VTG: Learning historical information with variable time granularity

Hao Xiong

Guangdong University of Technology, Guangzhou, China

2122105102@mail2.gdut.edu.cn

Keywords: Temporal knowledge graphs, Variable time granularity, Hierarchical weighted aggregation, Link prediction, Embedding model

Abstract: Temporal knowledge graph inference is pivotal for under- standing knowledge graph evolution. However, current approaches often underutilize temporal information. Moreover, there exists room for improvement in the critical task of link prediction within knowledge graph inference. This study introduces an innovative inference architecture that employs multi-domain attention and dynamic time strategies, enabling adaptive selection of historical data and temporal spans to enhance the timeliness and accuracy of temporal knowledge graph inference. Empiri- cal results demonstrate substantial performance enhancements in knowl- edge graph state prediction. Furthermore, the model excels in the task of link prediction across multiple temporal knowledge graph datasets. Our approach showcases substantial potential across diverse domains, including social networks, scientific literature, and intelligent recommendation systems, contributing to the advancement of knowledge graph research.

1. Introduction

The concept of knowledge graphs (KGs) was initially developed by Google as a knowledge base to support its semantic search. As the application of knowledge graph technology has deepened, KGs have become the most important form of knowledge representation in the era of big data. As a form of knowledge representation, a knowledge graph is a large-scale semantic network in which each fact is represented by a triplet (s, r, o) consisting of a head entity, relation and tail entity. However, many facts are based on time conditions, hence the concept of temporal knowledge graph (TKGs) has been introduced. Specifically, a timestamp T is added to the original triple (s, r, o) to form a quadruple (t, s, r, o). For example, when expressing the temperature of a day, it can be expressed as (June 15th, 2022, New York, average_temperature, 75 degrees Fahrenheit) and (June 16th, 2022, New York, average_temperature, 71 degrees Fahrenheit). TKGs usually have the problem of incompleteness, and link prediction tasks become one of their effective solutions. The link prediction task in TKGs aims to predict missing temporal facts based on existing facts in the graph. This task often relies on representation learning, which maps entities and relationships to lowdimensional vector spaces to enrich their semantic representation. The core issue of representation learning in TKGs is to represent the constantly updated graph structure information as low-dimensional vectors.

Most existing methods, such as RE-Net [12], DySAT [9] and ALRE-IR [3], tend to retain very limited graph structure information, selecting historical snapshots before the query time point for learning. These methods generally believe that historical facts closer to the query time point contain more information, while the information in the distant historical subgraph can be ignored. However, this way of selecting historical snapshots lose a large amount of historical information and cannot learn more complete node representations.

To address this issue, we propose VTG, an embedding model for TKGC, which abandons the fixed time window approach used in previous works to capture historical snapshots and instead adopts a variable time granularity historical snapshot selection mechanism. This mechanism can enhance the model's ability to learn more comprehensive node representations and improve link

prediction performance by selecting historical snapshots with variable time granularity based on different time points.

In addition, we introduce a hierarchical weighted aggregation method for neighborhood aggregation, which more intuitively and effectively aggregates neighborhood information. Specifically, by assigning a neutral strength score *valr* to each relationship, which indicates the level of adversarial or cooperative behavior in the relationship. Subsequently, it calculates the average of all relationship scores between any two pairs of nodes to determine the overall adversarial or cooperative level, denoted as *val(s,o)*. The hierarchical weighted aggregator uses a mechanism, similar to hierarchical attention, to learn node representations. It divides into relationship layer attention and entity layer attention.

The relationship layer attention differentiates the importance of facts by the semantic similarity (difference in neutral scores) between the query relationship and neighborhood relationships. The entity layer attention differentiates the importance of facts by the difference between the neutral scores of neighborhood relationships and the neutral scores between entities. Finally, a temporal sequence encoder is used to capture the time dependencies of historical subgraphs at different time points.

The main contributions of this paper are as follows:

(1) We propose an embedding model for TKGC, VTG, which adopts a variable time granularity mechanism to help the model learn more comprehensive node representations.

(2) We introduce a hierarchical weighted aggregation method for neighborhood aggregation, which enhances the interpretability of the inference by separately aggregating neighborhood information at the relationship and entity layers while incorporating query information into the aggregation process.

(3) We conducted extensive experiments on three public temporal knowledge graph datasets, demonstrating the effectiveness of VTG in the link prediction task.

2. Related Work

Temporal knowledge graphs is a type of knowledge graph that incorporates temporal information to represent and reason about facts that change over time, allowing for more accurate and dynamic representation of real-world knowledge. As a result, TKGs have begun to attract more attention from researchers. The main research directions are as follows:

2.1 Learning temporal information

Some researchers attempt to extend static knowledge graph embedding techniques to the domain of temporal knowledge graphs. TTransE [18] is an extension of the TransE [23] model, which represents relations as time-dependent transition matrices, using matrix multiplication to capture temporal dependencies between entities. In HyTE [16], timestamps are regarded as hyperplanes capable of projecting entity and relation vector representations. TComplEx [8], as an extension of ComplEx, defines temporal knowledge graphs as fourth-order tensors and introduces regularization techniques to prevent overfitting.

2.2 Dynamic embeddings

Dynamic embedding methods aim to model the dynamic evolution process of entities or relations to learn their dynamic embeddings. Some researchers use timestamps to represent dynamic information. The core idea of ATiSE [14] is to decompose time series into trend features, seasonal features, and random noise, treating each part as an independent embedding space. Entities or relations are modeled as Gaussian distributions on time steps, with KL divergence used to calculate fact scores.DE-SimplE [7], inspired by diachronic word embeddings, add a diachronic entity embedding function to static models, providing entity feature representations at any time point. The TeMP [10] model combines multihop message-passing structural encoders and time encoders to model dynamic graph data. The structural encoder learns the structural dependencies of entities at each time point and feeds the output to the time encoder.

2.3 Learning from graph snapshots

The evolution process of temporal knowledge graphs can be viewed as a collection of static knowledge graph snapshots or subgraphs. Recently, some researchers have attempted to construct autoregressive models to handle a series of subgraph slices obtained from splitting temporal knowledge graphs. DySAT [9] decouples the evolution process of temporal knowledge graphs into structural selfattention layers and temporal self-attention layers. The structural self-attention layer learns local neighborhood information of nodes in each snapshot, while the temporal self-attention layer flexibly weights historical representations to capture graph evolution processes over multiple time steps. RE-Net [12] models the dynamic evolution of facts in an autoregressive manner. It utilizes a multi-relation aggregator to encode global graph structural information and local multi-hop neighborhood information. Furthermore, static attributes of entities are used as constraints to further refine entity representations. In contrast to discrete evolution processes, Han et al.[5] employ continuous-time embeddings to encode temporal and structural information from historical snapshots. They adopt a time-dependent neighbor sampling approach, utilizing multi-relation graph convolution networks to capture graph structural information and neural ordinary differential equations (ODE)[15] to model the dynamic evolution process.

3. Method

In this section, we introduce VTG and begin with an overview of the VTG model's architecture, followed by a detailed discussion of its three main components: Variable time granularity sampling, Hierarchical weighted aggregator, and Time series encoder. Then, we delve into the implementation principles of each module and the connections between them. Lastly, we outline the model's inference process.

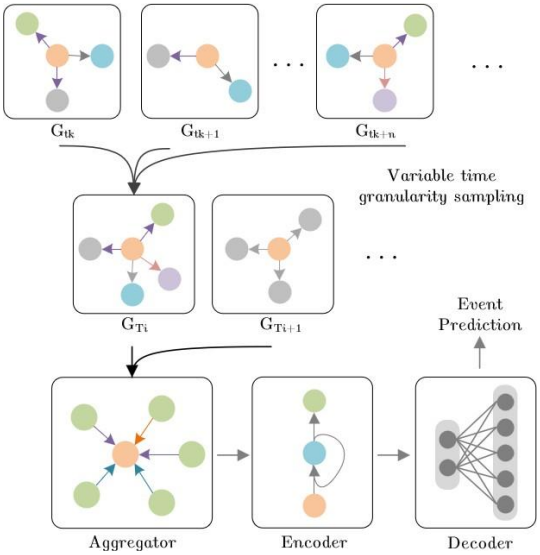

Fig. 1. Illustration of VTG

3.1 Overall architecture of VTG

Fig.1 illustrates the overall architecture of VTG, which consists of three submodules: variable time granularity sampling, hierarchical weighted aggregator, and time series encoder. In the temporal knowledge graph, all temporal facts are sorted in ascending order according to timestamps and divided into a series of subgraph snapshots, i.e., $G = \{G1, G2, \cdots, GT\}$ represents the static subgraph of the temporal knowledge graph G at timestamp t, containing all temporal facts at that moment. Given a query(s, r, o, t_q), the model first uses the variable time granularity sampler to sample historical snapshots before the timestamp t_q to capture as much historical fact information as possible. Simply put, this sampler merges multiple historical snapshots at different timestamps into a single snapshot in a reasonable manner. For example, in Fig.1, multiple historical subgraphs$\{G_{t_k}, G_{t_{k+1}}, \cdots, G_{t_{k+n}}\}$ are merged into one graph structure G_{Ti} . Next, the hierarchical weighted

aggregator is used to aggregate neighborhood information for node s to learn a node representation based on its neighborhood. After learning the node embeddings in different historical snapshots, the temporal sequence encoder is used to capture temporal dependencies between entities. Finally, the output of the temporal sequence encoder is passed to the decoder to provide a probability prediction distribution for the query.

3.2 Variable time granularity sampling

In existing research based on discrete time methods, they usually use limited graph structure information, such as RE-Net [12] and DySAT [9]. These methods typically use a fixed-length historical window for sampling historical snapshots. This approach has two drawbacks:

(1) The learned entity representations are not comprehensive enough. They use a fixed-length window for sampling, and the model can only capture limited graph structure information. Historical information farther from the query timestamp is completely discarded.

(2) It cannot distinguish the importance of historical snapshots at different timestamps.

To address these issues, we designed a variable time granularity sampling method to help the model learn comprehensive node representations. Intuitively, historical information generally has the following characteristics: the closer the historical information is to the prediction timestamp, the richer the valid information it contains, while the farther the historical information is, the less impact it has on the prediction results. Variable time granularity sampling is performed by defining a time span sequence to sample historical snapshots. The definition of the time span sequence is as follows:

$$T_{span} = (a^n, \cdots, a^1,)(m, \cdots, m) \tag{1}$$

T_{span} is a sampling sequence composed of time spans, where each time span represents the fusion of multiple historical snapshots into a single graph structure. Note that T_{span} consists of two parts: the first part uses the powers of a as a time span, with smaller powers closer to the query time point, up to the power of 1. The first part is a variable time granularity sampling sequence. The second part uses a fixed time span m, with a sequence length of *len*. The second part is a fixed time granularity sampling sequence.

In this section, the sampling method combines fixed time granularity sampling and variable time granularity sampling. Fixed time granularity sampling divides the historical snapshots near the query time point into *len* groups with a time span of m, which is used for learning the more critical historical information for the current query. Variable time granularity sampling divides the historical snapshots farther from the query time point into time spans with powers of a, where smaller powers are closer to the query time point, used for learning historical information with relatively less influence on the current query. Due to the use of the powers of a, it is easy to incorporate all historical information into the learning process. The farther the historical information is from the query time point, the less impact it has on the query, and it will be aggregated into a subgraph with a larger time span accordingly. At the same time, we remove duplicate facts in the aggregated subgraphs to reduce the model's computational load.

For example, when $a = 2$, $m = 1$, $len = 3$, the time span sequence for sampling a history snapshot of length 3 is (1, 1, 1). When sampling a history snapshot of length 50, the time span sequence is (17, 16, 8, 4, 2, 1, 1, 1). The time span sequence calculation formula used in this paper is only a relatively reasonable design method. In the future, more design methods are waiting to be explored.

3.3 Hierarchical weighted aggregator

Temporal knowledge graph datasets such as YOGO, ICEWS, etc., provide an observer-neutral intensity score for each relationship, indicating the level of hostility or cooperation between entities. The score ranges from [-10, 10]. To facilitate subsequent calculations, we normalize the score to the range [0, 1]. The neutral score of the relationship is defined as follows:

$$\text{valR} = \{\text{valr1}, \text{valr2}, \cdots, \text{valrN}\} \tag{2}$$

Where val_R represents the set of neutral scores for all relationships, and $valr_i$ represents the neutral score of relationship r_i, with a value range of [0,1]. At the same time, we assign a neutral score to each entity pair, representing the degree of hostility or cooperation between entities. The neutral score of the entity pair is calculated by averaging the neutral scores of all relationships between the two entities. The neutral score of the entity pair is defined as follows:

$$val(S,O) = \{val(si,oj)|si \in S, oj \in O\} \tag{3}$$

$$val(s_i, o_j) = \frac{1}{k} \sum_{r \in R(s_i, o_j)} val_r \tag{4}$$

Where $val_{(S,O)}$ represents the set of neutral scores for all entity pairs, $val_{(si,oj)}$ represents the neutral score between entity s_i and entity o_j, with a value range of [0, 1]. $R(s_i, o_j)$ represents the set of relationships between s_i and o_j, and k represents the size of $R(s_i, o_j)$, i.e., the number of relationships between entities.

Given a query (s, r_q, o, t_q), after capturing historical subgraphs using a variable time granularity sampler, the model uses a hierarchical aggregator to aggregate the new historical subgraphs, which consist of entity-level attention and relationship-level attention.

Entity-level attention considers that the weights of different tail entities under the same relationship r_i on the central node are different, distinguished by the difference between the neutral score val_{ri} of the relationship r_i and the neutral score $val_{(s,o)}$ between entities. The smaller the difference, the greater the weight. Specifically, when the values of val_{ri} and $val_{(s,o)}$ are closer, it indicates that the current level of hostility or cooperation between the entities is more consistent with the level of hostility or cooperation represented by the relationship, thus having a stronger representation and occupying a higher weight. The definition of entity-level attention is as follows:

$$H_T(s, r_i) = \frac{1}{n} \sum_{o \in O^T_{(s, r_i)}} (1 - |valr_i - val_{(s,o)}|) \cdot o \tag{5}$$

Here, $H_T(s, r_i) \in R^d$ represents the embedding representation of the head entity s in relation r_i in graph G_T . $O^T_{(s, ri)}$ represents the set of tail entities in graph G_T that have a relationship r_i with head entity s, and n is the size of the set.$|val_{ri} - val_{(s,o)}|$ represents the difference between the neutral score of relationship r_i and the neutral score between entities. $O \in R^d$ represents the embedding representation of tail entity o.

Relationship-level attention believes that different relationships should have different impacts on the central node. Based on this, the model introduces the query relationship to guide the aggregation of relationship-level attention. It considers that relationships with a neutral score val_{rq} close to the query relationship r_q have a greater weight on the representation of the central node. Specifically, when the values of val_{rq} and val_{ri} are closer, it indicates that the query relationship is more semantically similar to the current neighborhood relationship, and therefore, it has more say in predicting the current query. The definition of relationship-level attention is as follows:

$$H_T(s) = \frac{1}{m} \sum_{r_i \in R_s^T} (1 - |valr_q - valr_i|) \cdot \left[r_i \,\|\, H_T(s, r_i) \right] \tag{6}$$

Here, $H_T(s) \in R^{2d}$ represents the static representation of the head entity s in graph G_T , which includes the neighborhood information of the node in the graph. Rs^T represents the set of relationships related to the head entity s in graph G_T, where m is the size of the set. $(1 - |val_{rq} - val_{ri}|)$ is used to calculate the semantic similarity between the query relationship r_q and the relationship r_i. Here, $r_i \in Rd$ represents the embedding representation of the relationship r_i, and || denotes the concatenation operation.

3.4 Time series encoder

In VTG, we use GRU (Gated Recurrent Unit) as the time series encoder to capture the time

dependencies between different historical snapshots. After learning the static representation of nodes in different historical subgraphs through the hierarchical weighted aggregator, the sequence composed of the static representations of nodes at different time points is fed into the GRU sequence encoder to obtain the dynamic representation of nodes, which contains all the historical information related to the nodes. The specific implementation of GRU is as follows:

$$hT(s) = GRU(HT(s), hT-1(s)) \tag{7}$$

Where $h_T(s)$ represents the dynamic representation of node s at time T, and $H_T(s)$ represents the static representation of node s at time T.

3.5 Reasoning process

For a given query (s, r_q, o, t_q), we first obtain the dynamic representation $h_T(s)$ related to node s. We then concatenate this representation with the original node representation s and the relation representation r_q. The resulting concatenated vector is passed to a linear layer, and the output is normalized using the softmax function to obtain the final probability prediction distribution. The definition is as follows:

$$P(o|s, rq, tq) = softmax(W[s||rq||hT(s)]) \tag{8}$$

Where $\mathbf{W} \in R^{3d \times N}$ is a trainable parameter.

4. Experiments

In this section, we evaluate the model on three publicly available temporal knowledge graph datasets through a link prediction task. Firstly, we provide a detailed introduction to the experimental settings, including the datasets, evaluation metrics, parameter settings, and baseline methods. Next, we compare and analyze the experimental results and conduct ablation experiments to assess the importance of the variable time granularity sampling method.

4.1 Experimental settings

To evaluate the model, we selected three datasets commonly used in temporal knowledge graph link prediction tasks: YAGO [20], WIKI [21], and ICEWS05-15 [19].Table 1 provides a detailed description of the statistical data for these datasets.

We sort the temporal facts in each dataset by their timestamps and split them into training, validation, and test sets at a ratio of 80%/10%/10%. We adopt the widely used filtered version of the evaluation metric MRR (Mean Reciprocal Ranking) and Hits@1/3/10 to measure the performance of VTG.

Table 1. Statistics of the benchmark datasets

Dataset	Entities	Relations	Training	Validation	Test	Timestamps	Granularity
YAGO	10,623	10	161,540	19,523	20,026	189	1 year
WIKI	12,554	24	539,286	67,538	63,110	232	1 year
ICEWS05-15	10,488	251	368,868	46,302	46,159	4,017	24 hours

Regarding the parameter settings for the variable time granularity sampling method in the model, the sampling sequences of variable time granularity use powers of 2 as the time span. For the fixed time granularity sampling sequences, we set the fixed time span to 1 and the sequence length to 5. In other words, $a = 2$, $m = 1$, $len = 5$.

4.2 Baselines

We compared VTG with several static knowledge graph embedding models, including TransE[23], DistMult[22], DKGE-LFS[4] and RotatE[13], as well as several temporal knowledge graph embedding models, including HyTE[16], TTransE[18], TA-DistMult[17], RE-Net[12], CyGNet[11], StAR(Self-Adp)[6], TeCre[2] and FIT-CARL[1].

Table 2 Comparison of performance between VTG and other baseline models on three datasets

Method	YAGO			WIKI			ICEWS05-15		
	MRR	Hits@3	Hits@10	MRR	Hits@3	Hits@10	MRR	Hits@3	Hits@10
TransE	48.97	62.45	66.05	46.48	49.71	51.71	29.40	-	66.30
DKGE-LFS	46.00	47.90	53.50	43.12	39.73	47.52	35.33	-	62.95
DistMult	59.47	60.91	65.26	46.12	49.81	51.38	45.60	-	69.10
RotatE	65.09	65.67	66.16	50.67	50.71	50.88	30.40	35.50	59.50
HyTE	23.16	45.74	51.94	43.02	45.12	49.49	31.60	44.50	68.10
TTransE	32.57	43.39	53.37	31.74	36.25	43.45	27.10	-	61.60
TeCre	39.80	65.71	38.30	43.50	-	55.60	49.60	52.40	48.30
TA-DistMult	61.72	65.32	67.19	48.09	49.51	51.70	47.40	-	**72.80**
RE-NET	65.16	65.63	68.08	51.97	52.07	53.91	-	-	-
CyGNet	63.47	65.71	68.95	45.50	50.79	52.80	57.22	61.77	68.58
StAR(Self-Adp)	49.60	66.80	-	**56.30**	53.80	-	52.50	58.30	-
FITCARL	60.13	55.62	58.41	45.92	48.11	52.71	51.30	61.80	70.00
VTG	**67.13**	**67.55**	**69.65**	55.60	**55.98**	**58.16**	**58.81**	**61.89**	70.28

4.3 Main Results

Table 2 reports the performance comparison of the VTG model and baseline methods on the link prediction task for the three temporal knowledge graph datasets. The best results are highlighted in bold, and the second-best results are underlined.

We observe that the performance of temporal knowledge graph embedding methods is significantly better than that of static knowledge graph embedding methods. This is because static knowledge graph embedding methods cannot capture the temporal information in the facts and cannot model the dynamic evolution of entities and relations. Table 2 shows that our model outperforms the baseline methods on almost all metrics. Additionally, some previous works did not report the Hits@1 metric on the mentioned datasets, while the VTG model achieves 65.54%, 54.22% and 52.66% on YAGO, WIKI, and ICEWS05-15 datasets.

4.4 Ablation Study

In this section, we study the effect of variations in VTG on the YOGO and ICEWS05-15 dataset. We present the results in Table 3.

Table 3 Ablation study on the YOGO and ICEWS05-15 datasets.

Method	YAGO				ICEWS05-15			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
VTG w/o agg.	64.69	63.07	64.04	65.31	54.69	48.07	55.04	66.31
VTG w/o var.	62.06	61.47	61.98	63.62	51.32	45.83	54.77	65.02
VTG	**67.13**	**65.54**	**67.55**	**69.65**	**58.81**	**52.66**	**61.89**	**70.28**

The VTG w/o agg. model does not use hierarchical weighted aggregation, and directly uses the historical subgraph information as the input of the encoder; The VTG w/o var. Model directly uses a fixed time granularity to extract subgraph information, and then uses hierarchical weighted aggregation to extract subgraph information. In Table 3, it can be clearly seen that both the hierarchical weighted aggregation method and the variable time granularity method play a key role in the model effect.

5. Conclusion

In this paper, VTG adopt a variable time granularity sampling method to select historical snapshots, incorporating complete historical information into the learning process to obtain more comprehensive node representations. We also employ a hierarchical weighted aggregation method to capture node neighborhood information and introduce query information into the aggregation

process. Experimental results show VTG has good performance in temporal knowledge graph link prediction tasks. In future work, we plan to continue exploring how to effectively utilize the temporal information in snapshots based on variable time granularity.

References

[1] Zifeng Ding et al. "Improving Few-Shot Inductive Learning on Temporal Knowledge Graphs using Confidence-Augmented Reinforcement Learning". In: arXiv preprint arXivv2304.00613 (2023).

[2] Jiangtao Ma et al. "TeCre: A Novel Temporal Conflict Resolution Method Based on Temporal Knowledge Graph Embedding". In: Information 14.3 (2023), p. 155.

[3] Xin Mei et al. "An Adaptive Logical Rule Embedding Model for Inductive Reasoning over Temporal Knowledge Graphs". In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022, pp. 7304 - 7316.

[4] Tianxing Wu et al. "Efficiently embedding dynamic knowledge graphs". In: Knowledge-Based Systems 250 (2022), p. 109124.

[5] Zhen Han et al. "Learning neural ordinary equations for forecasting future links on temporal knowledge graphs". In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. 2021, pp. 8352- 8364.

[6] Bo Wang et al. "Structure-augmented text representation learning for efficient knowledge graph completion". In: Proceedings of the Web Conference 2021. 2021, pp. 1737 - 1748.

[7] Rishab Goel et al. "Diachronic embedding for temporal knowledge graph completion". In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. 04. 2020, pp. 3988 - 3995.

[8] Timothee Lacroix, Guillaume Obozinski, and Nicolas Usunier. "Tensor decompositions for temporal knowledge base completion". In: arXiv preprint arXivv2004.04926 (2020).

[9] Aravind Sankar et al. "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks". In: Proceedings of the 13th international conference on web search and data mining. 2020, pp. 519 - 527.

[10] Jiapeng Wu et al. "Temp: Temporal message passing for temporal knowledge graph completion". In: arXiv preprint arXivv2010.03526 (2020).

[11] Cunchao Zhu et al. "Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks". In: arXiv preprint arXivv2012.08492 (2020).

[12] Woojeong Jin et al. "Recurrent event network: Autoregressive structure inference over temporal knowledge graphs". In: arXiv preprint arXivv1904.05530 (2019).

[13] Zhiqing Sun et al. "Rotate: Knowledge graph embedding by relational rotation in complex space". In: arXiv preprint arXivv1902.10197 (2019).

[14] Chengjin Xu et al. "Temporal knowledge graph embedding model based on additive time series decomposition". In: arXiv preprint arXivv1911.07893 (2019).

[15] Ricky TQ Chen et al. "Neural ordinary differential equations". In: Advances in neural information processing systems 31 (2018).

[16] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P Talukdar. "HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding". In: EMNLP. 2018, pp. 2001 - 2011.

[17] Alberto Garcia-Duran, Sebastijan Dumancic, and Mathias Niepert. "Learning sequence encoders for temporal knowledge graph completion". In: arXiv preprint arXivv1809.03202 (2018).

[18] Julien Leblay and Melisachew Wudage Chekol. "Deriving validity time in knowledge graph".

In: Companion proceedings of the the web conference 2018. 2018, pp. 1771 - 1776.

[19] Jennifer Lautenschlager, Steve Shellman, and Michael Ward. "Icews event aggregations". In: Harvard Dataverse 3 (2015).

[20] Farzane Mahdisoltani, Joanna Biega, and Fabian M Suchanek. A knowledge base from multilingual Wikipedias-yago3. Tech. rep. Technical report, Telecom ParisTech. http://suchanek. name/work/publications …, 2014.

[21] Denny Vrandecic and Markus Krotzsch. "Wikidata: a free collaborative knowledgebase". In: Communications of the ACM 57.10 (2014), pp. 78 - 85.

[22] Bishan Yang et al. "Embedding entities and relations for learning and inference in knowledge bases". In: arXiv preprint arXivv1412.6575 (2014).

[23] Antoine Bordes et al. "Translating embeddings for modeling multi-relational data". In: Advances in neural information processing systems 26 (2013).