TempCaps: A Capsule Network-based Embedding Model for Temporal Knowledge Graph Completion

Guirong Fu[*1], **Zhao Meng**[*1], **Zhen Han**[*2,3], **Zifeng Ding**[2,3], **Yunpu Ma**[2], **Matthias Schubert**[2] **Volker Tresp**[2,3] **Roger Wattenhofer**[1]
[1]ETH Zürich [2]LMU Munich [3]Siemens AG
{fug, zhmeng, wattenhofer}@ethz.ch, zhen.han@campus.lmu.de
{zifeng.ding,volker.tresp}@siemens.com,
cognitive.yunpu@gmail.com, schubert@dbs.ifi.lmu.de

Abstract

Temporal knowledge graphs store the dynamics of entities and relations during a time period. However, typical temporal knowledge graphs often suffer from incomplete dynamics with missing facts in real-world scenarios. Hence, modeling temporal knowledge graphs to complete the missing facts is important. In this paper, we tackle the temporal knowledge graph completion task by proposing **TempCaps**, which is a **Caps**ule network-based embedding model for **Temp**oral knowledge graph completion. TempCaps models temporal knowledge graphs by introducing a novel dynamic routing aggregator inspired by Capsule Networks. Specifically, TempCaps builds entity embeddings by dynamically routing retrieved temporal relation and neighbor information. Experimental results demonstrate that TempCaps reaches state-of-the-art performance for temporal knowledge graph completion. Additional analysis also shows that TempCaps is efficient[1].

1 Introduction

Knowledge graphs (KGs) organize and integrate information in a structured manner, which is human-readable and suitable for computer processing. This advantage of knowledge graphs is helping to bridge the gap between humans and computers. Numerous real-world applications have benefited from KGs. In particular, recent advances in artificial intelligence have motivated researchers to use knowledge graphs to boost performance in downstream applications, including natural language processing (IV et al., 2019; Bosselut et al., 2019) and computer vision (Yu et al., 2021; Marino et al., 2017).

Despite the usefulness of knowledge graphs, existing knowledge graphs are often incomplete, which means important facts might be missing. To tackle this problem, researchers have developed various methods for the task of knowledge graph completion (Nickel et al., 2011; Bordes et al., 2013), aiming to recover missing facts for existing knowledge graphs. In particular, Nguyen et al. (2019) explored the Capsule Network (CapsNet) (Sabour et al., 2017) for modeling knowledge graphs. CapsE(Nguyen et al., 2019) demonstrate that each dimension of the entity, as well as relation, embeddings also have diverse variations in different contexts. Thus, they used capsules to encode many characteristics in the embedding triple and represent the entries at the corresponding dimension, showing superior performance to other KG models.

Existing studies, including CapsE, focus on completing static knowledge graphs. In reality, however, multi-relational data is often time-dependent. Moreover, static knowledge graphs fail to adequately describe the changing essence of the world, indicating that knowledge or facts being true in the past might not always stay true. For instance, social networks constantly change. Static knowledge graphs fail to model these changes. To this end, temporal knowledge graphs (tKGs) are introduced to grasp these dynamic changes. Specifically, temporal facts are represented as a quadruple by extending the static triplet with a timestamp describing when these facts occurred, i.e. (Barack Obama, inaugurated, president of the US, 2009). Similar to static KG, tKGs also suffer from the problem of incompleteness, making the task of temporal knowledge graph completion eminent (Bordes et al., 2013; Lin et al., 2015).

In this paper, we take advantage of the Capsule Network paradigm and generalize it for modeling tKGs. We introduce TempCaps, which is a **Caps**ule network-based embedding model for **Temp**oral knowledge graph completion. As shown in Figure 1, TempCaps consists of a neighbor selector, an

[*]The first three authors contributed equally to this work.
[1]Our code is available at https://github.com/fuguigui/tempcaps

entity embedding layer, a dynamic routing aggregator and a multi-layer perceptron (MLP) decoder. Unlike CapsE, we incorporate the temporal information of tKGs into our model: First, we pose temporal constraints on neighbor selection by introducing a time window. At a given time step, we only take the neighbors that interact with the source entity within the time window into account for capturing the entity features. Second, we propose a time-dependent dynamic routing mechanism that incorporates time information into routing weight matrix. Third, we exploit the temporal weighting vectors generated during the dynamic routing to calculate the output probability, which reflects how tightly lower capsules connect with higher capsules.

Our contributions are in the following: **(i)** We propose TempCaps, which leverages Capsule Networks by dynamically routing retrieved temporal relations and neighboring entities. An advantage of our model is that different capsules can capture different aspects of the same entity. Such advantage is important for modeling temporal knowledge graphs, which are dynamic, and often involve one entity in multiple timestamps. **(ii)** Our TempCaps improves the performance of temporal knowledge graph completion. Experimental results show that our model achieves state-of-the-art performance on the GDELT and ICEWS datasets. Furthermore, our model is light-weighted and efficient compared to previous methods for modeling tKGs. **(iii)** As far as we know, we are the first to use Capsule Networks for tKGs. Our experiments show that by leveraging dynamic routing, TempCaps is suitable for both discrete and continuous timestamps and can be easily generalized to unseen timestamps. **(iv)** We conduct additional ablation studies to understand how each part of TempCaps contributes to the model performance. We also show that TempCaps is efficient by analyzing time and space complexity.

2 Related Work

2.1 Knowledge Graph Embedding

Knowledge Graph Embedding (KGE) maps entities and relations into low-dimensional continuous vectors. Two types of KGEs, including static KGE and temporal KGE, have attracted attention from the community. In the rest of this subsection, we give an overview of static and temporal KGE.

Static Knowledge Graph Embedding. Embedding approaches for static KGs can generally be categorized into bilinear models and transition-based models. TransE (Bordes et al., 2013) leverages the transition-based approach, which measures the plausibility of a triple as the distance between the object entity's embedding and the embedding of the subject after the relational transition. Similarly, by using additional projection vectors, Wang et al. (2014) extend TransE to translate entity embeddings into the vector space of relations. Other works including RESCAL (Nickel et al., 2011), DisMult (Yang et al., 2015), and SimplE (Kazemi and Poole, 2018) use a bilinear score function, which represents predicates as linear transformations of entity embeddings. However, these KGE methods are not suitable for tKGs as they cannot capture the temporal dynamics of tKGs.

Temporal Knowledge Graph Embedding. Temporal KGE approaches aim to capture both temporal and relational information to improve the performance of the completion task. Han et al. (2021b) assessed well-known temporal embeddings of tKGE models via an extensive experimental study and released the first open unified open-source framework for temporal KG completion models with full composability. HyTE (Dasgupta et al., 2018) embeds time information in the entity-relation space by arranging a temporal hyperplane to each timestamp and uses TransE as interaction model to compute the plausibility score of facts. DE-SimplE (Goel et al., 2020) extends SimplE by exploring the diachronic function to model entity embeddings at different timestamps. TA-DistMult (García-Durán et al., 2018) utilizes recurrent neural networks to learn time-aware representations of relations and adopt DistMult as the score function. Moreover, Han et al. (2020a) introduced a non-Euclidean embedding approach that learns evolving entity representations in a product of Riemannian manifolds. Besides, Han et al. (2022) enhanced temporal knowledge embedding using contextualized language representations and achived state-of-the-art results. Besides the completion task, researchers have also paid attention to use temporal KGE for forecasting on tKGs (Trivedi et al., 2017; Jin et al., 2020; Han et al., 2020b,c, 2021a; Sun et al., 2021). Forecasting tasks predict future links based on past observations, while the completion tasks interpolate missing links at observed timestamps. In this work, we focus on the tKG completion task.

2.2 Capsule Network

Sabour et al. (2017) propose Capsule Networks to capture different entities in images by leveraging dynamic routing between different layers of Capsule Networks. As a result, capsule Networks reach comparable or even better performance when compared to convolutional neural networks, while at the same time being more efficient and more robust to affine transformation. Following Sabour et al. (2017), researchers have proposed various methods to improve the performance of Capsule Networks. Hahn et al. (2019) boost the performance of Capsule Networks by using a novel self-routing mechanism. Tsai et al. (2020) propose to use inverted dot-product attention routing to improve Capsule Networks. We give more details on the basics of Capsule Networks in Section 3.2.2.

Apart from the vision domain, previous work has shown that Capsule Networks are also useful for modeling static knowledge graphs. (Nguyen et al., 2019) propose CapsE, which represents each triplet fact *(subject, relation, object)* in a knowledge graph as a 3-column matrix, each of which corresponds to an entity in a fact. CapsE reaches state-of-the-art performance on static knowledge graph completion tasks.

This paper proposes TempCaps, which uses Capsule Networks to model tKGs. Despite all previous works on Capsule Networks, we are the first to model tKGs with Capsule Networks to the best of our knowledge. Experimental results show that TempCaps achieves competitive performance on the temporal knowledge graph completion task. We present the details of TempCaps in Section 3.2.

3 Methodology

3.1 Task Formulation

A temporal knowledge graph (tKG) is a collection of valid facts with temporal information. A fact in tKG is a quadruple of (s, r, o, t), which consists of subject s, relation r, object o, and timestamp t. We use E, R, and T to denote the sets of entities, relations, and timestamps involved in at least one fact in a given tKG. $|E|$, $|R|$ and $|T|$ are the number of elements in each set, respectively. A tKG can be viewed as the union of KG snapshots at each timestamp. Formally, we have:

$$G = G(t_1) \cup G(t_2) \cup \cdots G(t_i) \cdots \cup G(t_{max}),$$

where $G(t_i) = \{(s, r, o, t_i) | t_i \in T\}$ is a snapshot of G at timestamp t_i, and $t_{max} = \max(t_i | t_i \in T)$.

Temporal Knowledge Graph Completion (TKGC) aims to predict unobserved missing facts from incomplete tKGs. In TKGC, both unobserved and observed facts share the same period of time. Let O be the observed true facts from a complete tKG G (G contains both observed true facts and to-be-predicted facts), we denote the set of missing facts as $\bar{O} = G \setminus O$ which should be predicted in the context of TKGC. In our work, we only consider predicting the missing subject or the missing object of the missing facts. For every missing fact $(s, r, o, t) \in \bar{O}$, two prediction queries $(s, r, ?, t)$ and $(?, r, o, t)$ are generated, and our model aims to rank the ground-truth subject entity s from $(?, r, o, t)$, as well as the ground-truth object entity o from $(s, r, ?, t)$, as high as possible among all candidate entities. For simplicity, we present the equations and illustrate our method with only object prediction. During training and evaluation of our experiments, we include both subject prediction and object prediction.

3.2 Model Architecture

3.2.1 Overview

We propose TempCaps, a **Caps**ule network-based embedding model for **Temp**oral knowledge graph completion. TempCaps first selects two types of neighboring entities, i.e., local entities and global relational entities, for each entity of the tKG. Then it learns the embeddings of entities based on the retrieved neighbors using a dynamic routing module (see Section 3.2.5). Finally, TempCaps ranks the entities from the candidate set by feeding the embeddings of the entities to a scoring module. Figure 1 gives an illustration of TempCaps.

3.2.2 Capsule Network

Capsule networks are built with two critical components: capsules and the dynamic routing mechanism.

A capsule is a set of neurons processing different information about an entity, and the activities of the neurons within an active capsule represent the various properties of a particular entity (Sabour et al., 2017). We use a squash function proposed by Sabour et al. to guarantee that the length of the

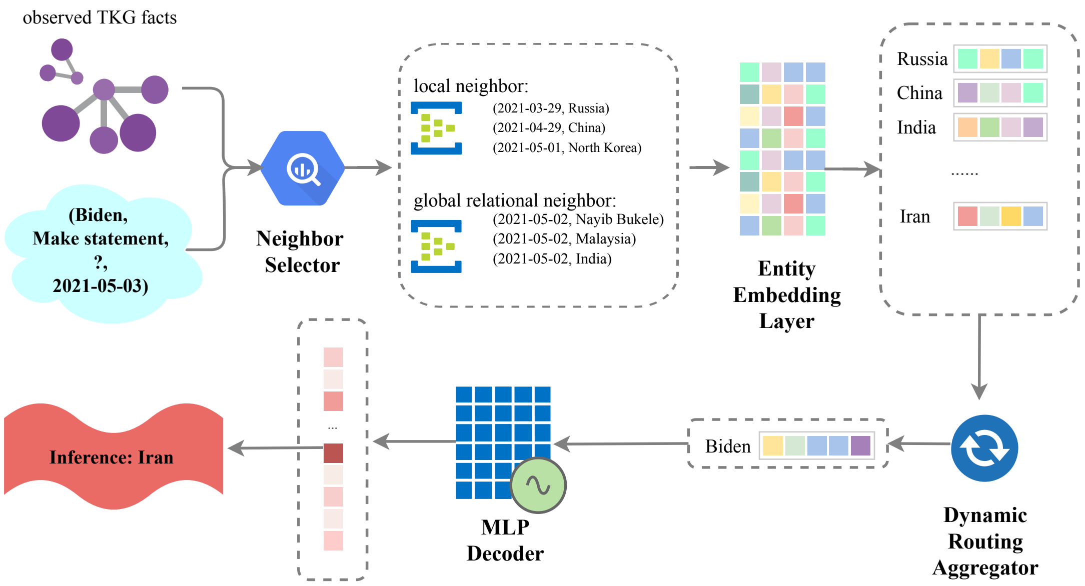


Figure 1: Overview of TempCaps. Assume we want to predict the ground truth object of a prediction query (Biden, Make statement, ?, 2021-05-03), given all the observed facts. TempCaps first selects different types of neighboring entities of the query subject Biden, and embeds these neighbors with capsules. Then it utilizes the dynamic routing aggregator to learn Biden's contextualized embedding. A multi-layer perceptron (MLP) decoder takes the learned embedding and performs a multi-class classification over all candidates, producing scores for every entity in the candidate set. The entity with the highest score (Iran in this example) is the predicted object.

vector stays between 0 and 1:

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}, \tag{1}$$

where $\mathbf{s}_j$ is the input of a capsule and $\mathbf{v}_j$ is its squashed output.

Routing by agreements regulates how capsules communicate between layers. The dynamic routing mechanism (Sabour et al., 2017) works as follows. All output vectors $\mathbf{u}_i$ of capsules in the lower layer are first multiplied by a weight matrix $\mathbf{W}_{ij}$. Then, the weighted sum of newly obtained vectors are input into a capsule $\mathbf{s}_j$ in the next layer:

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i, \quad \mathbf{s}_j \quad = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}, \tag{2}$$

where c_{ij} is the coupling coefficient between capsule i and capsule j. In our work, we initialize each entity's embedding with a capsule in the first capsule layer. By performing routing by agreements, we achieve information aggregation between an entity and its selected neighbors.

3.2.3 Neighbor Selector

Similar to static KGs, in tKGs, we can still treat entities as nodes (relations as edges). Inspired by previous works in graph neural network (Kipf and Welling, 2017; Velickovic et al., 2018; Xu et al., 2019), where the embeddings of nodes are derived by the n-hop neighbors of the nodes, TempCaps computes the embedding of each node, i.e., entity in the context of tKGs, by leveraging information from the temporal neighbors of that node in the tKG. Given a prediction query $(s, r, ?, t)$, TempCaps selects two types of neighbors, namely, local entities and global relational entities, for the query subject s.

A **local entity** is an object entity o' which originates from an observed fact (s, r, o', t'), where t' can be any timestamp within a fixed range around the query timestamp. We denote the set of all local entities at all timestamps as $E^l(s, r)$:

$$E^l(s,r) = \{o'|(s,r,o',t'), \\ \mathtt{max}(t - \Delta t_e, t_1) \leq t' \leq \mathtt{min}(t + \Delta t_e, t_{max})\}.$$

To avoid including excessive entities into E^l, TempCaps samples local entities from all observed facts within a pre-defined time window Δt_e.

A **global relational entity** is an object entity o' which originates from an observed fact (s', r, o', t'), where s' can be any entity and t' can be any timestamp within a fixed range around the query timestamp. We denote the set of all local entities at all timestamps as E^g:

$$E^g(r) = \{o' | (s', r, o', t'),$$
$$\texttt{max}(t - \Delta t_r, t_1) \le t' \le \texttt{min}(t + \Delta t_r, t_{max})\}.$$

Similarly, global relational entities are selected within a time window Δt_r.

We further define the set of all selected neighbors as $E^n = \{E^l, E^g\}$. By restricting neighbors within time windows around the query timestamp, TempCaps selects entities that have greater influence on the query subject s. We employ different time windows to select local entities, and global relational entities as different types of neighbors have different influence on the query subject s. We treat the time windows, i.e., Δt_e and Δt_r, as hyperparameters during finetuning.

3.2.4 Temporal Weighting Function

In CapsNet (Sabour et al., 2017), the log prior probability b_{ij} between two capsules i and j are learned depending on the locations and the types of both capsules. It is used to compute the coupling coefficient stated in Equation 2: $c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}$. Inspired by CapsNet, we initialize the log prior probability between the query subject s and its selected neighbor o' with a temporal weighting function, as we consider the time difference between these two entities as the difference of capsule locations. The intuition is that, for a prediction query $(s, r, ?, t)$, a neighbor that connects with s near to t should have more influence on s than a temporally-farther neighbor. Hence, we assign a higher probability to nearer neighbors than farther neighbors. Formally, given a prediction query $(s, r, ?, t)$ and a selected neighbor o' (derived from an observed fact at t'), $b_{o'}$ is initialized as:

$$b_{o'} = \frac{\gamma + 1}{\gamma + |t' - t| + 1}, \tag{3}$$

where γ is a hyperparameter. Figure 2 illustrates the temporal weighting function with different γ. The temporal weighting function with a lower γ leads to higher differences in the values of coupling coefficients regarding various neighboring entities.

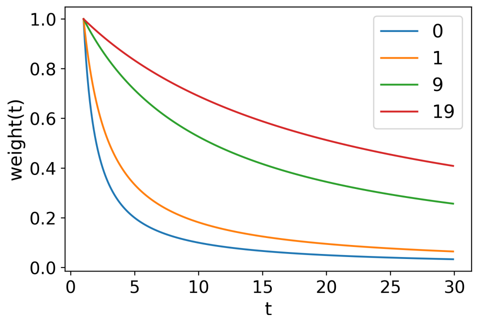


Figure 2: Temporal weighting function with different γ. The horizontal axis is t and the vertical axis is the value of $\texttt{weight}(t)$.

3.2.5 Dynamic Routing Aggregator

Based on the selected neighboring entities from the neighbor selector, TempCaps then learns the representation of an entity by leveraging a dynamic routing aggregator. Inspired by CapsE (Nguyen et al., 2019) that uses Capsule Networks to model static KGs, we design two layers of capsules for TempCaps, and then apply a modified dynamic routing algorithm. The first capsule layer consists of N capsules, where N is the number of the selected neighboring entities from the neighbor selector. Assume we have a prediction query $(s, r, ?, t)$, and for the query subject s, we have the selected neighbors E^n. For every neighboring entity $e \in E^n$, a capsule maps its embedding $\mathbf{u}^{(0)}$ with a multi-layer perceptron to obtain $\mathbf{u}^{(1)}$. Then in the second capsule layer, we use the dynamic routing algorithm to compute the contextualized representation $\mathbf{e}_s$ of the query subject s. Let $\sigma(\cdot)$ be an activation function, we use the following functions to compute contextualized representations:

$$\mathbf{u}_i^{(1)} = \sigma(\mathbf{W}\mathbf{u}_i^{(0)} + \epsilon), \tag{4}$$
$$\mathbf{e} = \texttt{DynamicRouting}(\mathbf{u}_1^{(1)}, \cdots, \mathbf{u}_N^{(1)}), \tag{5}$$

where W is the weighting matrix, ϵ is a bias, and N is the number of selected neighbors. Algorithm 1 shows the details of the dynamic routing module.

3.2.6 MLP Decoder

The multi-layer perceptron (MLP) decoder takes the representation $\mathbf{e}$ from the dynamic routing module as the input and estimates the probabilities of all candidates being the predicted answer by leveraging a softmax function:

$$P_{\text{MLP}}(o|s, r, t) = \frac{\exp(\sigma(\mathbf{W}_{\text{MLP}}\mathbf{e}_o + \epsilon_{\text{MLP}}))}{\sum_{o' \in E} \exp(\sigma(\mathbf{W}_{\text{MLP}}\mathbf{e}_{o'} + \epsilon_{\text{MLP}}))}, \tag{6}$$

where $\mathbf{W}_{\text{MLP}}$ is a weight matrix, $\epsilon_{\text{MLP}} \in \mathbb{R}^{|E|}$ is a bias vector, and $\sigma(\cdot)$ is the activation function.

Algorithm 1: Modified dynamic routing algorithm

```
input  : $\{\mathbf{u}_i^{(0)}\}$, number of iteration $m$
output : $\mathbf{e}, \mathbf{c}$
for all capsule $i \in$ first capsule layer do
    $b_i \leftarrow \texttt{weight}(t_i)$
end
for $m$ iterations do
    for all capsule $i \in$ first capsule layer do
        $c_i \leftarrow \frac{\exp(b_i)}{\sum_k \exp(b_k)}$;
    end
    for all capsule $i \in$ second capsule layer do
        $\mathbf{s} \leftarrow \sum_i c_i \mathbf{u}_i^{(0)}$;
    end
    for all capsule $i \in$ second capsule layer do
        $\mathbf{e} \leftarrow \texttt{squash}(\mathbf{s})$;
    end
    for all capsule $i \in$ first capsule layer do
        $b_i \leftarrow b_i + \mathbf{u}_i^{(0)\intercal} \cdot \mathbf{e}$
    end
end
```

3.2.7 Parameter Estimation and Inference

Following previous works about tKG reasoning (Jin et al., 2020; Zhu et al., 2021), we treat temporal knowledge graph completion as a multi-class classification task, where each class corresponds to a candidate entity. The learning objective is to minimize the negative log-likelihood L on all observed facts with the object (or subject) masked during training:

$$L = -\sum_{(s,r,o,t)\in G} \log[P(o|s,r,t)], \tag{7}$$

where $P(o|s,r,t) = (1-\alpha)\cdot P_{\texttt{MLP}}(o|s,r,t) + \alpha \cdot P_{\texttt{DyR}}(o|s,r,t)$ is the probability of the entity o being the ground truth missing object given $(s,r,?,t)$. This probability consists of two parts: $P_{\texttt{MLP}}(o|s,r,t)$ and $P_{\texttt{DyR}}(o|s,r,t)$, where $P_{\texttt{MLP}}(o|s,r,t)$ is defined by Equation 6 and $P_{\texttt{DyR}}(o|s,r,t)$ is the softmax output c from the last iteration of Algorithm 1. For the entities not selected into the set of neighbors, we force the value of their $P_{\texttt{DyR}}$ to 0. $\alpha \in [0,1]$ is the balancing parameter that controls the importance of each probability term.

During inference time, for a prediction query $(s,r,?,t)$, we follow the training process and retrieve the combined probabilities of all entities. The candidate entity with the highest combined probability is selected as the model prediction:

$$o_{\texttt{pred}} = \arg\max_{o'\in E} P(o|s,r,t). \tag{8}$$

The learning objective for subject prediction is similar. We omit it in the paper for simplicity.

4 Experimental Results

4.1 Experimental Setup

Datasets We use three datasets for evaluation in our experiments: Global Database of Events, Language, and Tone (GDELT) (Leetaru and Schrodt, 2013), two subsets of Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015), i.e, ICEWS05-15 and ICEWS14. GDELT collects human societal-scale behaviors and events occurring from April 1, 2015, to March 31, 2016 in news media. The ICEWS dataset records political events with timestamps. ICEWS14 and ICEWS05-15 are two subsets from ICEWS, which contains events in 2014, and from 2005 to 2015, respectively. For all our experiments, we split the dataset by 80%/10%/10% for train/validation/test. Table 2 gives the statistics of the datasets.

Metrics For each fact (s,r,o,t) in the dataset, we create two sub-tasks: (1) predicting the object $(s,r,?,t)$ and (2) predicting the subject $(?,r,o,t)$. We report four metrics for the two tasks separately and take the average between the two sub-tasks. The metrics we used are MRR and Hits@1/3/10. Let |Q| denote the number of queries. MRR, defined as $\frac{1}{|Q|}\sum_i \frac{1}{\text{rank}_i}$, is the average of reciprocal ranks. Hits@K $= \frac{1}{|Q|}\sum_i \mathbb{1}[\text{rank}_i \le K]$ shows the ratio of the cases where the ground-truth entities are ranked within the top K. We filter the candidate object set during evaluation in the same manner as (Goel et al., 2020) do. During the evaluation, in one timestamp, a subject may be connected with multiple objects under the same relation. Hence, objects except the groundtruth o are not necessarily wrong. We therefore filter the candidate set E during evaluation. In other words, instead of considering all the entities E, the model gives the rank of the actual missing object among entities in $o\cup\bar{E}_t$, where $\bar{E}_t$ are entities not connected to s under r at time t. To be specific, $\bar{E}_t = \{o'|(s,r,o',t) \notin G_t\}$.

Baselines We compare the performance of our model with both static and temporal state-of-the-art

Model	GDELT				ICEWS05-15				ICEWS14			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	11.3	0.0	-	15.8	29.4	9.0	-	66.3	28.0	9.4	-	63.7
DistMult	19.6	11.7	20.8	34.8	45.6	33.7	-	69.1	43.9	32.3	-	67.2
SimplE	20.6	12.4	22.0	36.6	47.8	35.9	53.9	70.8	45.8	34.1	51.6	68.7
HyTE	11.8	0.0	16.5	32.6	31.6	11.6	44.5	68.1	29.7	10.8	41.6	65.5
TA-DistMult	20.6	12.4	21.9	36.5	47.4	34.6	-	72.8	47.7	36.3	-	68.6
DE-SimplE	23.0	14.1	24.8	40.3	51.3	39.2	**57.8**	**74.8**	**52.6**	**41.8**	**59.2**	**72.5**
TempCaps	**25.8**	**18.0**	**27.7**	**40.4**	**52.1**	**42.3**	57.6	70.5	48.9	38.8	54.4	67.9

Table 1: Model performance on GDELT, ICEWS05-15 and ICEWS14. We use MRR and Hits@1/3/10 as our evaluation metric. Results of the baseline models are directly adapted from the original papers. "-" indicates the number is not available.

Dataset	#Ent	# Rel	#Train	#Valid	#Test	Gap	#Gaps
ICEWS14	7,128	230	72,826	8,941	8,963	24H	365
ICEWS05-15	10,488	251	368,962	46,275	46,092	24H	4,017
GDELT	500	20	2,735,685	341,961	341,961	24H	366

Table 2: Statistics on datasets. The columns are the name of the dataset, the number of all entities, the number of all relation types, the number of facts in the train/validation/test sets, the time gap, and total time gaps. In the column **Gap**, "H" indicates hours. For example, "24H" means that the difference between two consecutive timestamps is 24 hours.

KG embedding models. The static models include TransE (Bordes et al., 2013), DistMult (Yang et al., 2015) and SimplE (Kazemi and Poole, 2018) while temporal models are HyTE (Dasgupta et al., 2018), TA-DistMult (García-Durán et al., 2018), and DE-SimplE (Goel et al., 2020).

Implementations Details All our experiments are conducted on a single Titan Xp GPU. We use the ADAM optimizer with a weight decay rate of 1e-5. In addition, we set the learning rate to 1e-3, batch size to 300, the initial entity embedding size to 100, the size of the linear transformation in dynamic routing aggregator to 200 $\times$ 100, the routing iteration times as 1, the temporal weighting decay γ to 4, the loss balancing factor α to 0.1 and dropout rate to 0.3. The neighborhood candidate numbers are 80 for local entities and 40 for global relational entities.

4.2 Results

Table 1 gives the results of our model performance. We can observe that our model reaches state-of-the-art performance on the GDELT and ICEWS05-15 datasets. On GDELT, our model outperforms the baseline models on all four metrics. For MRR, our model outperforms the second-best model by 2.8%, and leads Hits@1 by 3.9%. On ICEWS05-15, our model is state-of-the-art on two of the most important metrics, MRR and Hits@1. Additionally, our model leads the second-best model by 3.1% for Hits@1, indicating that our model can retrieve the ground-truth entity with high accuracy.

On ICEWS14, our model is not the best but is still comparable to the state-of-the-art model. For example, our model reaches an MRR of 48.9% on ICEWS14, while the best-performed model DE-SimplE reaches an MRR of 52.6%.

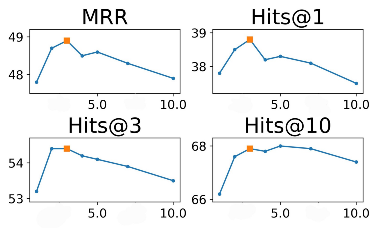
4.3 Ablation Studies

We study the following hyperparameters or design choices on ICEWS14: (1) the number of candidate entities (local/global relational);(2) the length of visible time window (t_r, t_e, t_a); (3) the number of routing iterations; (4) the temporal weighting decay rate γ; (5) whether or not we use an MLP decoder in the final layer of the model; (6) the loss balancing factor α. Table 3 details the results of the ablation studies.

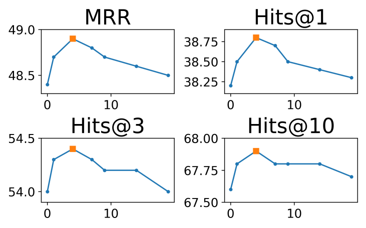
From model variants on the number of candidate numbers, we can see that mixing different types of neighbors is helping. The local entities are particularly helpful, and adding global relational entities further improves the performance.

For the length of visible time window, the optimal number is 6 days ($t_r = t_a = 3\ days$) according to the results in Figure 3(a). We argue that a too-short window results in insufficient information, while a too-long window would contain too much noise, which might harm the model performance.

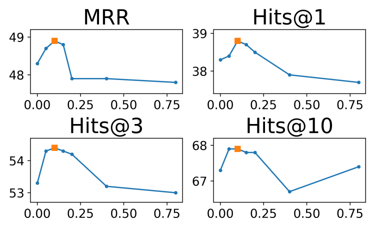
From Figure 4, we can see how the number of routing iterations affects model performance and that the dynamic routing aggregator outperforms the mean aggregator on MRR. Finally, figure 3(b) illustrates the model performance when using different weight decay rates γ, where we can observe that the optimal value of γ is 4. Additionally, we



(a) Time window size (days).



(b) Temporal weighting decay γ.



(c) Balancing factor α.

Figure 3: Ablation studies. The configuration in our final model is marked in orange.

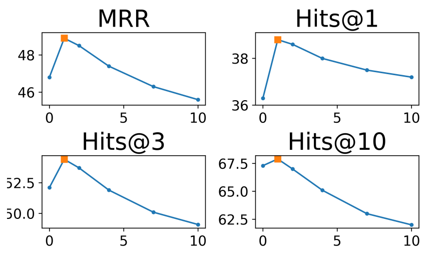


Figure 4: Ablation studies. Effects of the number of iterations. 0 indicates the model uses a mean aggregator, otherwise the model uses the dynamic routing aggregator.

notice that dropping the final MLP decoder results decreases model performance (see Table 3). In Figure 3(c), we show that a loss balance factor $\alpha = 0.1$ leads to better performance than when setting $\alpha = 0$. This indicates that our model benefits from both $P_{\texttt{MLP}}$ and $P_{\texttt{DyR}}$.

4.4 Analysis

We analyze the space and time complexity of our model from the empirical and theoretical points of view. As is shown in Figure 1, the trainable parameters of our model consists of three parts: (1) $\mathbf{E}_1 \in \mathbb{R}^{|E| \times D_1}$ in the entity embedding layer, (2) $\mathbf{W}_1 \in \mathbb{R}^{D_2 \times D_1}$ and $\mathbf{e}_1 \in \mathbb{R}^{D_2}$ in the dynamic routing aggregator and (3) $\mathbf{W}_2 \in \mathbb{R}^{|E| \times D_2}$ and $\mathbf{e}_2 \in \mathbb{R}^{|E| \times D_2}$ in the final MLP decoder. In summary, our model has $O(|E|)$ parameters, which is optimal for representing a knowledge graph with $|E|$ entities. In our experiments, taking the ICEWS14 dataset as an example, each training epoch costs only 54 seconds on average, and the total evaluation process for the testing dataset costs 21 seconds. This indicates our model is efficient both in training and inference and saves considerable time and memory compared to previous works for temporal knowledge graph completion.

Variants		**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Candidates	(120,0)	47.7	37.4	53.2	67.0
	(0,120)	16.7	8.3	18.0	34.8
	(60,60)	48.3	38.1	54.0	67.4
	(80,40)*	**48.9**	**38.8**	**54.4**	**67.9**
	(90,30)	48.6	38.4	54.1	67.7
	(100,20)	48.2	38.0	53.6	67.4
Neighbor length	1	47.8	37.8	53.2	66.2
	2	48.7	38.5	54.4	67.6
	3*	**48.9**	**38.8**	**54.4**	67.9
	4	48.5	38.2	54.2	67.8
	5	48.6	38.3	54.1	**68.0**
	7	48.3	38.1	53.9	67.9
	10	47.9	37.5	53.5	67.4
Iterations	0 (mean)	46.8	36.3	52.1	67.3
	1*	**48.9**	**38.8**	**54.4**	**67.9**
	2	48.5	38.6	53.7	67.0
	4	47.4	38.0	51.9	65.1
	7	46.3	37.5	50.1	63.0
	10	45.6	37.2	49.1	62.0
Weight decay γ	0	48.4	38.2	54.0	67.6
	1	48.7	38.5	54.3	67.8
	4*	**48.9**	**38.8**	**54.4**	**67.9**
	7	48.8	38.7	54.3	67.8
	9	48.7	38.5	54.2	67.8
	14	48.6	38.4	54.2	67.8
	19	48.5	38.3	54.0	67.7
Final MLP decoder	No	48.6	38.6	53.8	67.1
	Yes*	**48.9**	**38.8**	**54.4**	**67.9**
Loss weight α	0	48.3	38.3	53.3	67.3
	0.05	48.7	38.4	54.3	**67.9**
	0.1*	**48.9**	**38.8**	**54.4**	**67.9**
	0.15	48.8	38.7	54.3	67.8
	0.2	48.7	38.5	54.2	67.8
	0.4	47.9	37.9	53.2	66.7
	0.8	47.8	37.7	53.0	67.4
	1.0	0.017	0.017	0.017	0.017

Table 3: The complete results of our ablation studies. * indicates configurations used in our final model.

The space complexity of the embedding computation before aggregation is $O(|B|D_1D_2)$ where $|B|$ is the batch size and D_i is the embedding size defined in the model. Then, the space complexity of going through the dynamic routing aggregator (Algorithm 1) is $O(r|B||C|D_2^2)$, where $|C|$ is the candidate number. At last, the MLP decoder takes another $O(|B||E|D_2)$, where $|E|$ is the total number of entities. Thus, for each epoch of training or testing, the space complexity is $O(|Q|(D_1D_2+|C|D_2^2+|E|D_2))$, which can be simplified as $O(c \cdot |Q||E|)$. Here c is a constant related to pre-defined parameters, and $|Q|$ is the training/testing dataset size.

5 Conclusion

In this paper, we propose TempCaps, which is a light-weighted Capsule Network-based embedding model for temporal knowledge graph completion. TempCaps consists of a neighbor selector, a dynamic routing aggregator, and an MLP decoder. Experimental results show that TempCaps reaches state-of-the-art performance on the GDELT and ICEWS05-15 dataset. We conduct additional ablation studies to understand how each part of TempCaps and hyperparameter choices contribute to the model performance. Our analysis also shows that TempCaps is efficient both in time and space. In the future, we plan to extend TempCaps to forecasting in temporal knowledge graphs.

Acknowledgements

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

References

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *ACL*.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *EMNLP*.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *AAAI*.

Taeyoung Hahn, Myeongjang Pyeon, and Gunhee Kim. 2019. Self-routing capsule networks. In *NeurIPS*.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020a. Dyernie: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. In *EMNLP*.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020b. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *ICLR*.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021a. Temporal knowledge graph forecasting with neural ode. *arXiv preprint arXiv:2101.05151*.

Zhen Han, Ruotong Liao, Beiyan Liu, Yao Zhang, Zifeng Ding, Heinz Köppl, Hinrich Schütze, and Volker Tresp. 2022. Enhanced temporal knowledge embeddings with contextualized language representations. *arXiv preprint arXiv:2203.09590*.

Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020c. Graph hawkes neural network for forecasting on temporal knowledge graphs. In *AKBC*.

Zhen Han, Gengyuan Zhang, Yunpu Ma, and Volker Tresp. 2021b. Time-dependent entity embedding is not all you need: A re-evaluation of temporal knowledge graph completion models under a unified framework. In *EMNLP*.

Robert L. Logan IV, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *EMNLP*.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kalev Leetaru and Philip A. Schrodt. 2013. Gdelt: Global data on events, location, and tone. *ISA Annual Convention*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. 2017. The more you know: Using knowledge graphs for image classification. In *CVPR*.

Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *NAACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *NeurIPS*.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. TimeTraveler: Reinforcement learning for temporal knowledge graph forecasting. In *EMNLP*.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep reasoning in temporal knowledge graphs. In *ICML*.

Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. 2020. Capsules with inverted dot-product attention routing. In *ICLR*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *ICLR*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-vil: Knowledge enhanced vision-language representations through scene graphs. In *AAAI*.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI*.