Summary

In this project we design and implement a microservice-based architecture for building temporal question answering systems on knowledge graph embedding models. We use a microservice-based architecture, as it provides us with the inherent scalability that comes with microservices, where load-balancing, combined with pod-management, means that a developer will not have to spend a lot of effort to upscale a system to encompass more simultaneous users. Furthermore, this type of architecture also means that our system is highly modular, as individual components can be extended, switched, added and removed, without changing other components.

The architecture consists of the following modules: User Interface, Natural Language, Ensemble and Temporal Knowledge Graph Embedding. The User Interface module, provides users with an interface into the system, through command line interface or a graphical user interface. The Natural Language module converts question of a natural language form, into queries that can be solved using link prediction on temporal knowledge graph embeddings. The Ensemble module combines results from several temporal knowledge graph embeddings into one. The Temporal Knowledge Graph Embedding module, computes link prediction on incoming queries.

We implement the communication between the components of the architecture, using remote procedure call, a type of inter-process communication that allows code of one program to call code in another program, as it if were part of it's own code. Specifically, we use the RPC framework called gRPC, which allows API definition, to be written in a separate language called Protocol Buffer. This means that developers can write API definitions in a language agnostic manner, and use this definition to generate server and client code, for a wide variety of programming languages.

We also present a variation of ensemble learning, and explore how it can be used with temporal knowledge graph embeddings, to improve results of question answering over knowledge graphs. We show how the ensemble learning method known as Bayesian model averaging, when applied to question answering on knowledge graphs, using link prediction, provides better results, than what the knowledge graph embedding models can provide individually. We show this by performing an experimental study, where we test the impact of using both weighted and unweighted Bayesian model averaging on question answering. The experimental setting for this study, is constructed in the framework that we present earlier in the project.

We also present using bootstrap aggregating, an ensemble learning technique where multiple models trained on a subset of the original dataset are combined. Initial testing suggests that a large number of bootstrap models are required for this to work, and we therefore do not pursue this any further.

Finally, we present a discussion of the results of our experimental study, as well as present opportunities for future work, followed by the conclusion that ensemble learning can be easily applied to existing temporal knowledge graph embedding modules, to provide better results than the models individually.

MATQA: Microservice-based Architecture for Temporal Question Answering on Knowledge Graph Embeddings

Kristian Otte and **Kristian Simoni Vestermark**
Aalborg University
{kotte17, kveste16}@student.aau.dk

Abstract

Question answering over knowledge graphs (QA-KGs) is a vital topic within information retrieval. Questions with temporal intent are a special case of questions for question answering (QA) systems that has not received a large amount of attention in research. In this paper we propose using temporal knowledge graph embeddings (TKGEs) for temporal QA. We propose MATQA, a microservice-based architecture for building temporal QA systems on knowledge graph embeddings (KGEs). Furthermore, we present a variation of ensemble learning, Bayesian model averaging (BMA), where results of several link prediction tasks on separate different pre-trained TKGE models are combined and re-ranked, before being chosen as the final results. Our experiments on two datasets, ICEWS14 and ICEWS05-15, performed using this variation of ensemble, which we build using the microservice-based architecture, show that it provides better results, than using these TKGE models individually.

1 Introduction

Knowledge graphs (KGs), such as Wikidata [Leblay and Chekol, 2018], Freebase [Google, 2018], DBpedia [Lehmann *et al.*, 2014], and YAGO [Suchanek *et al.*, 2007], increases in size and popularity, KGs are attracting the attention of researchers and practitioners. With larger and more detailed KGs, QA using KGs for applications such as virtual home assistants, chat-bots, and recommender systems show significant performance gain. While QA on KGs is a major area of research [Mohammed *et al.*, 2018; Petrochuk and Zettlemoyer, 2018; Huang *et al.*, 2019; Lukovnikov *et al.*, 2019], little attention is given to QA on temporal KG. Temporal questions can involve both implicit and explicit notions of temporality, such as asking about a specific year, e.g. "Who won the Oscar for best supporting actor in 2019" (explicit) or asking about something that happened at the same time as something else, such as "Who was pope during the fifth crusade?" (implicit). Answers to these questions can also in some cases be answered with a timestamp, such as asking when something happened, or time can be used merely as a constraint for finding the answer to the question, such as asking who was president in a specific year. Complex temporal questions can even contain sub-questions, where answers to these sub-questions can be translated into the temporal constraint for the overall question. An example of this is "Where did Obama's children study when he became president?" where the sub-question "When did Obama become president?" can be solved, and the result of this can be used as temporal constraint for the question "Where did Obama's children study in 2009?". Furthermore, most research into QA-KG [Unger *et al.*, 2012; Jia *et al.*, 2018] utilize the KG itself as a form of database, translating natural language questions into a query language of some form (usually SPARQL), and then queries the KG for a result. Little research [Huang *et al.*, 2019; Jia *et al.*, 2021] has been made into using KGE for QA-systems. Using KGE models allows the system to answer questions under an open world assumption, as link prediction is able to synthesize facts that do not already exist in the KG, whereas using something like SPARQL limits the system to a strict closed world assumption.

In this paper, we present two contributions to the task of QA over temporal knowledge graphs (TKGs):

- MATQA: A microservice-based architecture/framework for building QA-KG systems in an extensible, modular, and scalable manner.
- An ensemble method for combining the results of link prediction tasks on multiple TKGEs, where the combination of results yield better results than using any of the TKGE models individually.

In Section 2 we present background and notation necessary for the understanding of the rest of this paper. Section 3 contains work related to our work. In Section 4 we present the architecture of MATQA, and in Section 5 we explain how ensemble learning can be used to achieve better results that individual models. Finally, in Section 6, we conduct an experimental study on the usage of ensemble learning for QA-KG, where the experimental setting is built using MATQA, and we end with a discussion of results in Section 7 and a conclusion to our work in Section 8.

2 Background and Notation

In this section, we define the most important concepts for describing our system, MATQA.

Knowledge Graph A knowledge graph is a directed graph with labeled vertices and edges. The vertices in a KG represent entities and edges represent relations between the entities. The entities and relations form entity-relation-entity triples called facts. Examples of KGs include Wikidata [Leblay and Chekol, 2018], Freebase [Google, 2018], DBpedia [Lehmann *et al.*, 2014], YAGO [Suchanek *et al.*, 2007], and ICEWS [Boschee *et al.*, 2015].

Fact A fact is an entity-relation-entity triple, also written as (h, r, t), where h is the head entity, r is the relation, and t is the tail entity. An example of a fact is *(Barack Obama, isPresidentOf, USA)*.

Temporal Knowledge Graph A temporal knowledge graph is an extension of a KG, where instead of using regular triple facts, it uses temporal facts that form entity-relation-entity-time quadruples. The time can be represented as either an interval [Leblay and Chekol, 2018] or a discrete timestamp [Boschee *et al.*, 2015]. If the TKG uses intervals, there can either be one temporal fact where the time is represented as an interval, or there can be two temporal facts where one has the start time and the other has the end time. For TKGs that use discrete timestamps only one temporal fact is necessary. KGs without temporal information, will henceforth, in this paper, be referred to as static KGs.

Temporal Fact A temporal fact is a fact, that has been extended to include time and is written as (h, r, t, τ), where τ is the time. If the TKG represents time as an interval, time τ can either be represented as an interval in one fact (i.e. *(Barack Obama, presidentOf, USA, 2009-2017)*) or as two temporal facts, where one of the temporal facts represent start time and the other represent end time (i.e. *(Barack Obama, becomePresidentOf, USA, 2009)* and *(Barack Obama, endPresidentOf, USA, 2017)*). If the TKG represents time as a discrete timestamp, the fact is representing an event or the beginning of an event (i.e. the previously used fact is represented as *(Barack Obama, presidentOf, USA, 2009)*).

Temporal Question A temporal question is a question, where either time is part of the question or the answer to the question is the time. An example of where time is part of the question is *"Who became president of the USA in 2009?"*, and an example of where time is the answer is *"When did Barack Obama become president of the USA?"*.

Answer An answer to a temporal question is an entity, relation, or time depending on the question. This answer is the result of using a TKGE model with the known information from the temporal question. Examples of answers are *(Barack Obama)* for the question *"Who became president of the USA in 2009?"* and *2009* for the question *"When did Barack Obama become president of the USA?"*.

Open and Closed World Assumptions Under the closed world assumption, knowledge not present in the KG is assumed to be false. Under open world assumption, knowledge not present in the KG can still be true, even though it is missing. Most KG operate under the closed world assumption, although KGE models can also often be used to predict knowledge not present in the KG through link prediction.

Link prediction The task of predicting if a fact in a KG is true using known entities and relations, whether or not the fact is in the KG. Link prediction is the most common task for KGE, and relates strongly to QA, as link prediction can be used to determine the likelihood of a fact being true, even if it is not known to be. Thus, if a question is asked, where the answer is not directly known, link prediction can be used to generate likely responses.

3 Related Work

Temporal Knowledge Graph Embeddings Knowledge Graph Embeddings have a long research history [Nickel *et al.*, 2011; Bordes *et al.*, 2013; Wang *et al.*, 2014; Lin *et al.*, 2015; Yang *et al.*, 2015; Trouillon *et al.*, 2016; Sun *et al.*, 2019] and more recently, research has begun on integrating temporal dimensions into these, mainly as variations of non-temporal KGE models [Jiang *et al.*, 2016; García-Durán *et al.*, 2018; Dasgupta *et al.*, 2018; Goel *et al.*, 2020; Jain *et al.*, 2020; Lacroix *et al.*, 2020; Sadeghian *et al.*, 2021]. In this paper, only TKGs are relevant. The focus of this paper is not to create a TKGE model, we explore existing state-of-the-art models. As a previous study shows [Otte and Vestermark, 2022], ChronoR [Sadeghian *et al.*, 2021] provides state-of-the-art results for link prediction tasks, which will be our main task on TKGs. ChronoR is inspired by rotational KGE methods, such as RotatE [Sun *et al.*, 2019], and uses the linear transformations of rotation and scaling, parameterized by time and relation, to get the embedding of a tail entity from a head entity. Unfortunately, as shown in [Otte and Vestermark, 2022], ChronoR does not provide freely available source code. The Diachronic Entity Embedding models [Goel *et al.*, 2020], and TimePlex [Jain *et al.*, 2020] do however provide available source code for us to use. The Diachronic Enitity Embedding models, titled DE-SimplE, DE-DistMult and DE-TransE, are all variations of non-temporal, or static, KGE models, that all embed time with entities, inspired by diachronic word embeddings. TimePlex [Jain *et al.*, 2020] is based on ComplEx [Trouillon *et al.*, 2016], and similarly, embeds entities, relations, and time as vectors in complex space.

Question Answering on Static Knowledge Graphs One of the most frequently researched use-cases of KGs are QA systems [Mohammed *et al.*, 2018; Petrochuk and Zettlemoyer, 2018; Huang *et al.*, 2019; Lukovnikov *et al.*, 2019]. QA on static KG usually do not use KGEs, however methods like KGEQA [Huang *et al.*, 2019] uses the embeddings of a KG from TransE or TransR. KGEQA uses recurrent neural networks, specifically a bi-LSTM, to find the embeddings of an entity and relation. The entity and relation is then used in a link prediction task to find the answer to the question. Other methods that also use recurrent neural networks are UPSQA [Petrochuk and Zettlemoyer, 2018] and BuboQA [Mohammed *et al.*, 2018]. UPSQA finds the top-k entities in the question using a bi-LSTM and then finds the relation using another bi-LSTM, given the question and a candidate entity. BuboQA also does entity detection using a bi-LSTM, and then uses fuzzy matching to determine the entity in the KG. The relation is then found using a bi-GRU over all the relations that are used with the found entity. Some methods, like BERTQA [Lukovnikov *et al.*, 2019], use BERT models. The BERT models are used 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.

Question Answering on Temporal Knowledge Graphs QA on temporal KG is a very recent topic of research, with systems such as TEQUILA [Jia *et al.*, 2018] and EXAQT [Jia *et al.*, 2021].

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, and then decomposes this question into a non-temporal sub-question, which can then be answered by the underlying QA-KG system, and a temporal constraint. The temporal constraint can then be solved using constraint reasoning on temporal intervals. Thus, TEQUILA uses TKGs for QA, but not TKGE models, as our contribution does.

EXAQT answers complex temporal questions with multiple entities, relations and associated temporal conditions using TKG. It does this in two steps. Firstly, question-relevant compact sub-graphs are computed within the KG and are enhanced with temporal facts using Group Steiner Trees and BERT models. Secondly, relational graph convolutional networks are created from the output of the first step, and these are enhanced with time-aware entity embeddings and attention of temporal relations. As such, EXAQT could be considered to use a form of TKGEs for the second step, although it does not use pre-existing state-of-the-art embedding models as our contribution does, but rather creates its own.

4 MATQA

We propose a microservice-based architecture for TKG based QA. A microservice-based architecture allows for a high degree of modularity between components, as they will be loosely coupled and communicate with APIs. Furthermore, microservices bring an inherent level of scalability. The architecture is based on a number of microservice modules with individual responsibilities. These are: the User Client/User Interface Module, the Natural Language Module, the TKGE Module/Modules, and optionally, the Ensemble Module. A model of this architecture is shown in Figure 1, where 1a shows a model without the Ensemble Module, and 1b, with the module. Each of these modules communicate with each other through outward facing APIs, which allows modules to be easily added, changed, or replaced based on the users specific task. In Section 4.5, we explain how we implement this API. We will now go through each of the modules and explain their purpose, API, and our current implementation.

4.1 User Client/User Interface

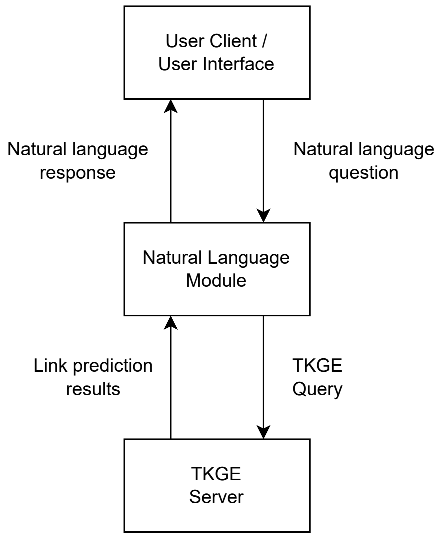
The purpose of this module is to allow a user to interface with the system in a user friendly manner. The module itself does not have an API, but it calls the API of the Natural Language Module. Our current implementation consists of a very simple command line interface, which allows a natural language question as a parameter. Other parameters for such an interface include number of answers to show, as QA based on KGs are often not perfect, and as such, it may be relevant to show multiple possible answers, as well as whether or not to show a confidence score for each possible answer. It would also be possible to implement a more sophisticated user interface, such as a web UI, or even skip using this module entirely, and use the Natural Language Module's API to allow other programs to interface with this system.

4.2 Natural Language Module

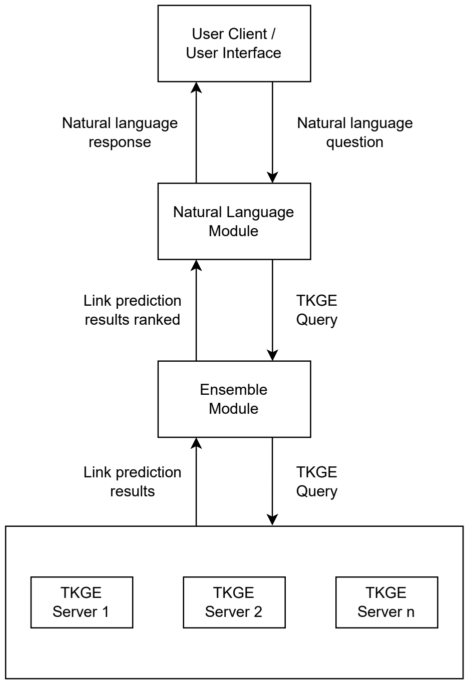
The Natural Language Module is responsible for translating natural language questions into TKG link prediction queries, such that these can be answered/solved by the underlying TKG. These queries consist of a TKG fact, where one of the members (head, relation, tail, or time) is missing, e.g. (h, r, ?, τ). After the query has been answered, the Natural Language Module then converts the answer-fact into a Natural Language response, which will be returned to the caller. Thus, the API of the module requires only a single string parameter (the natural language question), and returns a single string (the natural language response). Our implementation of this module does not include any actual natural language processing, instead it converts questions of a specific form (that we have generated, and thus can guarantee the form of), into queries, based on very simple logic. The questions are of the form "Who did *head relation* on *time*?", and "Who *relation tail* on *time*?", which has been generated from the relevant TKG dataset. We leave it as future work to implement a Natural Language Module that utilizes more advanced forms of natural language processing.

4.3 TKGE Server

The TKGE Server Module is the module responsible for solving link prediction queries provided by the Natural Language Module. This can theoretically be accomplished in any manner, although the purpose of the framework is to allow for different TKGE models to solve these queries. The API exposes a single endpoint, which as input parameter takes a link prediction query and then responds with the corresponding temporal fact. The simplicity of the API allows for developers to implement any TKGE model they want. We include an implementation, which covers 3 similar TKGE models from [Goel *et al.*, 2020]. These are DE-SimplE, DE-DistMult, and DE-TransE, all of which are variations of static KGE models, where Diachronic Entity Embeddings has been added, allowing for the time aspect to be modeled. We choose the models as [Otte and Vestermark, 2022] finds that these models provide freely available source code (along with TimePlex [Jain *et al.*, 2020]), and provide state-of-the-art results. Furthermore, we were able to implement the TKGE server for these models without substantial changes to the source code, which in the case of TimePlex [Jain *et al.*, 2020], we were not able to. We were unable to make TimePlex work as a pre-trained model for our architecture, as it had a different method for achieving scores of facts, that would not allow us to use it, without substantial changes to it's source code. Thus, we choose not to further explore using TimePlex as a TKGE model, in this paper.


User Client / User Interface
Natural language response
Natural language question
Natural Language Module
Link prediction results
TKGE Query
TKGE Server

(a) Module configuration with a single TKGE server and no ensemble module.



(b) Module configuration with ensemble module wrapping multiple TKGE server modules.

Figure 1: System architecture, showing the modules of the system, in the two different possible configuration.

4.4 Ensemble Module

The Ensemble Module is an optional addition that can be placed between the Natural Language Module and the TKGE server module(s). This module allows the system to encompass more than one TKGE server, combining the results from each of these, into a single response. The Ensemble Module uses the exact same API as the underlying TKGE server(s), which means that it can be placed between the Natural Language Module and the TKGE server(s) without any change to these. The Ensemble Module will be further explained in Section 5.

4.5 REST vs RPC

When implementing the APIs of the individual modules, we had to make a decision about how these APIs should be implemented. The decision was between the two most common methods: REST and RPC, where REST is the most common of these. Remote Procedure Call (RPC) is a inter-process communication method that allows a program to call a procedure of another program, as if it were part of it's own code. This allows for programmers to easily integrate the use of these procedures into their own code without thinking about how the communication between these works. gRPC[1] 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 allows for generation of language specific code, that implements the client and server part of services defined in the Protocol Buffer definition in a large amount of different programming languages, such as Go, C++, Java and Python. This allows developers a large degree of freedom for choosing which language they wish to develop both client and server-side systems in, while still behaving as if all the code exists in the same code base. Since we do not need the low-level access provided by REST APIs, and because our APIs do not need to be public facing, and do not need to be easy for third party applications to use, we choose to use RPC, and specifically gRPC, for our APIs.

[1]https://grpc.io/

5 Ensemble

We propose a method for increasing the accuracy of temporal link prediction using TKGE models, by utilizing a variation of ensemble learning. Ensemble learning are techniques used to give better predictive performance where multiple models are used together [Opitz and Maclin, 1999]. We propose using BMA [Hoeting *et al.*, 1999] to obtain a better result than the individual models. BMA uses the models individually and combines the scores each model predicted. BMA can be used both as an unweighted and weighted ensemble. This method uses pre-trained models, which means that many different models that capture different features can be used to improve weaknesses. We propose using BMA as both an unweighted and a weighted ensemble. For both methods the top-n scoring results from a temporal link prediction query is combined and re-ranked based on the combined score. The unweighted method takes the top-N results from each of the models, and gives them a score equal to their ranking within the top-N, for that model. The score s is defined as $s = N - r$, where N is the number of returned results and r is the rank. For example, if a model has given a result the rank of 0 (the model predicts this result is the most likely), between 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 2 shows the process of combining results from multiple TKGE servers, using this unweighted method.

The weighted method works similarly to the unweighted, but multiplies weights to the scores given before combination, such that different models may have a larger impact on the final score. This allows a developer greater freedom over which models are used as TKGE servers, as models that tend to under perform, compared to others, can be given a lower weight, and thus the results can still be used, as opposed to ignoring these models completely. This essentially allows us to use under performing models, to improve the accuracy of state-of-the-art models, by combining them, and giving the under performing model a lower weight. This method is shown in Figure 3.

We also propose using bootstrap aggregating to improve the accuracy. Bootstrap aggregating is a method of generating new datasets, by taking subsets of the original dataset, and then training multiple models with these subsets, and combining them using ensemble learning [Opitz and Maclin, 1999]. We create three bootstrap datasets, all the same size of the original dataset. As the bootstrap datasets are sampled with replacement, approximately 63% of the bootstrap dataset are unique samples, while the rest are duplicates [Aslam *et al.*, 2007]. However, only creating three bootstrap datasets does present some possible issues. As only approximately 63% of the original dataset is in each bootstrap dataset there is the possibility that some of the facts are not present in any of the bootstrap datasets. Initial testing with bootstrap aggregating models yield subpar results, shown in Appendix A, and as such, we will not explore this method further.

6 Experimental Study

6.1 Dataset

The most used datasets for state-of-the-art TKGE models are Integrated Crisis Early Warning System (ICEWS), namely the two subsets ICEWS14 and ICEWS05-15 [Otte and Vestermark, 2022; Boschee *et al.*, 2015], which feature facts from the ICEWS dataset from the year 2014 and the years 2005-2015, respectively. These datasets are event-based meaning every fact has a discrete timestamp. Since the datasets have a timestamp for every fact, these datasets have proven very good for TKGEs models. For the natural language processing (NLP) module, natural language questions are necessary. We generate simple questions from the ICEWS datasets by making a question with one part of the fact missing (i.e. the fact *(South Korea, Criticize, North Korea, 2014-05-13)* with the tail missing becomes "*Who did South Korea criticize on 13 May, 2014?*"). For each fact in the datasets four questions are generated; two with an entity missing, one with the relation missing, and one with the time missing. Table 1 shows a summary of the datasets. Even though, we generate questions with missing relation and time, all of the used TKGE models, do not support link prediction for missing relation and time, and as such, these questions are not used in the experiments. Only the questions that correspond to link prediction queries with missing head or tail are used.

6.2 Baselines

As a benchmark to test how good the TKGE server is, we test the Diachronic Entity Embedding models using the original source code as a baseline, and create an evaluation script that simulates the Natural Language module by sending TKGE queries to the TKGE server. When running these experiments, only the facts in the test set are loaded. This means the results are unfiltered and therefore appear worse. The evaluation part of the Diachronic Entity Embedding source code includes both filtered and unfiltered metrics, and we, therefore, compare our own results to the unfiltered results from the Diachronic Entity Embedding source code.

6.3 Metrics

All systems return a ranked list of the answers from link prediction. The answers consists of the facts and a score for each fact. With the list of answers we calculate the following metrics:

- MRR: The mean of the reciprocal of the rank of the correct answer.
- Hits@1: The percentage of facts, where the answer with the highest score is the correct answer.
- Hits@10: The percentage of facts, where the correct answer is within the 10 highest scored answers.

For each of the three metrics a higher score is better.

6.4 Testing Baselines and Ensemble

After running the tests on both the original source code and the TKGE server, we conclude we get the same unfiltered results. To test the ensemble, we use the three Diachronic Entity Embedding models, DE-SimplE, DE-DistMult, and DE-TransE, and run the evaluation in a configuration similar to the architecture shown in Figure 1b. The ensemble uses an unweighted scorer, that scores as mentioned in Section 5. Results of the models individually and the ensemble can be seen 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 under performing compared to DE-SimplE. The results are slightly worse on the ICEWS05-15 dataset, when using ensemble, compared to the best individual model. It should also be mentioned that all the Diachronic Entity Embedding models perform better on the ICEWS14 dataset in both Hits@1 and MRR, compared to the ICEWS05-15 dataset.

6.5 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 weighted results to

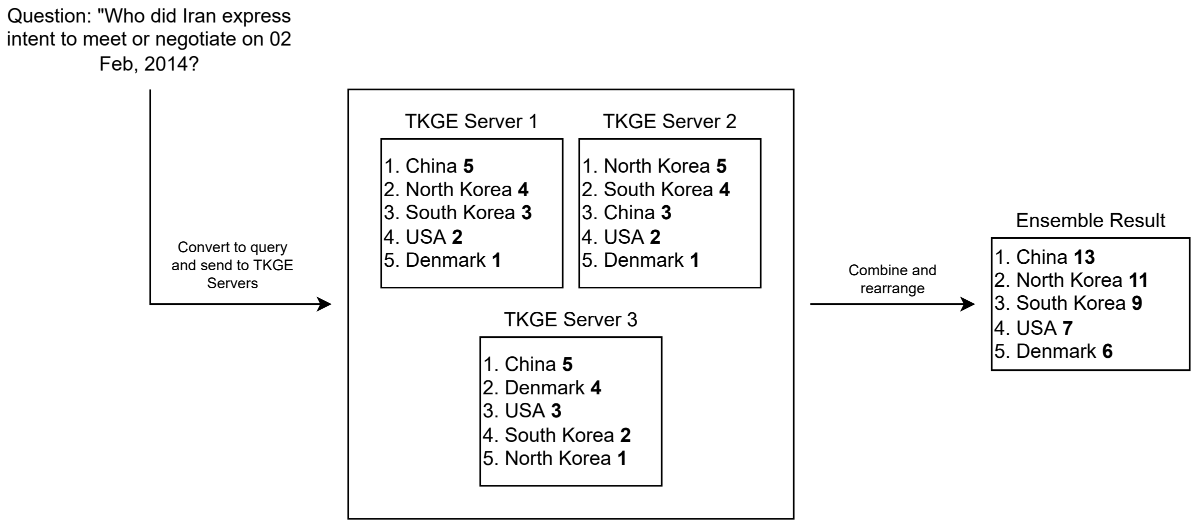


Figure 2: Process of combining results from TKGE servers, using the unweighted ensemble method. Scores given to the rankings, both before and after combination, are shown in bold.

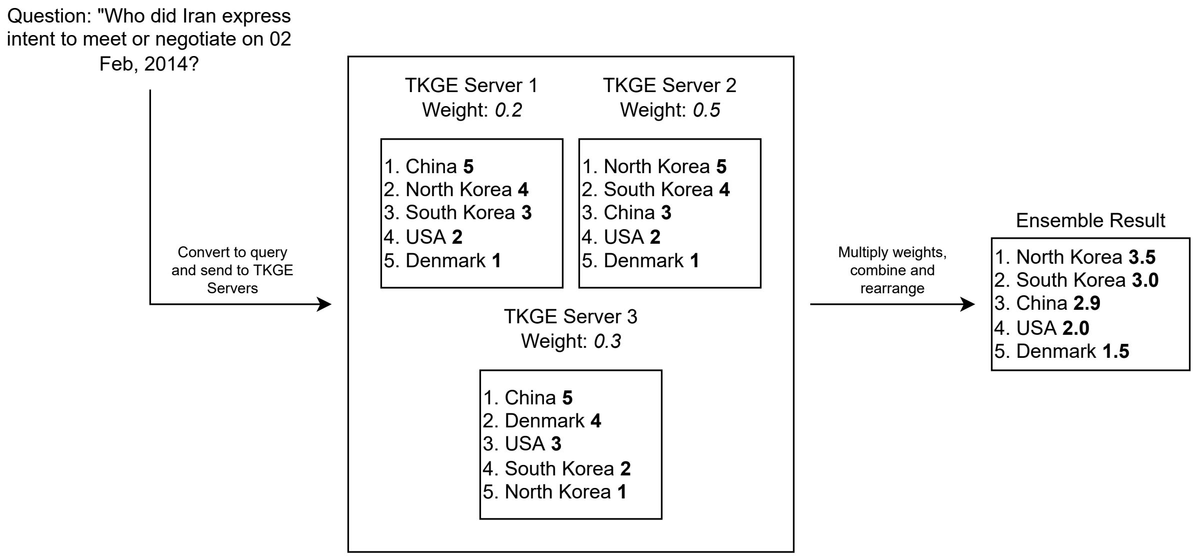


Figure 3: Weighted method of combining ranked results from TKGE servers. Results are multiplied by the weight given to the server, before they are added and rearranged.

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 1: The number of entities, relations, distinct timestamps as well as the number of facts for ICEWS14 and ICEWS05-15.

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	0.515	39.3	74.5	0.493	36.6	73.8
Ensemble (tuned)	0.518	39.6	74.7	0.497	36.9	74.2

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

see if eliminating some potential noise yields better results. To perform weighted ensemble, we first need to find optimal weights for each of the used models. To approximate the optimal weights we use Bayesian optimization using Gaussian Process [Mockus, 1989], instead of doing an exhaustive grid search, as approximation is likely good enough to prove that ensemble can perform better than the best individual model. 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 on the ICEWS14 dataset, and the approximated optimal weights found was; 0.48 for DE-SimplE, 0.4 for DE-DistMult, and 0.12 for DE-TransE. The found weights are then used for the ICEWS05-15 dataset as well. Results for the weighted ensemble is shown as Ensemble (tuned) in Table 2. The results show that on the ICEWS14 dataset, using a weighted ensemble further improves the accuracy and has a significant improvement over DE-SimplE, and has a slight improvement on the ICEWS05-15 dataset comepared 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 theorize that this is because the models are able to capture different aspects of the KG, so even if DE-TransE performs bad on it's own, it might be able to capture aspects that the other models do not, and thus a combination of them, provides even better results.

7 Discussion

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

Framework We chose to implement MAQA using a microservice-based architecture, as we wanted to have a very loosely coupled and modular framework, where components can easily 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. Furthermore, microservices are inherently scalable when data is independent, as is the case with the individual questions, which will be passed through our system. This means that a load balancer can be placed between the caller and the called module, which can then send individual question answering requests, to the microservice with the least load.

BMA As the results show, different models using BMA gives better results individual models used in the ensemble. This was shown with the Diachronic Entity Embedding models DE-SimplE, DE-DistMult, and DE-TransE on the ICEWS subsets ICEWS14 and ICEWS05-15. On ICEWS14 even using an unweighted ensemble proved to produce better results than DE-SimplE, which has the best accuracy of the three models. The weights found using Bayesian optimization further increased accuracy of the ensemble. As the weights were approximated using Bayesian optimization, the weights are most likely not the optimal weights. Running an extensive grid search would most likely give better results, as it was enough to approximate the weights to get better results than the individual models. The weights used for on the ICEWS05-15 dataset also were not approximated for that dataset, but were instead approximated on ICEWS14. This was done because of the large size of the ICEWS05-15 dataset, which would have meant a large time investment would have to be spent, doing this approximation. As such, there are likely better possible weights for that dataset. 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 different aspects of the temporal dimension.

Bootstrap aggregating As we only created three bootstrap datasets, the size of the out-of-bag dataset, the set of facts not used to train any of the models, is still relatively large. This can potentially decrease accuracy drastically, as there possibly are a large number of entities and relations, that neither of the models have used for training other than negative sampling. As with the ensemble of DE-SimplE, DE-DistMult, and DE-TransE, using the three bootstrap models in an ensemble did improve performance, but aggregating still had very poor accuracy compared to the non-bootstrap models. Further exploring of the impact of bootstrap aggregating on KGEs, will be left for future work.

Open World Assumption As previously explained, when using a KGE model for a QA-system, it is possible to answer questions under a open world assumption, by performing link prediction on known entities and relations, but in unknown fact configurations. One could imagine a QA-KG system, where standard querying of the KG, using a query language such as SPARQL, could be combined with a similar link prediction task, to achieve better results. The system could query the KG and if an answer is available, simply use that, but if not, it could convert the query into a link prediction task, and supply it to the KGE model, which could then provide a better-than-none result. We leave the building of such a system for future work.

8 Conclusion

We proposed MAQA, a microservice-based framework for QA-KG. This framework has shown to be modular, as we were able to use it with many different TKGE models, extensible, as we were able to substitute the TKGE server module for an ensemble module, and scalable, since the data between different microservices are independent, and this means that the microservices will inherently be scalable. We also proposed to use an ensemble method for combining multiple TKGE models to provide better results. We chose to use BMA, as it uses pre-trained models and allowed us to use different Diachronic Entity Embedding models, DE-SimplE, DE-DistMult, and DE-TransE. Using BMA was shown to provide better results on the ICEWS datasets, than using the models individually, which indicates that different models might capture different aspects of the KG, which when combined provide superior results. This particular theory, was enhanced by the fact that a far inferior model, DE-TransE, was still able to provide better results, in tandem with the other superior models, than when it was completely absent.

References

[Aslam *et al.*, 2007] Javed Aslam, Raluca Popa, and Ronald Rivest. On estimating the size and confidence of a statistical audit. *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, pages 8–8, 01 2007.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

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

[Dasgupta *et al.*, 2018] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[García-Durán *et al.*, 2018] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[Goel *et al.*, 2020] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3988–3995, Apr. 2020.

[Google, 2018] Google. Freebase data dumps, 2018.

[Hoeting *et al.*, 1999] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: a tutorial. *Statistical Science*, 14(4):382 – 417, 1999.

[Huang *et al.*, 2019] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 105–113, New York, NY, USA, 2019. Association for Computing Machinery.

[Jain *et al.*, 2020] Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3733–3747, Online, November 2020. Association for Computational Linguistics.

[Jia *et al.*, 2018] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1807–1810, 2018.

[Jia *et al.*, 2021] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. *Complex Temporal Question Answering on Knowledge Graphs*, page 792–802. Association for Computing Machinery, New York, NY, USA, 2021.

[Jiang *et al.*, 2016] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. Encoding temporal information for time-aware link prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2354, Austin, Texas, November 2016. Association for Computational Linguistics.

[Lacroix *et al.*, 2020] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*, 2020.

[Leblay and Chekol, 2018] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1771–1776, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[Lehmann *et al.*, 2014] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo

Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6, 01 2014.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[Lukovnikov *et al.*, 2019] Denis Lukovnikov, Asja Fischer, and Jens Lehmann. Pretrained transformers for simple question answering over knowledge graphs. In *The Semantic Web – ISWC 2019*, page 470–486, Berlin, Heidelberg, 2019. Springer-Verlag.

[Mockus, 1989] Jonas Mockus. *Bayesian Approach to Global Optimization*. Springer Netherlands, 1989.

[Mohammed *et al.*, 2018] Salman Mohammed, Peng Shi, and Jimmy Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Nickel *et al.*, 2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA, 2011. Omnipress.

[Opitz and Maclin, 1999] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *J. Artif. Int. Res.*, 11(1):169–198, jul 1999.

[Otte and Vestermark, 2022] Kristian Otte and Kristian S. Vestermark. A survey of temporal knowledge graph embedding approaches, 2022.

[Petrochuk and Zettlemoyer, 2018] Michael Petrochuk and Luke Zettlemoyer. SimpleQuestions nearly solved: A new upperbound and baseline approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 554–558, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[Sadeghian *et al.*, 2021] Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. ChronoR: Rotation based temporal knowledge graph embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6471–6479, May 2021.

[Suchanek *et al.*, 2007] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 697–706, New York, NY, USA, 2007. Association for Computing Machinery.

[Sun *et al.*, 2019] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016.

[Unger *et al.*, 2012] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 639–648, New York, NY, USA, 2012. Association for Computing Machinery.

[Wang *et al.*, 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1112–1119. AAAI Press, 2014.

[Yang *et al.*, 2015] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May 2015.

A Bootstrap aggregating

Model	MRR	Hits@1	Hits@10
DE-SimplE bootstrap 1	0.443	33.1	65.3
DE-SimplE bootstrap 2	0.440	32.8	65.0
DE-SimplE bootstrap 3	0.443	33.3	65.1
Bootstrap aggregating ensemble	0.450	33.6	66.6

Table 3: Results of DE-SimplE bootstrap aggregating models individually and ensemble on a link prediction task using the TKGE server on ICEWS14.