Data Mining and Knowledge Discovery (2025) 39:81
https://doi.org/10.1007/s10618-025-01156-w

JammyTS: joint attention and memory network for temporal scoping of facts

Chenxi Hu[1] · Tao Wu[1,2] · Chunsheng Liu[1] · Chao Chang[1]

Received: 25 February 2024 / Accepted: 6 August 2025

Abstract
Temporal Scoping of Facts is crucial for completing the temporal dimension of knowledge graphs. Current mainstream methods rely heavily on external resources for mining temporal information. However, the presence of noise in external resources, coupled with limitations in adaptively inferring non-continuous temporal dimensions with multiple temporal ranges, leads to low accuracy in predicting temporal ranges. To address these challenges, a model named JammyTS is proposed, which Joins an attention mechanism and a memory network for Temporal Scoping of facts. Specifically, JammyTS leverages attention to adjust the distribution of weights dynamically in memory networks and builds attention capsule-based networks to reduce the impact of noise in external resources. Furthermore, two linear classifiers are separately trained to infer the end and beginning timestamps of facts for inference of non-continuous temporal ranges. Extensive experiments on three datasets show that JammyTS improves the accuracy by up to 12.29% compared to the state-of-the-art.

Keywords Temporal scoping · Attention memory network · Attention capsule · Non-continuous temporal range

✉ Tao Wu
wutao20@nudt.edu.cn

Chenxi Hu
huchenxi20@nudt.edu.cn

Chunsheng Liu
liuchunsheng17a@nudt.edu.cn

Chao Chang
changchao17@nudt.edu.cn

[1] College of Electronic Engineering, National University of Defense Technology, Hefei, Anhui, China

[2] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

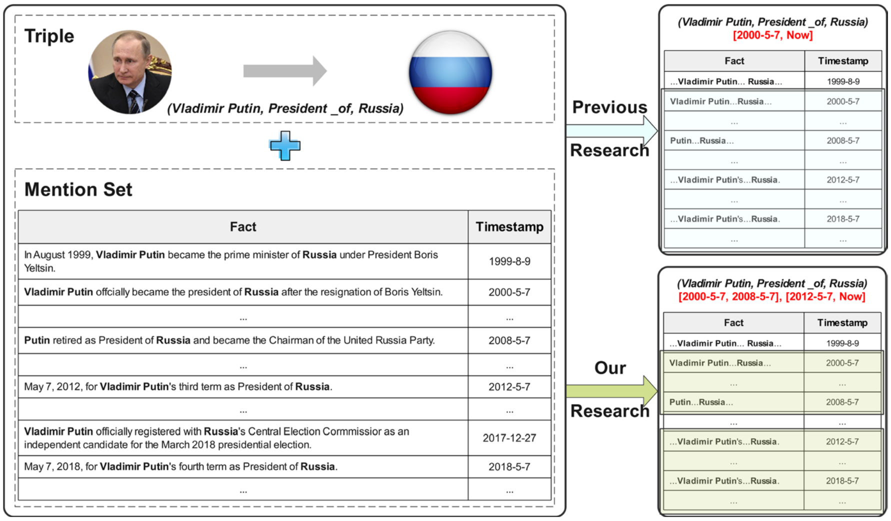
Published online: 06 September 2025

1 Introduction

Knowledge Graphs (KGs) are repositories rich in factual information and are vital for higher-level applications like question answering (Yu et al. 2023; Bae et al. 2023) and information retrieval (Cong et al. 2023; Hu et al. 2023; Fu et al. 2022; Yuan et al. 2022; Messner et al. 2022). Although certain facts are static, many details, such as occupations, marital partners, and athletes' team affiliations, change over time. Nonetheless, temporal data in many knowledge bases, such as YAGO (Hoffart et al. 2013) and Vrandečić and Krötzsch (2014), is sparsely available, which significantly hampers their effectiveness. As a result, recent scholarly focus has shifted toward the inference of temporal data for these dynamic facts, a process referred to as Temporal Scoping of Facts (TSF).

The objective of TSF is to determine the temporal range during which fact triples, structured as (*head*, *rel*, *tail*), remain valid. These triples are sensitive to time (Dikeoulias et al. 2022; Gao et al. 2023; Li et al. 2023; Zhang et al. 2023; Xu et al. 2023). For example, in financial KGs, triples related to "equity_ownership_of" may change due to activities associated with the *head*. TSF enhances the realism of KGs and provides higher-quality knowledge support for advanced applications.

External resources such as Wikipedia, external knowledge bases, or internet search engines can provide extensive relevant facts in sentence form. For example, as illustrated in Fig. 1 for the triple (Vladimir Putin, President_of, Russia), we can determine the temporal ranges through the Mention Set provided. Each sentence in the Mention Set includes the entities "Vladimir Putin" and "Russia", along with the timestamps of the relevant facts. Various methods, including statistical-based approaches (Wijaya et al. 2014; Talukdar et al. 2012), information extraction techniques (Gupta and Ber-



Fact	Timestamp
In August 1999, **Vladimir Putin** became the prime minister of **Russia** under President Boris Yeltsin.	1999-8-9
Vladimir Putin officially became the president of **Russia** after the resignation of Boris Yeltsin.	2000-5-7
...	...
Putin retired as President of **Russia** and became the Chairman of the United Russia Party.	2008-5-7
...	...
May 7, 2012, for **Vladimir Putin's** third term as President of **Russia**.	2012-5-7
...	...
Vladimir Putin officially registered with **Russia's** Central Election Commissioner as an independent candidate for the March 2018 presidential election.	2017-12-27
May 7, 2018, for **Vladimir Putin's** fourth term as President of **Russia**.	2018-5-7
...	...

(Vladimir Putin, President _of, Russia)
[2000-5-7, Now]

Fact	Timestamp
...**Vladimir Putin**... **Russia**...	1999-8-9
Vladimir Putin...Russia...	2000-5-7
...	...
Putin...Russia...	2008-5-7
...	...
...Vladimir Putin's...Russia.	2012-5-7
...	...
...Vladimir Putin's...Russia.	2018-5-7
...	...

(Vladimir Putin, President _of, Russia)
[2000-5-7, 2008-5-7], [2012-5-7, Now]

Fact	Timestamp
...**Vladimir Putin**... **Russia**...	1999-8-9
Vladimir Putin...Russia...	2000-5-7
...	...
Putin...Russia...	2008-5-7
...	...
...Vladimir Putin's...Russia.	2012-5-7
...	...
...Vladimir Putin's...Russia.	2018-5-7
...	...

Fig. 1 A scenario for the TSF and the difference between previous research and our research. The bold text in the sentences are entities

berich 2018; Sil and Cucerzan 2014; Cao et al. 2020), and hybrids of both (Rula et al. 2019), are employed to mine temporal information from diverse perspectives.

The aforementioned methods can infer the temporal range, but they struggle to identify and filter two types of noise in the Mention Set and lack the ability to infer the non-continuous temporal range of query triples. Firstly, the external facts contained in MS are obtained by retrieving the facts containing *head* and *tail* of the query triple. In the Mention Set, there exist external facts that, although containing *head* and *tail*, express semantics unrelated to the query triple. For example, for the triple (Vladimir Putin, President_of, Russia), an external fact such as "I recently discovered a painting depicting Putin and symbolic buildings in Russia at a cultural exhibition" is irrelevant. These facts do not use *head* or *tail* as the subject or object and need to be recognized as a noise fact. Secondly, within the extensive Mention Set, there are instances where facts with identical semantics have different timestamps. Assuming there is self-consistency in external resources, it is preferable to consider timestamps that appear less frequently as noise. Thirdly, previous methods typically fail to effectively infer non-continuous temporal dimensions that have multiple temporal ranges. For example, in Fig. 1, the triple (Vladimir Putin, President_of, Russia) has two temporal ranges in reality: [2000-5-7, 2008-5-7] and [2012-5-7, Now]. However, previous methods could only infer a single temporal range ([2000-5-7, Now]).

In this paper, we introduce a novel model, JammyTS, which Joins the attention mechanism and memory network for TSF. JammyTS integrates the attention mechanism with memory networks to precisely measure the attention distribution between query triples and external facts. This fusion enables memory networks to focus on external facts highly similar to the query triples and dynamically adjust weight allocation based on different inputs. To improve resistance against noise, we designed an attention capsule-based network that assesses the likelihood of noise for each fact and timestamp. This network estimates the probability of noise for each external fact and timestamp based on the attention distribution of the target fact throughout the Mention Set. JammyTS's learning process is divided into two phases: initially learning the end timestamp and subsequently the beginning timestamp. In the first phase, external facts chosen as potential end timestamps are evaluated for their semantic similarity to the query triple, relevance to the query triple's representation, and low noise probability. We extract two principal semantic similarities from the memory network, which, along with the noise probability, are used to train a linear classifier. This classifier assigns scores to each external fact's timestamp, ranking them as potential end timestamps. In the second phase, focusing on identifying the beginning timestamp, suitable external facts should exhibit high semantic similarity to the query triple, earlier timestamp values, and minimal noise interference. Each candidate fact is assigned a time-related weight. By integrating semantic similarity and noise probability, we train another linear classifier that scores and ranks each external fact's timestamp as a potential beginning timestamp.

Our contributions are as follows.

- We introduce an attention mechanism to dynamically adjust the distribution of memory slots within the network and propose an attention capsule-based network designed to effectively filter out noise from external resources.

- We present a novel method to separately learn the beginning and end timestamps of query triples, enabling effective inference across non-continuous time ranges.
- Experimental results demonstrate that JammyTS enhances the accuracy of TSF by up to 12.29% compared to the state-of-the-art on three datasets.

In the rest of the paper, we cover the related work in Sect. 2 and then define the TSF in Sect. 3. After presenting our method in Sect. 4, we report the experimental results in Section 5. We finally conclude our work and limitations in Sects. 6 and 7, respectively.

2 Related work

Existing research on TSF can be divided into two categories: inferring within temporal KGs and inferring with external resources.

2.1 Inferring within temporal KGs

This approach involves inferring the temporal range of facts by predicting their state in the future. The complexity of this process lies in the need for a profound understanding of the passage of time, the evolution of facts, and their interactions. In this field, Dasgupta et al. (2018) introduce a time-aware knowledge graph representation method, which utilizes temporal information as guidance for inference and prediction. By capturing subtle changes and trends in the flow of time, it accurately forecasts the future state of facts at a specific time point, thereby defining their temporal scope. Additionally, Leblay and Chekol (2018) approach from another perspective, focusing on handling Temporal Knowledge Graphs (TKGs) with missing time dimensions. They observed that even when temporal information for certain facts is absent, implicit associations still exist between these facts and other temporally annotated facts in the TKG. Building on this observation, they propose predicting the temporal scope of facts by mining and leveraging these implicit associations. Additionally, the research by Li et al. (2022a) and Li et al. (2022b) is noteworthy. Instead of directly forecasting the future state of facts, they shift towards analyzing past facts. They construct patterns based on the sequentiality, repetitiveness, and periodicity of past facts, revealing the regularities and trends in the evolution of facts over time. Through these patterns, it becomes possible to anticipate future occurrences of facts and obtain the temporal dimensions for given facts based on the TKG for each time period.

2.2 Inferring with external resources

This approach extensively relies on rich external resources such as Wikipedia and external knowledge bases for delineating temporal ranges. These resources offer valuable references and foundations for determining the temporal boundaries of facts. For instance, Wijaya et al. (2014) focus on modeling state changes of entities in external facts. By analyzing these changes comprehensively, specific timestamps of state changes caused by real-world events for given entities can be identified. Additionally, Sil and Cucerzan (2014) employ remote supervision methods for inferring temporal ranges. They devise a two-step classification method aimed at mitigating potential noise introduced during remote supervision training. Gupta and Berberich (2018) focus on supervised learning and utilize a text corpus labeled with temporal information as a training set to fill in missing temporal data in the knowledge graph. In addition, Wang et al. (2019) take a different approach, employing unsupervised learning. They predict the temporal ranges of queried facts based on the types of external facts at different times in context. Rula et al. (2019) match relevant evidence extracted from web documents with relevant temporal ranges extracted from linked data. This method provides confidence scores for each temporal range, thus more accurately delineating temporal boundaries. By leveraging information from web documents and linked data, this method offers reliable confidence assessments for defining temporal ranges. Cao et al. (2020) propose MemTimes, a model based on memory networks. This model measures the semantic similarity of external facts relevant to the queried fact using memory networks and designs a sliding window, resolved through heuristic algorithms, to delineate temporal ranges. However, the extent to which each external fact is semantically dissimilar to the triple is measured in the memory network, it is measured in the same memory network as the similarity between them by MemTimes. Different targets may require different treatments and weights, but using the same memory vector as a common base may lead to inaccurate weights and thus affect the predictive performance of the model (Lin et al. 2021; Lai et al. 2019; Fu et al. 2022; van Bokkem et al. 2023). Furthermore, the heuristic algorithm used in this model for TSF by semantic similarity ranking is often biased (de Souza 2021).

In this work, we introduce attention mechanisms to infer with external resources for TSF. Unlike previous research, we propose an attention capsule-based network to filter noise external facts and timestamps and design a novel method capable of inferring non-continuous temporal dimensions.

3 Problem statement

In this section, we formally define the problem of TSF as follows.

Problem 1 (Temporal Scoping). Given a query fact triple Tri =(*head*, *rel*, *tail*), consisting of three elements: the head entity, the relation, and the tail entity. Also, given the mention set $MS(Tri) = \{(f_i, t_i)\}_i^M$ associated with this query fact triple, where *f* denotes external facts in sentence form containing *head* and *tail*, and *t* denotes the timestamp. The goal of this task is to infer the time range [t^{begin}, t^{end}] of the query triple based on the sentence-form external facts in the mention set *MS*(*Tri*), where t^{begin} denotes the start timestamp and t^{end} denotes the end timestamp.

4 Methodology

This section specifically describes the JammyTS to solve the aforementioned problems. The query fact triple and mention set are first input into a framework for computing semantic similarity and noise probabilities, thus providing a data foundation for subsequent temporal range inference. This framework consists of six modules, structured as depicted in Fig. 2. Firstly, the input module receives raw data such as the query fact triple *Tri* and *MS*(*Tri*). During processing, *Tri* is input into the antonym query module for deep embedding representation and antonym query operations. Meanwhile, *MS*(*Tri*) is divided into two sets: the fact set $F = \{f_i | i = 1, \ldots, M\}$ and the timestamp set $T = \{t_i | i = 1, \ldots, M\}$. Each element in fact set F inputs the Fact Embedding module to obtain embedding representation, similarly, each timestamp in the timestamp set T inputs the Timestamp Embedding module to embed time information. Subsequently, the Attention Memory Network module receives the embedding vectors of the query fact triple and external facts as input to compute their similarity. Additionally, the framework includes an Antonym Query module, which processes the query fact to obtain its antonym phrases and embedding vectors. Finally, the Noise Detection module is responsible for identifying and handling noise data.

After obtaining semantic similarity and noise probabilities, they are further utilized to train two linear classifiers, each responsible for determining t^{end} and t^{begin}. During training, it is observed that for two external facts describing the same entity,

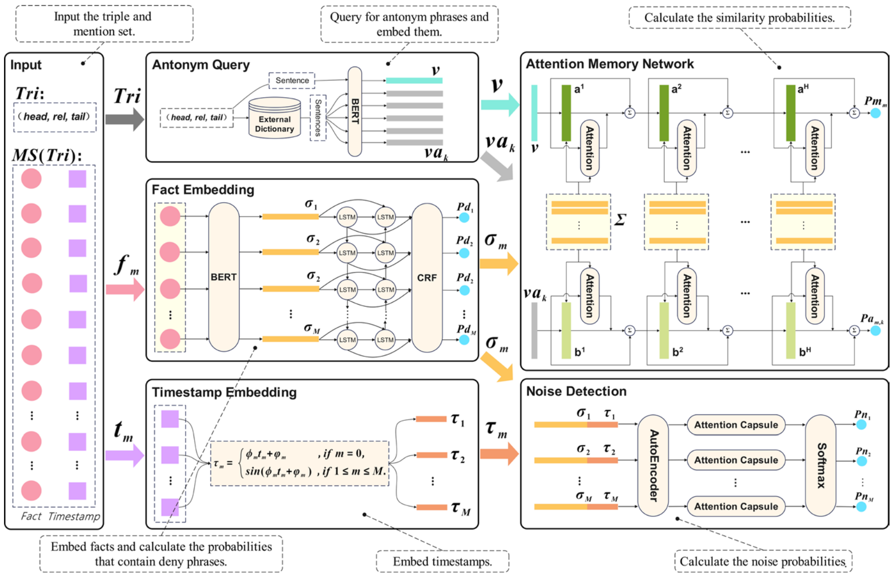


Fig. 2 Main framework of JammyTS. Query triples, external facts, and timestamps from the Input module are embedded in the Antonym Query, Fact Embedding, and Timestamp Embedding modules. Then, the triple and fact embeddings are input into the Attention Memory Network module to measure similarity for sorting the facts. Furthermore, each fact embedding and the corresponding timestamp embedding are concatenated and input to the Noise Detection module to obtain the noise probability

even if their semantics are opposite, their embedding representations in the vector space often exhibit relatively high similarity instead of theoretically low similarity. To address this issue and accurately determine t^{end}, particular attention is paid to two types of relation terminations within semantically similar external fact texts: deny phrases and antonym phrases. Intuitively, the inclusion of negation phrases (such as "not" and "no more") often indicates the end of a relation or fact. Therefore, the probability of external fact texts containing these deny phrases is specifically calculated within the Fact Embedding module. Similarly, antonym phrases, such as "retire", may express semantics opposite to those in the query fact, thus potentially indicating the end of the relation or fact. Consequently, the query fact triple is transformed into a concise sentence and conventional antonym phrases retrieved from external dictionaries (e.g., WordNet) are transformed into this concise sentence. Subsequently, the modified sentence's embedding vector is input into the Attention Memory Network module, utilizing its semantic calculation capability to measure the similarity between external facts and the modified sentence. Therefore, besides calculating the semantic similarity between external facts in the $MS(Tri)$ and the query fact Tri using the Attention Memory Network module, it is also employed to compute the similarity between external facts and facts expressing the opposite semantics of the query fact.

In the following sections, we provide a detailed description of modules from Sect. 4.1 to Sect. 4.5. The linear classifiers will be described in Sect. 4.6.

4.1 Fact embedding

This module is utilized for embedding external facts. When selecting embedding methods, it's crucial to consider the practicality of computational resources, effectiveness, and widespread adoption in the natural language processing community. Based on these considerations, the pre-trained language model BERT (Devlin et al. 2019) is employed for this task. Renowned for its robust semantic comprehension and wide applicability, BERT is utilized in this section to transform input external facts f into embeddings σ.

Additionally, special attention needs to be paid to the presence of denial phrases when handling external facts. Inspired by the widely used BERT-BiLSTM-CRF framework for addressing NER tasks, after BERT provides initial embedding representations, BiLSTM is employed to further encode embeddings using its bidirectional long short-term memory capabilities to capture temporal dependencies in the text. Finally, CRF is used for decoding, in a supervised learning fashion, to obtain the probability Pd of external facts containing deny phrases.

4.2 Timestamp embedding

The function of this module is to embed timestamps associated with external facts. Timestamps, as crucial information recording the moment when facts occur, are essential for accurately defining time intervals. To fully leverage this information, inspired by Han et al. (2021) Kazemi et al. (2019) are drawn upon. The following method is employed to convert timestamps into a feature vector: this vector not only encompasses periodicity information but also captures the chronological order of time.

$$\tau_m = \begin{cases} \phi_m t_m + \varphi_m, & \text{if } m = 0. \\ \sin(\phi_m t_m + \varphi_m), & \text{if } 1 \leqslant m \leqslant M, \end{cases} \tag{1}$$

where $\tau_m \in \mathbb{R}^{d_t}$, ϕ and φ_m are trainable parameters.

4.3 Antonym query

The primary task of this module is to query and embed phrases expressing the opposite semantic meaning of the *rel* in the query fact triple *Tri*, as well as obtain the embedding vector of this fact triple *Tri*. To achieve this goal, initially, antonymous phrases are obtained by querying external lexicons (such as WordNet). To ensure that these antonymous phrases and the embedding vector of *Tri* are located in the same vector space as the embedding vector of external facts, the BERT model is also used to represent them in embeddings.

When encoding *Tri* to obtain a semantically rich embedding representation, a simple method was adopted: *Tri* was transformed into a short sentence for input into BERT. For example, the triplet in Fig. 1 can be transformed into “Vladimir Putin is the president of Russia." Similarly, the retrieved antonymous phrases are inserted into this sentence to form a new statement, which is then input into the BERT model to obtain the embedding vectors *va* for the antonymous phrase of *rel* and *v* for *Tri*. Finally, *va* and *v* are input into the Attention Memory Network module to calculate the semantic similarity between *va* and *v*, and each embedding vector of external facts.

4.4 Attention memory network

In this module, a dual attention memory network is used to calculate the similarity between both *v* and antonym phrase of *v* and each fact embedding. Specifically, we use *v*, *va*, and $\boldsymbol{\Sigma}$ as inputs and output the similarity *Pm* and Pa_k, where *Pm* indicates the similarity between *v* and σ, and Pa_k indicates the similarity between va_k and σ.

In the *h*-th hop of the memory network, we transform $\boldsymbol{\Sigma}$ into a memory matrix $\boldsymbol{a}^h$ by a learnable weight matrix $\boldsymbol{A}$ as $\boldsymbol{a}^h = \boldsymbol{A}^T\boldsymbol{\Sigma}$, where $\boldsymbol{A} \in \mathbb{R}^{d_m \times d_v}$, d_m denotes the dimension of $\boldsymbol{\Sigma}$, and d_v denotes the dimension of *v*. To learn the similarity, we obtain the internal states $\boldsymbol{U}^h = \boldsymbol{C} \cdot \boldsymbol{v}$ and $\boldsymbol{Ua}_k^h = \boldsymbol{Ca} \cdot va_k$ through other learnable weight matrices $\boldsymbol{C}$ and $\boldsymbol{Ca}$ having the same dimension as $\boldsymbol{A}$, respectively. Next, we introduce the calculation for the similarity between *v* and σ. The cosine similarity is first calculated as

$$score^r = \frac{\boldsymbol{U}^{h^T} \boldsymbol{W}_c \boldsymbol{a}^h}{\| \boldsymbol{U}^{h^T} \| \| \boldsymbol{a}^h \|}, \tag{2}$$

where $\boldsymbol{a}^h$ is a memory slot, $score^r$ indicates the similarity score between $\boldsymbol{U}^h$ and $\boldsymbol{a}^h$, $\|\cdot\|$ is the $L2$ norm, and $\boldsymbol{W}_c$ is a weight. The weight of each memory slot is then obtained based on the similarity scores as

$$\boldsymbol{\alpha} = \frac{\exp(score^r)}{\sum_{i=1}^{|\Sigma|}\exp(score_i^r)}. \tag{3}$$

Next, the attention is used to transform the σ. σ and $\boldsymbol{U}^h$ are first mapped into the Key, Value and Query vectors through three different linear transformations as

$$\begin{aligned}\boldsymbol{K} &= \boldsymbol{U}^h\boldsymbol{W}_k,\\ \boldsymbol{V} &= \boldsymbol{U}^h\boldsymbol{W}_v,\\ \boldsymbol{Q} &= \Sigma\boldsymbol{W}_q,\end{aligned} \tag{4}$$

where $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d_v}$, $\boldsymbol{Q} \in \mathbb{R}^{d_v \times d_m}$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v$, and $\boldsymbol{W}_q$ are the three learnable weight matrices. Then, the similarity score is calculated between $\boldsymbol{Q}$ and $\boldsymbol{K}$, which is used to calculate the attention value of $\boldsymbol{U}^h$ to σ and is denoted as

$$score^a = \frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_v}}, \tag{5}$$

where d_v is the dimension of *v*. For the attention value $\{score_i^a | i = 1, \dots, M\}$, we need to scale and normalize them into $\{\boldsymbol{\beta}_i^a | i = 1, \dots, M\}$ via Softmax layer to prevent the inner product from becoming too large and leading to the problem of gradient explosion.

Afterwards, the attention weights $\boldsymbol{\beta}^a$ is multiplied with the corresponding vectors $\boldsymbol{V}$ to obtain the weighted sum vectors $\boldsymbol{b}^h$ as the output and the formula is

$$\boldsymbol{b}^h = \sum_{i=1}^{|\Sigma|} \boldsymbol{\beta}_i^a \boldsymbol{V}, \tag{6}$$

the weighted sum of the memory slot $\boldsymbol{b}^h$ is calculated based on the weights obtained from Eq. (3) as

$$\boldsymbol{S}^h = \sum_i \boldsymbol{\alpha}_i \boldsymbol{b}^h. \tag{7}$$

Finally, $\boldsymbol{S}^h$ and $\boldsymbol{U}^h$ are input to the binary classification and the semantic similarity between *Tri* and the *m*-th fact can be calculated as

$$Pm^h = \text{Sigmoid}(\boldsymbol{W}(\boldsymbol{S}^h + \boldsymbol{U}^h)), \tag{8}$$

where $\boldsymbol{W}$ is a learnable weight matrix. $\boldsymbol{W}_c$, $\boldsymbol{W}_m$, $\boldsymbol{A}$, $\boldsymbol{Ct}$, and $\boldsymbol{Ca}$ learn jointly in training. The method to calculate the similarity between each antonym phrase of *Tri* and fact is the same.

The input of the H-th hop is $\boldsymbol{U}^H = \boldsymbol{S}^{H-1} + \boldsymbol{U}^{H-1}$. Both the input $\boldsymbol{U}^H$ and output $\boldsymbol{S}^H$ of that hop are input into the binary classification to obtain the final *Pm* as

$$Pm = \text{Sigmoid}(\boldsymbol{W}_t(\boldsymbol{S}^H + \boldsymbol{U}^H)). \tag{9}$$

After obtaining the prediction results $\hat{Pm}$, we use it to denote whether the specified relation exists in the instance. When $\hat{Pm} = 1$, it indicates existence, otherwise, it signifies non-existence, in which case $\hat{Pm} = 0$. Thus, the Binary Cross-Entropy (BCE) loss function is formulated as follows:

$$\mathcal{L}_{\text{AMN}} = -\sum_{i=1}^{|\Sigma|} \hat{Pm}_i \log Pm_i + (1 - \hat{Pm}_i) \log(1 - Pm_i), \tag{10}$$

The semantic similarity Pa_k between va_k and $\boldsymbol{\sigma}$ is calculated in the same way, and we chose the max value in $\{Pa_1, Pa_2, \ldots, Pa_K\}$ to be the final *Pa*.

4.5 Noise detection

The Noise Detection module is responsible for calculating the probability that the timestamp of each external fact or the fact itself is noise, which affects the degree to which we consider that fact as a candidate one. The input of this module has two kinds, one is Σ, and another is the *MS*(*Tri*). For the latter one, we first concatenate the fact embedding and its timestamp embedding: $y = \sigma \oplus \tau$, where $y \in \mathbb{R}^{d_v + d_t}$. Then, we input *y* into the module, and the output is the probability *Pn* that *t* is noise.

4.5.1 AutoEncoder for dimensionality reduction

To enable the model to focus on the most critical features and ignore redundant and secondary information, we first use AutoEncoder to reduce the dimensionality of *y*.

AutoEncoder uses an encoder to extract high-dimensional features of σ and downscales them to output text features *e*, and the decoder rebuilds input *y* from the encoder to obtain y', with the purpose of fitting a constant function using a neural network. The input vector is encoded to get the reduced-dimensional vector and the training is completed by the decoder as follows.

$$e = f_1(\boldsymbol{W}_1 y + b_1), \tag{11}$$

$$y' = f_2(\boldsymbol{W}_2 e + b_2), \tag{12}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are learnable weight matrices, b_1 and b_2 are bias, *y* is the input vector, $\hat{y}$ is the prediction vector, $e \in \mathbb{R}^{d_e}$ is the vector after dimensionality reduction, and f_1 and f_2 are the mapping functions. The loss function is as follows.

$$\mathcal{L}_{\mathrm{AE}} = \frac{1}{n}\sum_{m=1}^{|\Sigma|} ||y_m - y'_m||^2 + \frac{\lambda}{2}||\boldsymbol{W}_n||_2^2, \tag{13}$$

where λ is the penalty factor, $\boldsymbol{W}_n$ is the parameters of the model and the $L2$ norm is used to regularize the parameters.

4.5.2 Attention capsule

Here we design an attention capsule for noise detection. For the reduced-dimensional concatenating vector e of facts and timestamps, its weights on any other pair of facts and timestamps are calculated to adaptively measure a representation of their semantic similarity, and the network structure of the attention capsule is shown in Fig. 3. To model the similarity between a certain external fact and others, we introduce the outer product of the two and stack the three. We then use a $d_e \times d_e$ convolution kernel to extract the features and connect them to a single fully connected layer to obtain the weight. The input to the attention capsule are e_i and $\boldsymbol{E}^{\mathrm{T}} = [e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_M]$ and the calculation formula is

$$\boldsymbol{W}_i^{\mathrm{ac}} = g(e_i, \boldsymbol{E}), \tag{14}$$

where $g(\cdot)$ is a feedforward network. First, the outer product of e_i and $\boldsymbol{E}_j$ are calculated as

$$\boldsymbol{O}_i = e_i \otimes \boldsymbol{E}_j, \tag{15}$$

we then stack $\boldsymbol{O} \in \mathbb{R}^{d_e \times d_e}$, e_i, and $\boldsymbol{E}_j$ and obtain $\boldsymbol{C} \in \mathbb{R}^{(d_e+2)\times d_e}$. Next, a convolution kernel $\boldsymbol{K}$ is used to convolve them as

$$\boldsymbol{I}_{i,j} = \Sigma_{u=1}^{d_e}\Sigma_{v=1}^{d_e}\boldsymbol{C}_{i+u-1,j+v-1} \cdot \boldsymbol{K}_{u,v}, \tag{16}$$

we then use the ReLU function for linear activation as

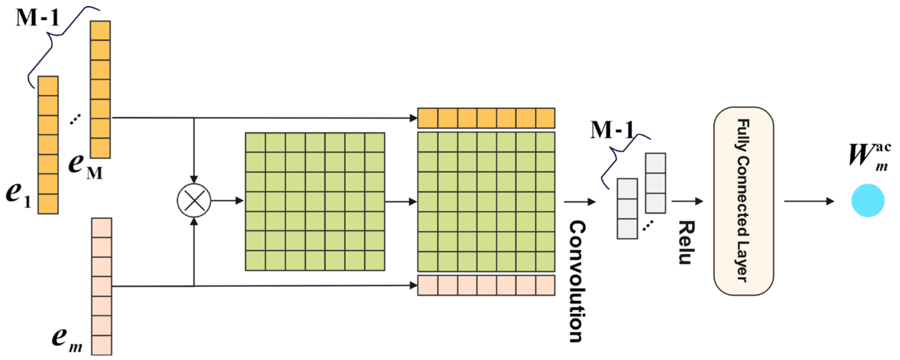


Fig. 3 Structure of the attention capsule

$$\boldsymbol{I}^{r} = \mathrm{ReLU}(\boldsymbol{I}), \tag{17}$$

and $\boldsymbol{I}^{r}$ is connected to a learnable fully connected layer to obtain the weight $\boldsymbol{W}^{\mathrm{ac}}$.

After obtaining the weight of each fact, the softmax layer is added for normalization as

$$\boldsymbol{W}_{i}^{\mathrm{ac}'} = \frac{\exp(\boldsymbol{W}_{i}^{\mathrm{ac}})}{\sum_{j}^{|\Sigma|}\exp(\boldsymbol{W}_{j}^{\mathrm{ac}})} \tag{18}$$

Recall that the attention capsule calculates the probability that the timestamp of each fact is true. Therefore, we proceed with the following transformation to obtain the probability that it is noise:

$$Pn = 1 - \boldsymbol{W}^{\mathrm{ac}'}. \tag{19}$$

Finally, we use BCE loss to learn the noise probability:

$$\mathcal{L}_{\mathrm{noise}} = -\sum_{i=1}^{|\Sigma|} \hat{Pn_i} \log Pn_i + (1 - \hat{Pn_i}) \log(1 - Pn_i), \tag{20}$$

where $\hat{Pn_i}$ and Pn_i are prediction and ground-truth label of noise external fact.

4.6 Learning temporal range boundary

4.6.1 Learning the end timestamp

In this section, we construct a linear classifier for learning the end timestamp with similarity probability *Pm*, noise probability *Pn*, deny phase probability *Pd*, and antonym probability *Pa*. During training, their different contributions to the final outcome are captured by assigning learnable weights to each variable. Specifically, a score y^{e} representing the probability that the timestamp *t* of the external fact *Tri* is predicted as the end timestamp is obtained by calculating the sum of these weighted variables. This process can be formalized as a linear equation:

$$y^{\mathrm{e}} = \beta_0^{\mathrm{e}} + \beta_1^{\mathrm{e}} Pm + \beta_2^{\mathrm{e}} Pn + \beta_3^{\mathrm{e}} Pa + \beta_4^{\mathrm{e}} Pd + \varepsilon^{\mathrm{e}}, \tag{21}$$

where β_0^{e} denotes the intercept, $\beta_i^{\mathrm{e}}(i = 1, \ldots, 4)$ is the learnable weight corresponding to each sample, ε^{e} denotes a bias. We use BCE loss to learn the end timestamp score:

$$\mathcal{L}_{\mathrm{end}} = -\sum_{i=1}^{|\Sigma|} \hat{y_i^{\mathrm{e}}} \log y_i^{\mathrm{e}} + (1 - \hat{y_i^{\mathrm{e}}}) \log(1 - y_i^{\mathrm{e}}), \tag{22}$$

where $\hat{y^{\mathrm{e}}}$ and y^{e} are prediction and ground-truth label of end timestamp.

We test the example in Fig. 1 to obtain y^{e} and the final binary curves of "Timestamp-Probability" are shown in Fig. 4. The "2008-5-7" is determined as the t^{end} with the highest score.

4.6.2 Learning the beginning timestamp

The objective of this section is to compute the score y^{b} for a certain timestamp *t* as the starting timestamp of *Tri*. To achieve this goal, two key variables, *Pm* and *Pn*, are utilized. Additionally, it is noted that the value of the timestamp *t* itself significantly impacts the score y^{b}. Specifically, smaller timestamps often yield higher scores. To incorporate this influence into the computation, a normalized timestamp t^{n} is introduced as another crucial variable. During the calculation process, learnable weights are assigned to each variable to ensure their contributions to the final score y^{b} are reasonably assessed. By summing the weighted variables, a comprehensive score is derived:

$$y^{\mathrm{b}} = \beta_0^{\mathrm{b}} + \beta_1^{\mathrm{b}} Pm + \beta_2^{\mathrm{b}} Pn + \beta_3^{\mathrm{b}} t^{n} + \varepsilon^{\mathrm{b}}, \tag{23}$$

where β_0^{b} denotes the intercept, $\beta_i^{b}(i = 1, \dots, 3)$ is the weight corresponding to each feature, ε^{b} denotes the error. The BCE loss function is denoted as

$$\mathcal{L}_{\mathrm{begin}} = -\sum_{i=1}^{|\Sigma|} \hat{y_i^{\mathrm{b}}} \log y_i^{\mathrm{b}} + (1 - \hat{y_i^{\mathrm{b}}}) \log(1 - y_i^{\mathrm{b}}), \tag{24}$$

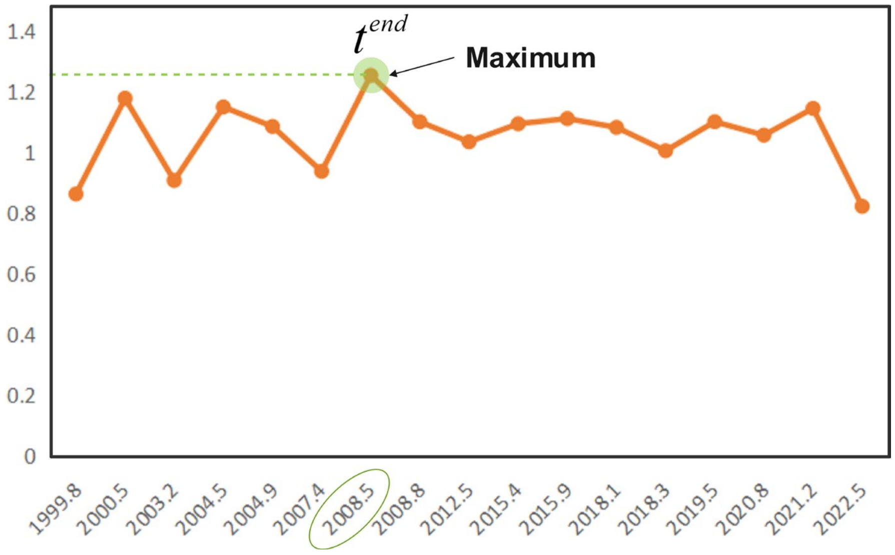


Fig. 4 Timestamp-Probability curve for the end timestamp. The timestamp in the green circle is the predicted end timestamp

where $\hat{y}^{\mathrm{b}}$ and y^{b} are prediction and ground-truth label of beginning timestamp.

We also test the example in Fig. 1 to obtain $Score^{\mathrm{b}}$, and the binary curve of "Timestamp-Probability" is shown in Fig. 5. After inferring the t^{end}, we prefer to locate the maximum value on the left side of t^{end} as the first beginning timestamp. On the right side, if there are any extreme values, we choose the largest extreme value as the second beginning timestamp. From Fig. 5, we can see that "2000-5-7" and "2012-5-7" are selected as the beginning timestamps t_1^{begin} and t_2^{begin}.

5 Experiments

In this section, we perform extensive experiments on various datasets, with the objective of answering the following research questions:

RQ1 Does JammyTS outperform other methods in accuracy?
RQ2 How does the learning end timestamp process work to improve performance?
RQ3 How does the attention contribute in this task?
RQ4 How does each probability contribute in this task?

5.1 Experimental setup

In this section, we describe the datasets, comparable models, parameter settings, and metrics of our experiments.

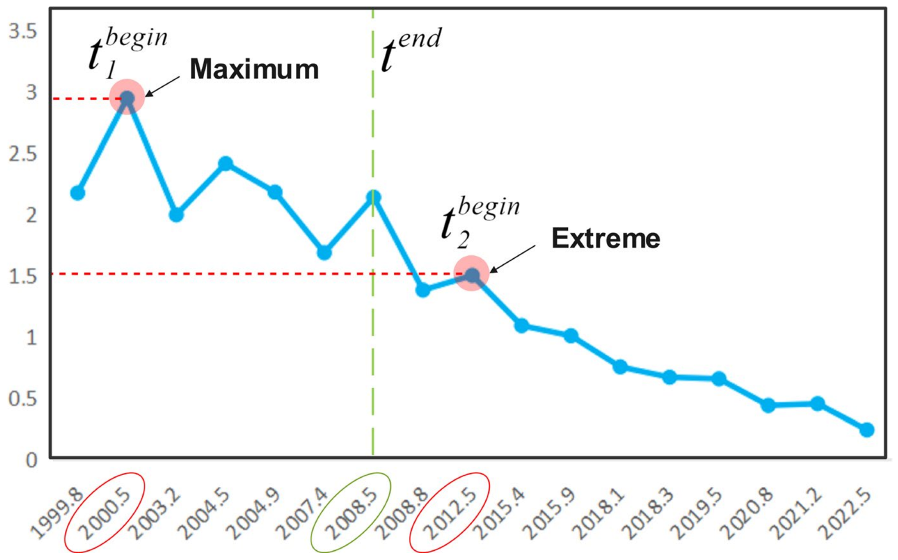


Fig. 5 Timestamp-Probability curve for the beginning timestamp. The timestamp in the green circle is the predicted end timestamp, while the one in the red circle is the predicted beginning timestamp

5.1.1 Dataset

In this experiment, we verify our model on three datasets: the TAC dataset, the Wikidata dataset, and the Cyber Security (CS) dataset.

TAC (Cao et al. 2020): This dataset was proposed in the Temporal Slot Filling task at the Text Analysis Conference (TAC-TSF) in 2013. As shown in Table 1, the dataset consists of seven relations. We preprocess the dataset to extract triples that include these relations and collect facts (in the form of sentences) associated with each triple from Wikipedia[1]. All the collected facts contain the *head* and *tail* of the relevant triples as well as timestamps. Finally, we sort these facts by timestamps to form the mention set of each triple.

Wikidata(Cao et al. 2020): We choose triples in Wikidata[2] that contain seven relations with timeliness and construct the mention set as described above.

CS [3]: We also collect facts about cyber security, which contain seven relationships that are time-sensitive. We choose this type of triples because there exists a large number of cyber security facts with multiple time ranges, such as the "$IP\_address$", where we know that the IP address of a certain server changes over time, while there may also be multiple time periods with the same IP address. (Table 1)

5.1.2 Comparable model

We compare JammyTS with three categories of TSF methods: the manually labeled standard datasets (i.e., LDC (manual)), the TAC-TSF models (i.e., BLENDER, UNED, and TSRF), and the memory networks-based models (i.e., MemTimes).

LDC (manual) : A standard dataset manually labeled by experts.[4]

[1] http://zh.wikipedia.org

Table 1 The relations, size of triples, and size of facts in three datasets

	TAC	Wikidata	Cybersecurity
Rel_1	Spouse	Spouse	Attack_vector
Rel_2	Title	Title	Access_level
Rel_3	Live_in	Countries_of_residence	IP_address
Rel_4	Cities_of_residence	Award_received	Affected_system
Rel_5	Employee_of	Employee_of	Attack_type
Rel_6	Statesorprovinces_of_residence	Member_of_team	Infection_method
Rel_7	Countries_of_residence	Educate_at	Authentication_protocol
Rel_8	16,430	21,831	3,246
Rel_9	68,370	113,218	7,243

[2] https://query.wikidata.org

[3] http://github.com/nickhcx/JammyTS

[4] https://catalog.ldc.upenn.edu

BLENDER (Taylor Cassidy et al. 2011): A structured method is used to combine query entity features and syntactic dependency features to develop a dependency path kernel for this task.
UNED (Guillermo Garrido et al. 2011): This method designs a remote supervised learning system that automatically collects training data from an external knowledge base and generates new features using Named Entity Recognition (NER), Syntactic Dependency Tree (SDT), etc.
TSRF (Sil and Cucerzan 2014): A linguistic model of a specific relationship is learned from Wikipedia, and then a model consisting of manually selected triggers is used to perform this task.
MemTimes (Cao et al. 2020): A method utilizing memory network for TSF with external resources.

5.1.3 Implementation detail

In the Fact Embedding module, the dimension of embeddings is set to 768. In the Attention Memory Network module, the number range of hops and iterations are [1, 2, 3, 4, 5], the dropout is 0.5, the number of epochs is 30, the learning rate is 0.01, and the optimization algorithm is Stochastic Gradient Descent (SGD). In the ND module, the encoder of the AutoEncoder contains input, hidden, and output layers with 1024, 1000, and 256 neurons, respectively.

For the comparable methods, we use the same pre-trained model BERT for text embedding, and for MemTimes which also uses memory networks, we use the same optimizer, initial learning rate, and number of epochs.

5.1.4 Metrics

In the experiments of previous reach, Boundary Accuracy (BA) is to measure the accuracy by comparing the gap between the two beginning (end) timestamps of the prediction range and the ground-truth range as follows.

$$BA = 0.5[\frac{1}{1+|t_b^{ev}-t_b^{pr}|}+\frac{1}{1+|t_e^{ev}-t_e^{pr}|}], \tag{25}$$

where t_b^{ev} and t_e^{ev} are the beginning and ending timestamps of the true range, and t_b^{pr} and t_e^{pr} are the beginning and ending timestamps of the predicted range. However, this metric is not applicable to non-continuous temporal ranges. If the predicted temporal range has only one segment, while the true temporal range has multiple segments, we cannot specify which pair of beginning timestamps or ending timestamps to compare.

To address this issue, we focus accuracy on the intersection of the predicted temporal range and the true temporal range. Here we also use the *Overlap Rate* (*OR*) as a metric to evaluate the experimental results. OR denotes the ratio of the intersection of the predicted range and the true range to the union of them. The formula for OR is as follows.

$$OR = \frac{I}{U}, \tag{26}$$

where I and U denote the intersection and the union between the prediction range and the ground-truth range, respectively.

5.2 Performance comparison (RQ1)

The empirical results of our proposed model JammyTS and the comparable models are shown in Table 2, which are averages obtained after ten times of repeated experiments. The main findings can be summarized as follows.

First, our proposed JammyTS has the best performance in each case, achieving 73.20% and 71.02% of the LDC standard dataset on OR and BA, respectively. JammyTS proves particularly advantageous on datasets containing more non-continuous temporal dimensions, such as the CS dataset, where it outperforms the comparable models by 12.29$\sim$35.56% on OR and 3.7$\sim$30.9% on BA. In addition, the OR and BA performance on other datasets show improvements of 6.53$\sim$33.58% and 1.96$\sim$ 29.58%, respectively. This observation highlights the effectiveness of JammyTS,

Table 2 The OR (%)/BA (%) comparison of different models on three datasets

Dataset	Rel	LDC	BLENDER	UNED	TSRF	MemTimes	JammyTS*
TAC	Rel_1	67.34/69.87	28.58/31.19	26.18/26.20	30.29/31.94	41.18/44.43	**48.21/48.75**
	Rel_2	50.16/60.22	11.03/13.07	5.07/6.88	32.32/36.06	35.65/41.04	**45.12/47.63**
	Rel_3	83.14/91.18	32.66/34.68	14.56/19.34	22.53/27.35	40.26/43.69	**50.06/49.36**
	Rel_4	74.01/81.10	25.14/29.04	12.15/14.47	30.19/33.04	41.82/46.82	**43.18/46.18**
	Rel_5	50.56/58.26	12.18/14.93	5.28/8.16	25.44/32.85	31.25/39.25	**44.27/47.23**
	Rel_6	65.12/72.27	23.12/26.71	12.14/15.24	32.57/40.12	36.17/41.93	**46.80/48.15**
	Rel_7	51.13/54.07	15.19/17.24	11.26/14.41	30.22/31.85	36.24/40.59	**42.62/46.21**
	All	62.18/68.84	22.13/23.42	12.21/14.79	29.46/33.15	39.26/43.75	**45.79/47.08**
Wikidata	Rel_1	71.16/73.15	31.08/32.18	27.89/29.16	31.13/33.75	43.96/48.36	**53.18/54.13**
	Rel_2	62.24/66.52	14.05/15.16	11.54/13.05	24.82/26.13	48.13/52.16	**56.33/56.23**
	Rel_3	80.15/81.43	29.14/33.15	22.21/24.36	39.08/41.28	42.14/48.78	**53.76/52.84**
	Rel_4	61.78/65.48	33.08/35.81	14.58/16.02	25.43/28.15	34.28/40.12	**44.91/45.63**
	Rel_5	57.45/59.30	22.58/25.63	12.55/15.29	35.84/36.12	45.23/50.86	**55.29/56.18**
	Rel_6	60.15/62.45	30.47/35.13	17.93/20.13	26.74/29.78	33.21/36.13	**44.21/39.12**
	Rel_7	75.28/78.19	14.56/16.10	15.76/16.83	28.86/30.69	41.26/48.92	**43.50/50.86**
	All	64.15/66.15	25.13/26.93	18.73/23.12	30.08/33.45	41.29/48.12	**52.14/54.92**
CS	Rel_1	62.13/70.96	10.88/16.36	10.12/15.16	23.18/26.17	30.31/34.85	**47.13/50.36**
	Rel_2	74.25/82.10	22.93/28.13	11.13/16.20	21.09/25.15	28.05/36.84	**37.42/38.13**
	Rel_3	61.67/69.15	12.08/15.46	13.43/15.42	14.46/18.96	31.91/40.56	**50.85/52.84**
	Rel_4	72.46/76.69	21.94/23.85	4.34/6.86	20.34/24.16	30.59/38.16	**39.01/39.56**
	Rel_5	80.83/83.64	18.34/22.37	8.49/12.69	17.18/18.69	35.01/40.12	**52.31/52.36**
	Rel_6	68.47/72.78	28.58/30.56	9.73/13.52	23.79/25.36	39.38/43.69	**47.16/50.13**
	Rel_7	71.24/73.56	17.14/20.14	11.17/15.85	20.58/23.64	36.79/38.14	**43.64/50.12**
	All	70.78/74.33	18.05/22.85	9.51/14.93	20.19/21.69	32.78/42.13	**45.77/45.83**

Bold indicates the highest value for each metric, and underline indicates the second highest

We record the results of the various models for each selected relation, as well as the overall results

Table 3 The OR (%) comparison of searching for different “ending” signs on CS dataset

Sign	Rel_1	Rel_2	Rel_3	Rel_4	Rel_5	Rel_6	Rel_7	*All*
w/oPA	37.83	29.03	40.01	29.98	37.47	38.45	34.82	35.48
w/oPD	41.78	33.54	47.82	31.79	41.72	42.13	39.73	39.18
JammyTS	**47.13**	**37.42**	**50.85**	**39.01**	**52.31**	**47.16**	**43.64**	**45.77**

Bold indicates the highest value, and underline indicates the second highest

“*PA*”: Probability of containing Antonym phrases, “*PD*”: Probability of containing Deny phrases

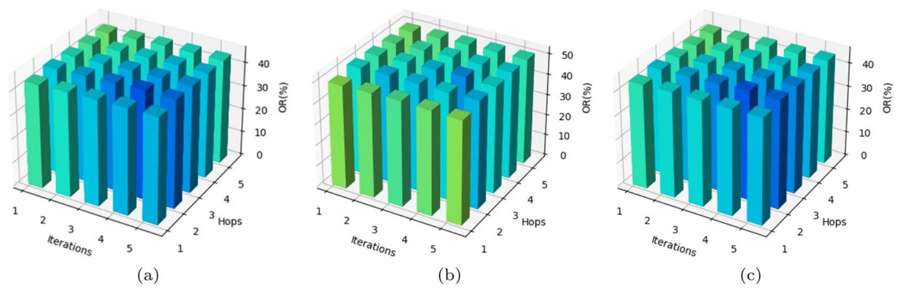


Fig. 6 Comparison of OR for Attention Memory Network module on **a** TAC, **b** Wikidata, and **c** CS datasets with different hops and iterations. The deeper the color, the higher the accuracy

which excels in two aspects. As shown in Table 3, the attention mechanisms in JammyTS improve the flexibility of memory networks and reduce the interference of noise in the Mention Set. Additionally, JammyTS provides a novel method of dividing TSF into two subtasks and predicts fine-grained influences in determining the temporal ranges using different neural networks.

Second, it is noteworthy that memory network-based approaches, such as MemTimes and JammyTS, consistently outperform traditional methods like BLENDER, UNED, and TSRF. This observation underscores the potential of memory networks in effectively modeling the semantic similarity between triples in knowledge graphs (KGs) and associated facts. However, it is essential to address a limitation of the MemTimes approach. While it successfully captures the semantic similarity through memory networks, it employs a heuristic algorithm to determine the temporal range using a sliding window. This method is susceptible to becoming trapped in local optima and fails to infer non-continuous temporal dimensions. Hence, despite the promising performance of memory network-based methods, it is crucial to address the challenges associated with effectively modeling non-continuous temporal dimensions. This may involve exploring alternative strategies beyond heuristic algorithms to ensure accurate and comprehensive reasoning of temporal ranges in KGs.

Experiments are conducted within the Attention Memory Network module using different numbers of hops and iterations, with results displayed in Fig. 6. It is observed that the accuracy of JammyTS initially increases and then slightly decreases as the number of hops and iterations rises. This indicates that the benefits derived from the number of hops and iterations in this module are limited (Cao et al. 2020).

5.3 Analysis on learning end timestamp (RQ2)

In this section, we examine the effectiveness of our proposed learning end timestamp process in enhancing the accuracy of prediction. This process focuses on identifying the non-continuous temporal dimensions present in temporal KGs. To demonstrate its effectiveness, we specifically analyze the CS dataset, which contains a higher number of non-continuous temporal dimensions. As seen in Fig. 7, we present the distribution of relations in the dataset where this specific scenario occurs, along with the experimental results of various methods. Notably, we observe a correlation between the proportion of triples with non-continuous temporal dimensions and the performance of the JammyTS method. As the proportion of such triples increases, the JammyTS method outperforms other approaches in TSF.

Additionally, the subtask “determining the t^{end}” consists of two components. The first component involves calculating the probability that each fact contains deny phrases, while the second component entails calculating the probability of each fact containing antonym phrases. In order to assess the significance of each probability, we conducted an ablation study for both components, and the results can be found in Table 3. The results reveal that relying solely on a single probability calculation is inadequate. It is imperative to consider both the probability of deny phrases and the probability of antonym phrases to obtain accurate results. Furthermore, the impact of each probability is highly dependent on the unique characteristics and nuances of the dataset under consideration.

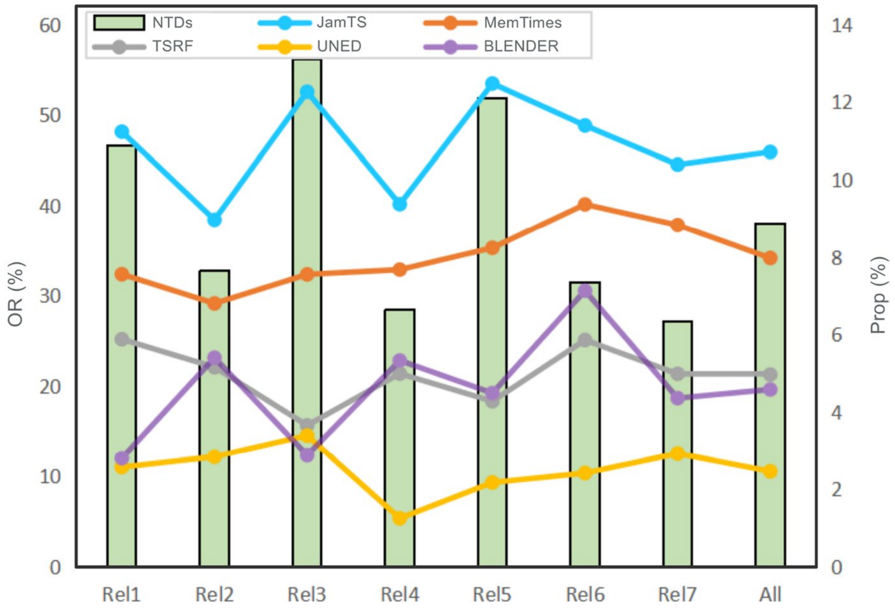


Fig. 7 The OR (%) comparison of different models, and the proportion (Prop) of Non-continuous Temporal Dimensions (NTDs) in each relation and the whole dataset. Curves and bars indicate the values of OR and Prop, respectively

5.4 Analysis on attention (RQ3)

Instead of computing the semantic similarity of both triples and facts and assessing the extent to which facts are noise in a memory network, we propose a novel method that utilizes attention capsules. This method aims to calculate the probability of noise in the timestamp of each fact and the fact itself within the Mention Set. To evaluate the effectiveness of our approach, we conducted an ablation study, the results of which are presented in Table 4. The results demonstrate that the incorporation of attention capsules brings about noteworthy improvements in relation accuracy, particularly in relations like "live_in" and "spouse". While the accuracy improvement in the "live_in" relation is relatively less significant, the accuracy of the "spouse" relation exhibits a substantial enhancement. Upon examining the dataset, we discovered that the "live_in" relation had relatively fewer false timestamps associated with it, while the "spouse" relation possessed a larger number of false timestamps. Furthermore, we observed an overall performance enhancement for each relation after integrating the attention mechanisms into the memory networks. This finding reinforces the effectiveness and value of attention capsules in improving the accuracy and reliability of the relations within JammyTS.

To better demonstrate JammyTS's ability to recognize noise in external facts, we select 10 different external facts from the CS dataset, which included three noise facts (semantically completely distinct from other facts), as shown in Fig. 8a . Additionally, we chose five external facts that conveyed similar semantics along with their corresponding timestamps, one of which included a noise timestamp, as shown in Fig. 8b . Finally, we visualize the attention distribution of the Noise Detection module for these two types of inputs, as depicted in Fig. 9. We can observe that the noise facts or timestamps receive lower attention scores among all inputs, resulting in higher probabilities of being identified as noise.

5.5 Ablation study (RQ4)

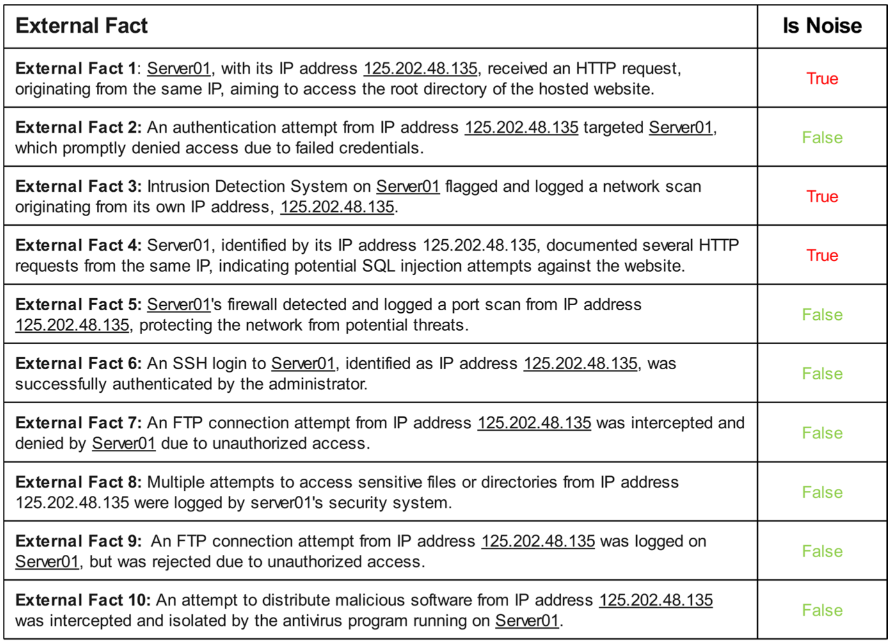
Table 5 shows the OR obtained by selecting different probability combinations. We can observe that when *Pn* is not selected, JammyTS only has a suboptimal result on rel_4 (which we analyze in Section 5.4), and JammyTS obtained optimal results on all other relations. In addition, without filtering external resources for noise, removing the two signs indicating the end of relations will result in a decrease in performance. The results when only selecting *Pn* are generally not as good as when only selecting

Table 4 The OR (%) comparison of different attention-using states on TAC dataset

Model	Rel_1	Rel_2	Rel_3	Rel_4	Rel_5	Rel_6	Rel_7	*All*
w/oA, AC	40.19	36.14	39.73	40.01	33.73	38.13	36.07	37.41
w/oA	43.07	38.07	41.73	**45.72**	<u>36.74</u>	<u>41.83</u>	39.73	40.73
w/oAC	<u>44.18</u>	<u>39.83</u>	<u>42.14</u>	<u>43.97</u>	36.72	40.12	<u>41.43</u>	<u>41.02</u>
JammyTS	**48.21**	**45.12**	**50.06**	43.18	**44.27**	**46.80**	**42.62**	**45.79**

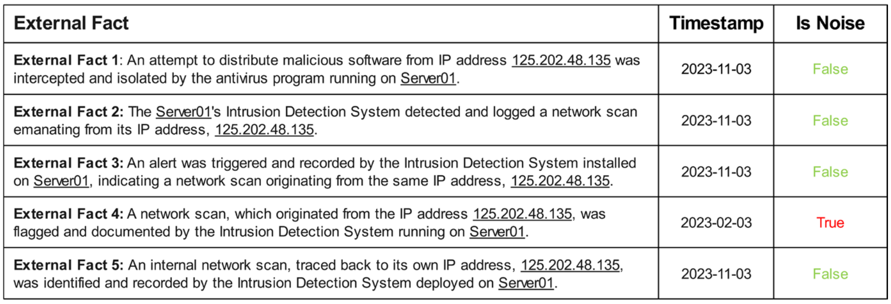
Bold indicates the highest value, and underlining indicates the second highest

"*A*": Attention in the memory network, "*AC*": Attention Capsules



External Fact	Is Noise
External Fact 1: Server01, with its IP address 125.202.48.135, received an HTTP request, originating from the same IP, aiming to access the root directory of the hosted website.	True
External Fact 2: An authentication attempt from IP address 125.202.48.135 targeted Server01, which promptly denied access due to failed credentials.	False
External Fact 3: Intrusion Detection System on Server01 flagged and logged a network scan originating from its own IP address, 125.202.48.135.	True
External Fact 4: Server01, identified by its IP address 125.202.48.135, documented several HTTP requests from the same IP, indicating potential SQL injection attempts against the website.	True
External Fact 5: Server01's firewall detected and logged a port scan from IP address 125.202.48.135, protecting the network from potential threats.	False
External Fact 6: An SSH login to Server01, identified as IP address 125.202.48.135, was successfully authenticated by the administrator.	False
External Fact 7: An FTP connection attempt from IP address 125.202.48.135 was intercepted and denied by Server01 due to unauthorized access.	False
External Fact 8: Multiple attempts to access sensitive files or directories from IP address 125.202.48.135 were logged by server01's security system.	False
External Fact 9: An FTP connection attempt from IP address 125.202.48.135 was logged on Server01, but was rejected due to unauthorized access.	False
External Fact 10: An attempt to distribute malicious software from IP address 125.202.48.135 was intercepted and isolated by the antivirus program running on Server01.	False

(a)



External Fact	Timestamp	Is Noise
External Fact 1: An attempt to distribute malicious software from IP address 125.202.48.135 was intercepted and isolated by the antivirus program running on Server01.	2023-11-03	False
External Fact 2: The Server01's Intrusion Detection System detected and logged a network scan emanating from its IP address, 125.202.48.135.	2023-11-03	False
External Fact 3: An alert was triggered and recorded by the Intrusion Detection System installed on Server01, indicating a network scan originating from the same IP address, 125.202.48.135.	2023-11-03	False
External Fact 4: A network scan, which originated from the IP address 125.202.48.135, was flagged and documented by the Intrusion Detection System running on Server01.	2023-02-03	True
External Fact 5: An internal network scan, traced back to its own IP address, 125.202.48.135, was identified and recorded by the Intrusion Detection System deployed on Server01.	2023-11-03	False

(b)

Fig. 8 The raw text of the input for the Noise Detection module, including **a** only facts and **b** facts with timestamps. The bold text in the sentences are entities

Pa and *Pd*, indicating that our emphasis on inferring non-continuous temporal ranges is meaningful.

6 Conclusion

This paper introduces JamTS, a model for TSF. Leveraging the attention mechanism, the model enhances the performance of memory networks in temporal range delineation tasks by computing the attention distribution of query triples on external resources within storage components. Additionally, through an attention capsule network, it analyzes the noise probability of each external fact based on the atten-

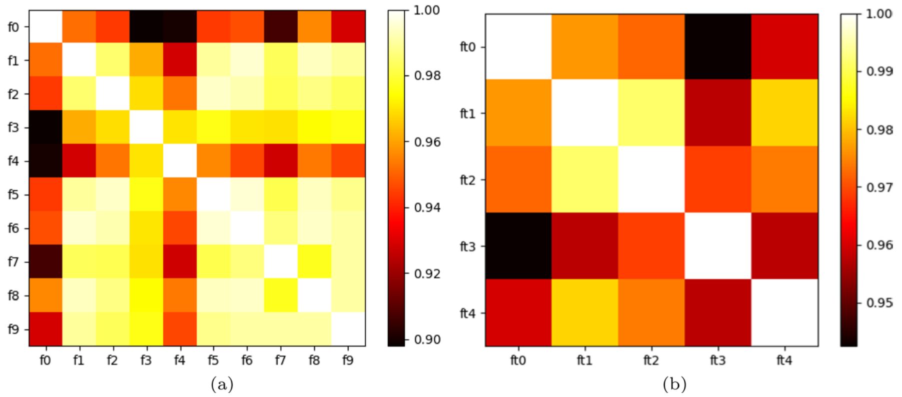


Fig. 9 Attention heat map when inputting **a** only facts and **b** facts with timestamps into the Noise Detection module

Table 5 The OR (%) comparison of containing different probabilities on TAC dataset

Pm	*Pn*	*Pa*	*Pd*	Rel_1	Rel_2	Rel_3	Rel_4	Rel_5	Rel_6	Rel_7	*All*
✓		✓	✓	45.18	39.83	42.14	**43.97**	36.72	40.12	41.43	41.02
✓		✓		44.63	38.41	41.27	42.61	34.78	39.52	40.41	40.46
✓			✓	43.52	36.44	40.75	42.04	33.54	37.58	38.74	39.85
✓	✓			45.34	37.15	40.15	42.42	33.42	38.15	38.56	39.15
✓				42.15	35.56	39.54	41.24	33.08	36.92	37.18	38.03
JammyTS				**48.21**	**45.12**	**50.06**	43.18	**44.27**	**46.80**	**42.62**	**45.79**

Bold indicates the highest value, and underlining indicates the second highest

"*Pm*": similarity probability, "*Pn*": noise probability, "*Pa*": antonym phrase probability, "*Pd*": deny phrase probability

tion distribution of target external facts in the Mention Set. Furthermore, we propose a learning method tailored for non-continuous time ranges. Specifically, the model learns and effectively infers temporal dimension information containing non-linear time ranges via two linear classifiers. Experimental results on three datasets thoroughly validate the effectiveness of JamTS in TSF.

7 Limitation

There are still some limitations that need to be addressed in JammyTS. Firstly, JammyTS demonstrates stable performance when inferring one or two periods of the temporal ranges. However, its effectiveness in handling a larger number of periods is yet to be explored and improved. Secondly, JammyTS currently only takes into account two signs of relation endings, overlooking other relevant signs that exist in real-world scenarios. It is crucial to consider and incorporate these additional signs to enhance the model's comprehensiveness and practicality.

Author contributions Chenxi Hu, Tao Wu, Chunsheng Liu, and Chao Chang conceived and designed the experiments. Chenxi Hu and Chunsheng Liu performed the experiments. Chenxi Hu and Tao Wu prepared the figures and processed the data. Chenxi Hu wrote the main manuscript. All the authors contributed to the discussion and revision of the content.

Funding Open access funding provided by The Hong Kong Polytechnic University

Data availability Data is provided within the manuscript.

Declarations

Conflict of interest The authors declare no competing interests.

References

Bae S, Kyung D, Ryu J, Cho E, Lee G, Kweon S, Oh J, Ji L, Chang E, Kim T, Choi E (2023) Ehrxqa: a multi-modal question answering dataset for electronic health records with chest x-ray images. Adv Neural Inf Process Syst 36:3867–3880

Bokkem D, Hemel M, Dumancic S, Yorke-Smith N (2023) Embedding a long short-term memory network in a constraint programming framework for tomato greenhouse optimisation. Proceedings of the AAAI conference on artificial intelligence, 15731–15737

Cao S, Yang Q, Li Z, Liu G, Zhang D, Xu J (2020) Memtimes: Temporal scoping of facts with memory network. In: Nah Y, Cui B, Lee S-W, Yu JX, Moon Y-S, Whang SE (eds) Database Systems for Advanced Applications. Springer International Publishing, Berlin Heidelberg, pp 70–86

Cong X, Yu B, Fang M, Liu T, Yu H, Hu Z, Huang F, Li Y, Wang B (2023) Universal information extraction with meta-pretrained self-retrieval. Find Assoc Comput Linguist ACL 2023:4084–4100

Dasgupta SS, Ray SN, Talukdar P (2018) HyTE: Hyperplane-based temporally aware knowledge graph embedding. In: proceedings of the 2018 conference on empirical methods in natural language processing, pp. 2001–2011

Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186

Dhingra B, Cole JR, Eisenschlos JM, Gillick D, Eisenstein J, Cohen WW (2022) Time-aware language models as temporal knowledge bases. Trans Assoc Comput Linguist 10:257–273

Dikeoulias I, Amin S, Neumann G (2022) Temporal knowledge graph reasoning with low-rank and model-agnostic representations. In: proceedings of the 7th workshop on representation learning for NLP, pp. 111–120

Fu G, Meng Z, Han Z, Ding Z, Ma Y, Schubert M, Tresp V, Wattenhofer R (2022) TempCaps: A capsule network-based embedding model for temporal knowledge graph completion. In: proceedings of the sixth workshop on structured prediction for NLP, pp. 22–31

Fu S, Zhong S, Lin L, Zhao M (2022) A novel time-series memory auto-encoder with sequentially updated reconstructions for remaining useful life prediction. IEEE Trans Neural Netw Learn Syst 33(12):7114–7125

Gao Y, He Y, Kan Z, Han Y, Qiao L, Li D (2023) Learning joint structural and temporal contextualized knowledge embeddings for temporal knowledge graph completion. Find Assoc Comput Linguist ACL 2023:417–430

Guillermo Garrido, Bernardo Cabaleiro, Anselmo Penas, ,Alvaro Rodrigo, Damiano Spin (2011) A distant supervised learning system for the tac-kbp slot filling and temporal slot filling tasks. In: proceedings of the fourth text analysis conference, TAC 2011, Gaithersburg, Maryland, USA, November 14-15, 2011

Gupta D, Berberich K (2018) Identifying time intervals for knowledge graph facts, pp. 37–38

Han Z, Zhang G, Ma Y, Tresp V (2021) 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, pp. 8104–8118

Hoffart J, Suchanek FM, Berberich K, Weikum G (2013) Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artif Intell 198:28–61

Hu X, Chen X, Qi P, Kong D, Liu K, Wang WY, Huang Z (2023) Language agnostic multilingual information retrieval with contrastive learning. Find Assoc Comput Linguist ACL 2023:9133–9146

Kazemi SM, Goel R, Eghbali S, Ramanan J, Sahota J, Thakur S, Wu S, Smyth C, Poupart P, Brubaker M (2019) Time2Vec: Learning a Vector Representation of Time

Lai T, Tran QH, Bui T, Kihara D (2019) A gated self-attention memory network for answer selection. In: proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp. 5953–5959

Leblay J, Chekol MW (2018) Deriving validity time in knowledge graph. Companion Proceedings of the The Web Conference 2018:1771–1776

Li Z, Guan S, Jin X, Peng W, Lyu Y, Zhu Y, Bai L, Li W, Guo J, Cheng X (2022) Complex evolutional pattern learning for temporal knowledge graph reasoning. In: proceedings of the 60th annual meeting of the association for computational linguistics (Volume 2: Short Papers), pp. 290–296

Lin P, Yang M, Lai J (2021) Deep selective memory network with selective attention and inter-aspect modeling for aspect level sentiment classification. IEEE/ACM Trans Audio Speech Lang Process 29:1093–1106

Li J, Su X, Gao G (2023) TeAST: Temporal knowledge graph embedding via archimedean spiral timeline. In: proceedings of the 61st annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 15460–15474

Li Y, Sun S, Zhao J (2022) Tirgn: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In: proceedings of the thirty-first international joint conference on artificial intelligence, IJCAI-22, pp. 2152–2158

Messner J, Abboud R, Ceylan II (2022) Temporal knowledge graph completion using box embeddings. Proceedings of the AAAI conference on artificial intelligence, 7779–7787

Rula A, Palmonari M, Rubinacci S, Ngomo A-CN, Lehmann J, Maurino A, Esteves D (2019) Tisco: Temporal scoping of facts. Journal of Web Semantics, 72–86

Sil A, Cucerzan S-P (2014) Towards temporal scoping of relational facts based on Wikipedia data. In: proceedings of the eighteenth conference on computational natural language learning, pp. 109–118

Souza M (2021) Automatic design of heuristic algorithms for binary optimization problems. In: proceedings of the thirtieth international joint conference on artificial intelligence, IJCAI-21, pp. 4881–4882

Talukdar PP, Wijaya D, Mitchell T (2012) Coupled temporal scoping of relational facts. In: proceedings of the Fifth ACM international conference on web search and data mining, pp. 73–82

Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jiawei Han, Dan Roth, Jing Zheng (2011) Cuny-uiuc-sri tac-kbp2011 entity linking system description. In: proceedings of the fourth text analysis conference, TAC 2011, Gaithersburg, Maryland, USA, November 14-15, 2011

Vrandečić D, Krötzsch M (2014) Wikidata: a free collaborative knowledgebase. Commun. ACM, 78–85

Wang X, Zhang H, Li Q, Shi Y, Jiang M (2019) A novel unsupervised approach for precise temporal slot filling from incomplete and noisy temporal contexts. In: the world wide web conference, pp. 3328–3334

Wijaya DT, Nakashole N, Mitchell TM (2014) CTPs: Contextual temporal profiles for time scoping facts using state change detection. In: proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1930–1936

Xu W, Liu B, Peng M, Jia X, Peng M (2023) Pre-trained language model with prompts for temporal knowledge graph completion. Find Assoc Comput Linguist ACL 2023:7790–7803

Yuan L, Li Z, Qu J, Zhang T, Liu A, Zhao L, Chen Z (2022) Trhyte Temporal knowledge graph embedding based on temporal-relational hyperplanes. In: Bhattacharya, A., Lee Mong Li, J., Agrawal, D., Reddy, P.K., Mohania, M., Mondal, A., Goyal, V., Uday Kiran, R. (Eds) Database Systems for Advanced Applications. Springer International publishing. Berlin Heidelberg. pp. 137–152

Yu X, Min S, Zettlemoyer L, Hajishirzi H (2023) CREPE: Open-domain question answering with false presuppositions. In: proceedings of the 61st annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 10457–10480

Zhang M, Xia Y, Liu Q, Wu S, Wang L (2023) Learning latent relations for temporal knowledge graph reasoning. In: proceedings of the 61st annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp. 12617–12631