sampling performs worse than the time-corrupted negative sampling for both datasets. Since the time interval prediction also requires the model to distinguish facts with different time points, this difference is expected. Moreover, in time-corrupted cases, the number of negative samples does not result in marginal changes in gaeIOU@1 metric, which is the most stringent metric.

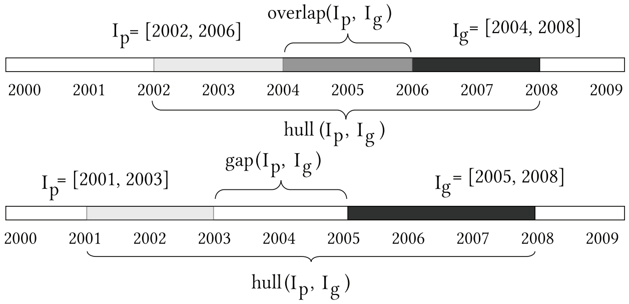


Fig. 3. Time prediction evaluation terms

Table 7. The variance values over the test set

Methods	gIOU@1	aeIOU@1	gaeIOU@1	gIOU@10	aeIOU@10	gaeIOU@10
			YAGO11k			
TEMT$_N$	0.0485	0.0286	0.0317	0.0546	0.0894	0.1037
TEMT$_{ND}$	0.0466	0.0246	0.0280	0.0466	0.0832	0.0990
			Wikidata12k			
TEMT$_N$	0.0411	0.0203	0.0223	0.0385	0.0670	0.0828
TEMT$_{ND}$	0.0424	0.0281	0.0334	0.0407	0.0716	0.0897
			ind-YAGO11k			
TEMT$_N$	0.0478	0.0291	0.0330	0.0522	0.0953	0.1097
TEMT$_{ND}$	0.0499	0.0290	0.0331	0.0636	0.1013	0.1178
			ind-Wikidata12k			
TEMT$_N$	0.0411	0.0209	0.0238	0.0453	0.0746	0.0897
TEMT$_{ND}$	0.0482	0.0292	0.0332	0.0450	0.0803	0.0969

7.2 Evaluation Terminology

In this section, we illustrate the evaluation terms employed within our interval metrics. In Figure 3, we demonstrate two scenarios when the predicted interval overlaps with the gold interval (top figure) and when it does not (bottom figure). In the case of the former, given $I_p = [2002, 2006]$ and $I_g = [2004, 2008]$, we get the following: the $hull(I_p, I_g) = [2002, 2008]$, and the $overlap(I_p, I_g) = [2004, 2006]$. In the latter case, given $I_p = [2001, 2003]$ and $I_g = [2005, 2008]$, then the $hull(I_p, I_g) = [2001, 2008]$ and $|gap(I_p, I_g)| = 3$.

7.3 Variance Analysis

Since our experimental results are averaged over the test triples, we report the variance values in Table 7.1. The results across the datasets and the variants illustrate the effectiveness of our model.

References

1. Agarap, A.F.: Deep learning using rectified linear units (relu). ArXiv **abs/1803.08375** (2018)
2. Alam, M.M., Rony, M., Nayyeri, M., Mohiuddin, K., Akter, M.M., Vahdati, S., Lehmann, J.: Language model guided knowledge graph embeddings. IEEE Access **10**, 76008–76020 (2022)
3. AlKhamissi, B., Li, M., Celikyilmaz, A., Diab, M.T., Ghazvininejad, M.: A review on language models as knowledge bases. ArXiv **abs/2204.06031** (2022)
4. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Neural Information Processing Systems (2013)
5. Cai, B., Xiang, Y., Gao, L., Zhang, H., Li, Y., Li, J.: Temporal knowledge graph completion: A survey. In: International Joint Conference on Artificial Intelligence (2022)
6. Cai, L., Janowicz, K., Yan, B., Zhu, R., Mai, G.: Time in a box: Advancing knowledge graph completion with temporal scopes. Proceedings of the 11th Knowledge Capture Conference (2021)
7. Chen, Z., Xu, C., Su, F., Huang, Z., Dou, Y.: Incorporating structured sentences with time-enhanced bert for fully-inductive temporal relation prediction. Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (2023)
8. Dasgupta, S.S., Ray, S.N., Talukdar, P.P.: Hyte: Hyperplane-based temporally aware knowledge graph embedding. In: Conference on Empirical Methods in Natural Language Processing (EMNLP) (2018)
9. Daza, D., Cochez, M., Groth, P.T.: Inductive entity representations from text via link prediction. Proceedings of the Web Conference 2021 (2020)
10. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI Conference on Artificial Intelligence (2017)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: North American Chapter of the Association for Computational Linguistics (2019)
12. Gu, K., Budhkar, A.: A package for learning on tabular and text data with transformers. In: MAIWORKSHOP (2021)
13. Han, Z., Liao, R., Liu, B., Zhang, Y., Ding, Z., Köppl, H., Schütze, H., Tresp, V.: Enhanced temporal knowledge embeddings with contextualized language representations. ArXiv **abs/2203.09590** (2022)
14. Jain, P., Rathi, S., Mausam, Chakrabarti, S.: Temporal knowledge base completion: New algorithms and evaluation protocols. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2020)
15. Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems (2020)
16. Jia, Z., Pramanik, S., Roy, R.S., Weikum, G.: Complex temporal question answering on knowledge graphs. Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR) (2015)
18. Lacroix, T., Obozinski, G., Usunier, N.: Tensor decompositions for temporal knowledge base completion. International Conference on Learning Representations (ICLR) (2020)

19. Leblay, J., Chekol, M.W.: Deriving validity time in knowledge graph. Companion Proceedings of the The Web Conference 2018 (2018)
20. Li, M., Wang, B., Jiang, J.: Siamese pre-trained transformer encoder for knowledge base completion. Neural Processing Letters (2021)
21. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. ArXiv **abs/1907.11692** (2019)
22. Mahdisoltani, F., Biega, J.A., Suchanek, F.M.: Yago3: A knowledge base from multilingual wikipedias. In: Conference on Innovative Data Systems Research (2015)
23. Mavromatis, C., Subramanyam, P.L., Ioannidis, V.N., Adeshina, S., Howard, P., Grinberg, T., Hakim, N., Karypis, G.: Tempoqr: Temporal question reasoning over knowledge graphs. In: AAAI Conference on Artificial Intelligence (2021)
24. Nguyen, D.Q., Nguyen, D.Q., Nguyen, T.D., Phung, D.Q.: A convolutional neural network-based model for knowledge base completion and its application to search personalization. Semantic Web (2019)
25. Ostendorff, M., Bourgonje, P., Berger, M., Schneider, J.M., Rehm, G., Gipp, B.: Enriching bert with knowledge graph embeddings for document classification. ArXiv **abs/1909.08402** (2019)
26. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Weiss, R.J., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research (2011)
27. Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.H., Riedel, S.: Language models as knowledge bases? Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (2019)
28. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Conference on Empirical Methods in Natural Language Processing (EMNLP) (2019)
29. Ren, H., Hu, W., Leskovec, J.: Query2box: Reasoning over knowledge graphs in vector space using box embeddings. International Conference on Learning Representations (2020)
30. Rezatofighi, S.H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I.D., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
31. Rogers, A., Kovaleva, O., Rumshisky, A.: A primer in bertology: What we know about how bert works. Transactions of the Association for Computational Linguistics (2020)
32. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. International conference on machine learning (2016)
33. Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Neural Information Processing Systems (2017)
34. Wang, L., Zhao, W., Wei, Z., Liu, J.: Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (2022)
35. Weikum, G., Dong, L., Razniewski, S., Suchanek, F.M.: Machine knowledge: Creation and curation of comprehensive knowledge bases. Found. Trends Databases (2020)

36. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: AAAI Conference on Artificial Intelligence (2016)
37. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. International Conference on Learning Representations (ICLR) (2015)
38. Yao, L., Mao, C., Luo, Y.: Kg-bert: Bert for knowledge graph completion. ArXiv **abs/1909.03193** (2019)
39. Zhang, X., Cheng, W., Zong, B., Chen, Y., Xu, J., Li, D., Chen, H.: Temporal context-aware representation learning for question routing. Proceedings of the 13th International Conference on Web Search and Data Mining (2020)