Towards A Question Answering System over Temporal Knowledge Graph Embeddings

Kristian Otte[†], Kristian Simoni Vestermark[†], Huan Li and Daniele Dell'Aglio

Department of Computer Science, Aalborg University, Aalborg, Denmark

Abstract

Question Answering (QA) over knowledge graphs is a vital topic within information retrieval. Questions with temporal intent are a special case of questions for QA systems that have received only limited attention so far. In this paper, we study using temporal knowledge graph embeddings (TKGEs) for temporal QA. Firstly, we propose a microservice-based architecture for building temporal QA systems on pre-trained TKGE models. Secondly, we present a Bayesian model average (BMA) ensemble method, where results of several link prediction tasks on separated TKGE models are combined to find better answers. Within the system built using the microservice-based architecture, the experiments on two benchmark datasets show that BMA provides better results than the individual models.

1. Introduction

Knowledge graphs (KGs), such as Wikidata [1], DBpedia [2], and YAGO [3] have attracted increasing attention of researchers and practitioners. Large and detailed KG set the foundations for question answering (QA), a core task in applications such as home assistants, chat-bots, and recommender systems [4]. Many question answering over knowledge graph (QA-KG) studies [5, 6] have treated the KG as a database, where the natural language questions are translated into queries, e.g. in SPARQL, to and are evaluated over the KG. As an alternative approach, recent research [7, 8, 9, 10] has proposed knowledge graph embedding (KGE) for QA systems. KGE models allow systems to find answers that are not explicitly stated in the KG.

In this study, we focus on the task of QA on temporal knowledge graphs (TKGs). *Temporal questions* are a common type of human questions that involve a time context. Such a context can be explicit, e.g., "Who won the Oscar for best-supporting actor *in 2022*?", or implicit, e.g., "Who was pope *during the fifth crusade*?". Answers to these questions may also contain temporal information, such as questions starting with *when*.

The state of the art includes dedicated systems that focus on answering temporal questions [11, 6]. Some of them, like EXAQT [11], adopt TKG or time-aware embeddings that are learned in the context of the QA task. We observe that such systems are hardly extendable, for example it is challenging to replace the embedding model with pre-existing models learned using temporal

Workshop at ISWC 2022 on Deep Learning for Knowledge Graphs

[†]These authors contributed equally.

kotte17@student.aau.dk (K. Otte); kveste16@student.aau.dk (K. S. Vestermark); lihuan@cs.aau.dk (H. Li); dade@cs.aau.dk (D. Dell'Aglio)
0000-0003-0084-1662 (H. Li); 0000-0003-4904-2511 (D. Dell'Aglio)

knowledge graph embedding (TKGE) methods [12, 13, 14, 15, 16, 17, 18]. To overcome such a limitation, we present Verðandi a QA system built on top of pre-trained TKGEs. In Verðandi, each pre-trained model is exposed through a microservice and invoked by other services to solve temporal link prediction and temporal question answering tasks.

Verðandi can support the research on TKGE by providing an environment where to compare and contrast existing methods, as well as design and test new techniques. Based on Verðandi, we start investigating *whether different TKGE models capture different information from the original TKGs*. To do that, we design an ensemble-based module that aggregates the outcomes from each model. In our experiments using two temporal QA benchmark datasets, we find that the ensemble module can produce better results than the individual TKGE models, suggesting that our hypothesis may hold.

To summarise, the main contributions of this paper are as follows:

- An open-source[1] microservice-based framework that enables an extensible QA-KG.
- An ensemble method that combines the link prediction results from multiple TKGEs, where the combination yields better results than using an individual TKGE model.

Sections 2 and 3 present the background and the related work, respectively. Section 4 introduces the architecture of Verðandi, and Section 4.3 describes how Verðandi uses ensemble learning to achieve better results than the individual models. Section 5 presents our experimental study on ensemble learning for QA-KG. Section 6 discusses the results and the limitations, and identifies future actions of our research. Section 7 concludes the paper.

2. Background and Notation

A **knowledge graph** is a directed graph with labeled vertices that represent entities, and edges that denote the relations between entities. Examples of KGs include Wikidata [1], Freebase [19], DBpedia [2], YAGO [3], and ICEWS [20]. The entities and relations form the entity-relation-entity triples called **facts**, written as (h, r, t), where h is the head entity, r is the relation, and t is the tail entity. An example fact is (`Obama`, `isPresidentOf`, `USA`). **Temporal knowledge graphs** extend KGs by adding temporal annotations to form entity-relation-entity-time quadruples, called **temporal facts**. We indicate a temporal fact as (h, r, t, τ), where τ is the time. The time can be either an interval [1] or a discrete timestamp [20]. A temporal fact with an interval, e.g., (`Obama`, `presidentOf`, `USA`, [2009,2017]), can be converted into two temporal facts with discrete timestamps indicating the beginning and the ending of the event, e.g., (`Obama`, `becomePresidentOf`, `USA`, 2009) and (`Obama`, `endPresidentOf`, `USA`, 2017). This study focuses on temporal facts with discrete timestamps.

A **temporal question** is a question where a time context is part of the question or the answer. An example of where time is part of the question is (Q1) *"Who became president of the USA in 2009?"*, and an example of where time is the answer is (Q2) *"When did Barack Obama become president of the USA?"*. Depending on the type of question, an **answer** to a temporal question is an entity, relation, or time information. This answer is the result of using a TKGE model with the known information from the temporal question. Examples of answers are `Obama` for Q1 and 2009 for Q2 mentioned above.

[1]https://gitlab.com/tkge, MIT licence

Link prediction is the task of predicting a valid fact in a KG. It is usually defined as the problem of finding a tail given a head and a relation (denoted $(h, r, ?)$), or a head given a relation and a head (denoted $(?, r, t)$). Link prediction is one of the most popular tasks for KGE. Using link prediction, one can solve QA tasks, as link prediction can be used to determine the likelihood of an answer to be true. Thus, when a question is issued, link prediction can be used to generate likely responses.

3. Related Work

KGEs have been extensively investigated in the last decade [4]. Recently, research has begun on integrating temporal dimensions into KGE, mainly extending non-temporal KGE techniques [12, 13, 14, 15, 16, 17, 18]. ChronoR [18] is inspired by rotational KGE methods, such as RotatE [21]. It uses the linear transformations of rotation and scaling, parameterized by time and relation, to obtain the embedding of a tail entity from a head entity. The Diachronic Enitity Embedding methods [15] include DE-SimplE, DE-DistMult, and DE-TransE, as time-aware versions of SimplE [22], DistMult [23], and TransE [24]. All these methods embed time with entities, inspired by diachronic word embeddings [25]. TimePlex [16] is based on ComplEx [26], and embeds entities, relations, and time as vectors in the complex space.

QA is a popular KGE task [8, 9, 10], where embeddings of an entity and relation are extracted from a question and are used in a link prediction task to find the answers. UPSQA [8] finds the top-k candidate entities in the question using a bi-LSTM. Then, given the question and a candidate entity, it finds the relation using another bi-LSTM. BuboQA [9] first solves an entity detection task using a bi-LSTM as UPSQA. Next, it uses a fuzzy matching mechanism to identify a candidate entity from the KG. The relation is then found using a bi-GRU over all the relations that are used within the candidate entity. BERTQA [10] uses BERT to detect entities and classify relations. Entities in the KG are then linked using fuzzy matching and candidate facts are formed by combining each relation found with the entity that has the highest probability.

QA on TKGs has recently emerged with systems such as TEQUILA [6] and EXAQT [11]. TEQUILA is an enabler method for temporal QA, which can run on top of any static QA-KG system. It detects if a question has temporal intent: if yes, it decomposes the question into a non-temporal sub-question and a temporal constraint. While the underlying QA-KG system is handling the non-temporal sub-question, the temporal constraint is solved using constraint reasoning on temporal intervals. EXAQT answers complex temporal questions with multiple entities, relations, and associated temporal conditions using TKG in two steps. First, question-relevant compact sub-graphs are computed within the KG and are enhanced with temporal facts using Group Steiner Trees and BERT models. Second, it creates relational graph convolutional networks enhanced with time-aware entity embeddings and attention of temporal relations. When designing Verðandi, we got inspired by TEQUILA on using TKGs and in the idea of being an enabler system where the data component can be changed. Similarly to EXAQT we rely on TKG for designing the QA mechanism. However, differently from it, we rely on pre-trained state-of-the-art TKGEs models instead of computing ad-hoc embeddings based on the QA task.

4. Verðandi

When we designed Verðandi, we decided to rely on a microservice-based architecture as it brings modularity and helps us in conceptualising the interfaces in a TKGE-agnostic fashion. Furthermore, microservices bring an inherent level of scalability, setting the foundations for future studies on large-scale QA systems.

4.1. The Verðandi Modules

We defined three different types of modules based on the functionalities and the individual responsibilities. These are: (i) the user interface module, (ii) the natural Language module, and (iii) the TKGE module.

The purpose of the **User Interface Module** is to allow a user to interface with the system in a user-friendly manner. The module itself does not have an API, but it invokes the API of the Natural Language Module.

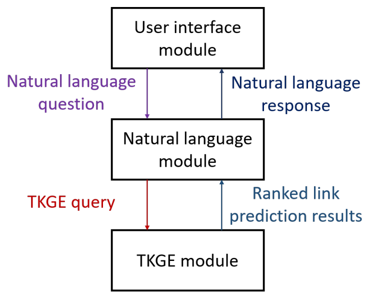
The **Natural Language Module** translates natural language questions into TKG link prediction queries that can be handled by TKGE models. Extending the link prediction queries for KGE, these queries consist in finding missing head, tail or time annotation from a temporal fact, e.g., $(h, r, ?, \tau)$. After the translation, the module submits the link prediction query to a TKGE module, receiving a list of answers as a result. The Natural Language Module then converts such responses into natural language responses and returns them to the invoker module. Thus, the API of the module receives a string-type natural language question as an input and returns a string-type natural language response as the output.

The **TKGE Module** is responsible for evaluating link prediction queries. The API exposes an endpoint that gets as input the link prediction query, where the answer is a list of temporal facts annotated with confidence scores. We intentionally designed a simple API to achieve flexibility and let developers implement any query evaluation mechanism. For example, although our goal is to answer queries using TKGE models, one may implement a TKGE module that answers the query as a triple pattern SPARQL query.

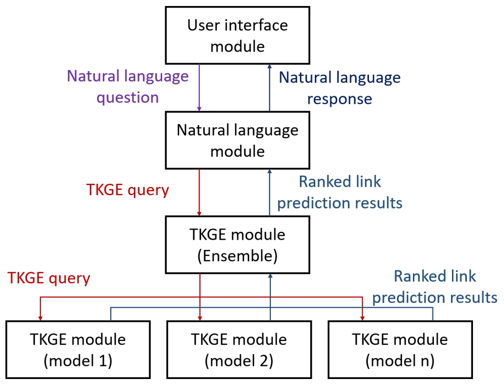
Two possible systems built on these modules are shown in Figure 1. Figure 1a shows a system where there is one TKGE model exposed through a TKGE module. Figure 1b shows a system employing a TKGE module enabling model ensemble: each module in the bottom embeds a different TKGE model and receives the same TKGE query, producing a ranked list of results an an answer. The ensemble module receives the answers and combines them to produce a unified ranked list of results, which is then communicated to the natural language module.

4.2. Verðandi Implementation

Microservice technologies We considered two technologies for developing the API, namely REST (REpresentational State Transfer) and Remote Procedure Call (RPC). RPC is an interprocess communication method that allows a program to call a procedure of another program as if it were part of its own code. This allows for programmers to easily integrate the use of these procedures into their code without thinking about how the communication between these


User interface module
Natural language question
Natural language response
Natural language module
TKGE query
Ranked link prediction results
TKGE module

(a) Configuration with a single TKGE module.



(b) Configuration with the ensemble module orchestrating n TKGE modules.

Figure 1: Two possible system architectures implemented using Verðandi.

works. gRPC[2] is a modern, open source, and efficient RPC framework. It uses a service definition language called Protocol Buffer, which allows developers to define services, endpoints, and messages in a language agnostic manner. Automatic tools are then provided, which enable the generation of language-specific code that implements the client and server part of services defined in Protocol Buffer in various programming languages, such as Go, C++, Java, and Python. Since we do not need the low-level access provided by REST APIs, and because our APIs do not need to be public facing, we choose to use RPC, and specifically gRPC.

Current implementation The current implementation of the **User Interface Module** consists of a command line interface, which allows submitting questions expressed in natural language. Additionally, the request can contain the number of answers to be retrieved. As QA based on KGEs usually find answers ordered according to some confidence score, the correct answer may not be first. As such, it can be relevant to retrieve multiple answers, optionally annotated with the relative confidence score. In future iterations, we plan to implement a more user-friendly interface, such as a web-based one.

In our current implementation of the **Natural Language Model**, we adopted a template-based approach to convert questions into TKGE queries. For example, one of the patterns that the module recognises is: “Who did *relation tail* on *time*?". When the user inputs a query like: “Who became president of USA on November 2019?", the module generates a query (?, `becamePresident`, `USA`, 11.2019). In our future work, we aim to implement more sophisticated state-of-the-art NLP algorithms to ease the conversion of the questions into queries.

Currently, Verðandi includes four **TKGE Modules**. Three are implementations of TKGE techniques from [15], namely DE-SimplE, DE-DistMult, and DE-TransE. We picked such methods as the authors provide high-quality open source code, which is ideal for showcasing how to create a module for existing libraries. In our future plans, we aim to develop new TKGE modules. We are in the process of adding TimePlex [16] as, similarly to [15], provides open source code.

The fourth TKGE module we have implemented so far is the Ensemble Module. This module

[2]Cf. https://grpc.io/ (last accessed: July 2022)

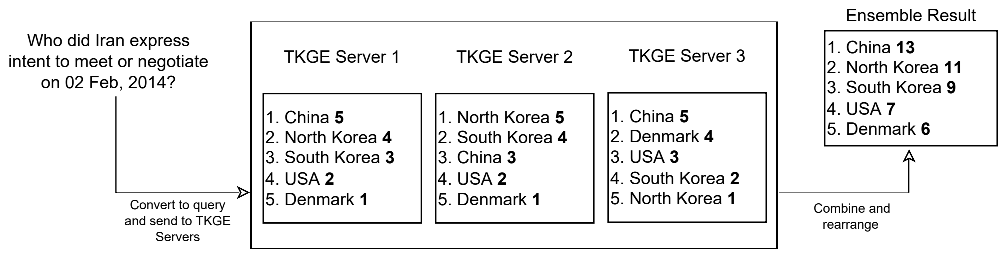
allows the system to orchestrate two or more TKGE modules, combining the results from each of these into a single response. The idea behind the Ensemble Module will be further described in the next section.

4.3. Ensemble of the TKGE Models

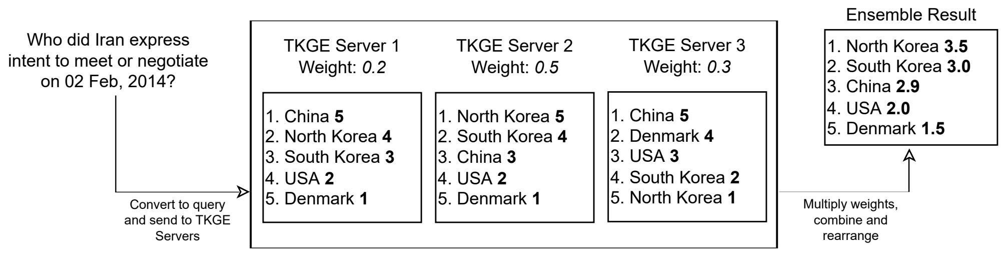
We have used Verðandi to set the basis on studying existing TKGE methods. In particular, we asked ourselves whether different TKGE methods are capturing the same information from the TKG, and if not, whether a combination of them may lead to better QA answers. We investigate these ideas by using ensemble learning, which in other contexts has shown to be able to achieve better predictive performance by combining multiple models [27]. The goal is to retrieve the top-N scoring results from a temporal link prediction query evaluated on different models, and combine the individual answers to create a unified one.

We use Bayesian model averaging (BMA) [28] for the ensemble, which runs the models individually and combines the scores each model predicted. This method uses pre-trained models, which means that multiple existing models that capture different features can be used to improve weaknesses. We employ BMA in two different versions: *unweighted*, which assigns to each model the same weight, and a *weighted*, where we learn a set of weights for each model.

The unweighted BMA takes the top-N results from each of the models and gives them a score equal to their ranking within the top-N. The score s is defined as $s = N - r$, where N is the number of returned results and r is the rank returned by the model. For example, if a model gives a result with the rank of 0 (the model predicts this result is the most likely), among



(a) Process of combining results from TKGE modules, using the unweighted BMA. The scores given to the rankings are shown in bold.



(b) Weighted BMA combining ranked results from three TKGE modules. The results are multiplied by a weight before being added and reordered.

Figure 2: Unweighted and weighted BMA.

10 results ($N = 10, r = 0, s = 10 - 0 = 10$), then the score of that result will be 10. Then if another model gives it a rank of 1, then the combined score for this result, based on these two models, will be 19. Figure 2a shows the process of combining results from multiple TKGE servers using the unweighted method.

The weighted BMA, shown in Figure 2b, works similarly to the unweighted BMA, but it multiplies weights to the scores before comgining them. In this way, different models have different impacts on the final score.

5. Evaluation

Data One of the most used KGs used in literature for testing link prediction in TKGE models is Integrated Crisis Early Warning System (ICEWS). Researchers, in particular, created two datasets named ICEWS14 and ICEWS05-15 [20], which feature facts from the ICEWS dataset from the year 2014 and the years 2005-2015, respectively. These datasets are event-based, meaning that every fact is annotated with a discrete timestamp.

For testing Verðandi, however, we need questions and answers. Therefore, we generate questions from the ICEWS datasets by fitting ICEWS temporal facts into question templates. For example, from the temporal fact (`SouthKorea`, `criticize`, `NorthKorea`, 2014-05-13), we generate questions like "Who did South Korea criticize on 13 May 2014?". For each fact in the datasets, four questions are generated, one for each temporal fact element missing, i.e., head, relation, tail, time. Table 1 shows a summary of the datasets.

In our experiments, however, we use only the questions with missing subject or object because the methods we implemented at the moment do not support queries where these two elements are missing.

Methods As explained in Section 4.2, in our experiments we consider three Diachronic Entity Embeddingvariants: DE-SimplE, DE-DistMult, and DE-TransE. We also use unweighted and weighted BMA as described in Section 4.3. For weighted BMA, there is an extra learning step where the weights are learned. We search the weights using Bayesian optimization [29].

Metrics All systems return a ranked list of the answers from link prediction. The answers consist of the facts and a score for each fact. With the list of answers, we calculate: (i) MRR, i.e., the mean of the reciprocal of the rank of the correct answer, (ii) Hits@1, i.e., the percentage of facts, where the answer with the highest score is the correct answer, and (iii) Hits@10, i.e., the percentage of facts where the correct answer is within the 10 highest scored answers. For all the metrics, the higher the score, the better. When running experiments, only the facts in the

Table 1
Statistics of the datasets ICEWS14 and ICEWS05-15.

Dataset	#Ent.	#Rel.	#Time	Train	Valid	Test	Total	Questions
ICEWS14	7,128	230	365	72.8k	8.9k	8.9k	90.7k	362.9k
ICEWS05-15	10,488	251	4,017	368.9k	46.3k	46.1k	479.3k	1.9M

Table 2
Results of Diachronic Embedding models and ensemble on a link prediction task using the TKGE server.

Model	ICEWS14			ICEWS05-15		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
DE-SimplE	0.505	38.2	73.2	0.496	36.7	**74.4**
DE-DistMult	0.484	36.8	70.7	0.471	34.8	71.4
DE-TransE	0.312	10.1	68.8	0.304	9.5	68.3
Ensemble (unweighted)	0.515	39.3	74.5	0.493	36.6	73.8
Ensemble (weighted)	**0.518**	**39.6**	**74.7**	**0.499**	**37.1**	**74.4**

test set are loaded. This means our results are *unfiltered* and thus may appear worse than the filtered results reported e.g. in [15].

Baselines vs Ensemble We compare the baselines to the ensemble method using an unweighted scorer (see Section 4.3), and report the results in Table 2. The results show that having all the models together perform better than the best individual model on the ICEWS14 dataset, even though DE-DistMult and DE-TransE are underperforming compared to DE-SimplE. The results are slightly worse on the ICEWS05-15 dataset when using an unweighted ensemble compared to the best individual model. It should also be noted that all Diachronic Entity Embedding models perform better on ICEWS14 in both Hits@1 and MRR compared to ICEWS05-15.

Weighted Ensemble As the DE-TransE model is significantly worse at Hits@1 and MRR compared to DE-SimplE and DE-DistMult, we hypothesize that DE-TransE introduces some noise to the ensemble. To test this hypothesis, we do a weighted ensemble to see if eliminating some potential noise yields better results. We search the optimal weights for the ensemble using Bayesian optimization using Gaussian Process [29]. To get a better approximation, we first run some evaluations, where one of the weights is changed at a time to see what impact the individual models have. This allows for narrower bounds during the Bayesian optimization and should yield a better result. The Bayesian optimization ran 25 iterations, and the found weights in the case of ICEWS14 were 0.48 for DE-SimplE, 0.4 for DE-DistMult, and 0.12 for DE-TransE. In the case of ICEWS05-15, the weights were 0.66 or DE-SimplE, 0.33 for DE-DistMult, and 0.01 for DE-TransE. Results for the weighted ensemble is shown as Ensemble (weighted) in Table 2. The results disclose that on the ICEWS14 dataset, using a weighted ensemble further improves the accuracy and has a significant improvement over DE-SimplE, and a slight improvement on the ICEWS05-15 dataset compared to DE-SimplE. The results also show that even though DE-TransE mostly introduces some noise to the ensemble, using it with a lower weight is still better than not using the model. We hypothesize that this is because the models are able to capture different aspects of the TKG, so even if DE-TransE performs bad on its own, it might be able to capture aspects that the other models do not, and thus a combination of them, provides even better results.

6. Discussion

We will now discuss the results both from an architectural viewpoint and an experimental one, as well as provide opportunities for future work.

Framework We chose to implement Verðandi using a microservice-based architecture, as we wanted to have a loosely coupled and modular framework, where components can be substituted for other similar components. This architecture allowed us to easily extend the system to use an ensemble module, encompassing multiple TKGE modules, placing this between the Natural Language module, and the existing TKGE module(s). Furthermore, microservices are inherently scalable when data is independent, as is the case with the individual questions passed through the system. This means that a load balancer can be placed between the caller and the called module, which can then send individual QA requests to the microservice with the least load.

Optimization of Ensembles The results in Table 2 show that the models using BMA gives better results than individual models being used. On ICEWS14, even an unweighted ensemble obtained better results than DE-SimplE, which has the best accuracy of the three models. The weights found using Bayesian optimization further increased the accuracy of the ensemble. As the weights were approximated using Bayesian optimization, the weights are most likely not the optimal weights. Running an exhaustive grid search would further improve the result at the cost of a more expensive parameter search. It is possible that different TKGE models may perform better than others in answering specific questions. We plan, therefore, to investigate the relation between the TKGE models and the question types, with the goal of dynamically varying the weights in the ensemble based on the input question. Furthermore, it is also very likely that using several different TKGE models, such as TimePlex and ChronoR, together with the Diachronic Entity Embedding models, would provide even better results than only using the Diachronic models, as they would likely capture temporal information from different angles.

Hybrid QA Process When using a KGE model for a QA-KG system, it is possible to answer questions by performing link prediction on known and unknown relations. One could imagine a QA-KG system that combines a SPARQL query engine with a link prediction task to achieve better results, following the intuition that known facts should be weighted more. We leave building such a system for future investigations.

7. Conclusion

We proposed Verðandi, a microservice-based framework for QA-KG. We built the framework to be modular, as we were able to use it with many different TKGE models; extensible, as we built an ensemble module on top of it. In future works, we plan to study the scalability of the Verðandi, to test to which extent the microservice architecture can manage high question workloads.

We also proposed to use an ensemble method for combining multiple TKGE models for better results. We chose to use BMA, as it uses pre-trained models and allows us to use different TKGE models. Our experiments suggest that using ensemble methods can provide better results than considering individual models. We consider this result as a first promising step towards investigating whether different models might capture different aspects of the KG. We will continue to investigate this direction, also by exploiting alignment techniques to align the entitites of the TKGs [30]. Finally, we plan to study if we can improve the performance of Verðandi by assigning different weights to different groups of questions.

References

[1] J. Leblay, M. W. Chekol, Deriving Validity Time in Knowledge Graph, in: WWW (Companion Volume), 2018, pp. 1771–1776.

[2] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. v. Kleef, S. Auer, C. Bizer, DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia, Semantic Web 6 (2015) 167–195.

[3] F. M. Suchanek, G. Kasneci, G. Weikum, YAGO: A Core of Semantic Knowledge, in: WWW, 2007, pp. 697–706.

[4] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge Graph Embedding: A Survey of Approaches and Applications, IEEE Trans. Knowl. Data Eng. 29 (2017) 2724–2743.

[5] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, P. Cimiano, Template-based Question Answering over RDF Data, in: WWW, 2012, pp. 639–648.

[6] Z. Jia, A. Abujabal, R. S. Roy, J. Strötgen, G. Weikum, TEQUILA: Temporal Question Answering over Knowledge Bases, in: CIKM, 2018, pp. 1807–1810.

[7] X. Huang, J. Zhang, D. Li, P. Li, Knowledge Graph Embedding Based Question Answering, in: WSDM, 2019, pp. 105–113.

[8] M. Petrochuk, L. Zettlemoyer, SimpleQuestions Nearly Solved: A New Upperbound and Baseline Approach, in: EMNLP, 2018, pp. 554–558.

[9] S. Mohammed, P. Shi, J. Lin, Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks, in: NAACL-HLT (2), 2018, pp. 291–296.

[10] D. Lukovnikov, A. Fischer, J. Lehmann, Pretrained Transformers for Simple Question Answering over Knowledge Graphs, in: ISWC, volume 11778, 2019, pp. 470–486.

[11] Z. Jia, S. Pramanik, R. S. Roy, G. Weikum, Complex Temporal Question Answering on Knowledge Graphs, in: CIKM, 2021, pp. 792–802.

[12] T. Jiang, T. Liu, T. Ge, L. Sha, S. Li, B. Chang, Z. Sui, Encoding Temporal Information for Time-Aware Link Prediction, in: EMNLP, 2016, pp. 2350–2354.

[13] A. García-Durán, S. Dumancic, M. Niepert, Learning Sequence Encoders for Temporal Knowledge Graph Completion, in: EMNLP, 2018, pp. 4816–4821.

[14] S. S. Dasgupta, S. N. Ray, P. P. Talukdar, HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding, in: EMNLP, 2018, pp. 2001–2011.

[15] R. Goel, S. M. Kazemi, M. Brubaker, P. Poupart, Diachronic Embedding for Temporal Knowledge Graph Completion, in: AAAI, volume 34, 2020, pp. 3988–3995.

[16] P. Jain, S. Rathi, Mausam, S. Chakrabarti, Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols, in: EMNLP, 2020, pp. 3733–3747.

[17] T. Lacroix, G. Obozinski, N. Usunier, Tensor Decompositions for Temporal Knowledge Base Completion, in: ICLR, 2020.

[18] A. Sadeghian, M. Armandpour, A. Colas, D. Z. Wang, ChronoR: Rotation Based Temporal Knowledge Graph Embedding, in: AAAI, volume 35, 2021, pp. 6471–6479.

[19] Google, Freebase Data Dumps, 2018. URL: https://developers.google.com/freebase, (Last Accessed: July 2022).

[20] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, M. Ward, ICEWS Coded Event Data, 2015. doi:10.7910/DVN/28075, (Last Accessed: July 2022).

[21] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space, in: ICLR (Poster), 2019.

[22] S. M. Kazemi, D. Poole, SimplE Embedding for Link Prediction in Knowledge Graphs, in: NeurIPS, 2018, pp. 4289–4300.

[23] B. Yang, W.-t. Yih, X. He, J. Gao, L. Deng, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, in: ICLR (Poster), 2015.

[24] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: NeurIPS, 2013, pp. 2787–2795.

[25] A. Kutuzov, L. Øvrelid, T. Szymanski, E. Velldal, Diachronic word embeddings and semantic shifts: a survey, in: COLING, Association for Computational Linguistics, 2018, pp. 1384–1397.

[26] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex Embeddings for Simple Link Prediction, in: ICML, 2016, pp. 2071–2080.

[27] D. W. Opitz, R. Maclin, Popular Ensemble Methods: An Empirical Study, J. Artif. Intell. Res. 11 (1999) 169–198.

[28] J. A. Hoeting, D. Madigan, A. E. Raftery, C. T. Volinsky, Bayesian Model Averaging: A Tutorial, Statistical Science 14 (1999) 382 – 417.

[29] J. Mockus, Bayesian Approach to Global Optimization, Springer, 1989.

[30] M. Baumgartner, D. Dell'Aglio, H. Paulheim, A. Bernstein, Towards the Web of Embeddings: Integrating multiple knowledge graph embedding spaces with FedCoder, J. Web Semant. 75 (2023) 100741.