MST5 — A Transformers-Based Approach to Multilingual Question Answering over Knowledge Graphs

Nikit Srivastava[1], Mengshi Ma[2], Daniel Vollmers[1], Hamada M. Zahera[1], Diego Moussallem[2] and Axel-Cyrille Ngonga Ngomo[1]

[1]Data Science Group, Heinz Nixdorf Institute, Paderborn University, Germany
[2]Paderborn University, Germany

Abstract

Knowledge Graph Question Answering (KGQA) simplifies querying vast amounts of knowledge stored in a graph-based model using natural language. However, the research has largely concentrated on English, putting non-English speakers at a disadvantage. Meanwhile, existing multilingual KGQA systems face challenges in achieving performance comparable to English systems, highlighting the difficulty of generating SPARQL queries from diverse languages. We introduce a streamlined, multilingual KGQA framework that injects linguistic context (e.g., POS tags, dependency relations) and candidate entity identifiers directly into a single pretrained multilingual transformer. Unlike prior approaches that employ separate encoders to handle auxiliary signals, our method lets the same model process both the question and the supplemental data, markedly improving its ability to translate a natural-language query into an accurate SPARQL statement. It demonstrates promising results on the most recent Wikidata-based QALD datasets, namely QALD-9-Plus and QALD-10. Furthermore, we introduce and evaluate our approach on Chinese and Japanese, thereby expanding the language diversity of the existing datasets.

1. Introduction

The aim of research in Knowledge Graph Question Answering (KGQA) is to establish an interactive methodology enabling users to access extensive knowledge stored within a graph-based model via natural language queries [1]. Recent research efforts have witnessed a notable upsurge in addressing KGQA concerns [2]. However, it is noteworthy that a substantial proportion of these systems is confined to the English language domain [3, 4, 5, 6, 7, 8]. Furthermore, among the currently available multilingual systems [9, 10, 11], only a limited subset of widely spoken or written languages are supported [12, 13]. Moreover, majority of these systems do not achieve performance levels comparable to those attained in English when addressing questions in other languages. This difference highlights the inherent difficulty KGQA systems face in processing the multilingual natural language queries by trying to recognize the hidden, repeated patterns that are common across different languages. This poses a challenge for the vast majority of web

Wikidata'25: Wikidata workshop at ISWC 2025
nikit.srivastava@uni-paderborn.de (N. Srivastava); mengshim@mail.uni-paderborn.de (M. Ma); daniel.vollmers@uni-paderborn.de (D. Vollmers); hamada.zahera@uni-paderborn.de (H. M. Zahera); diego.moussallem@uni-paderborn.de (D. Moussallem); axel.ngonga@uni-paderborn.de (A. N. Ngomo)


users whose native language is not English.

Knowledge Graphs (KGs) have been conceptualized as language-agnostic solution for the organization and storage of information [14, 15]. As a result, the approach to KGQA paves the way for retrieving language-agnostic answers, contingent upon its ability to comprehend natural language questions in a multilingual context. Recent advancements in the field of language modeling have made numerous state-of-the-art language models freely accessible [16, 17, 18, 19]. These language models can be effectively employed within a sequence-to-sequence task setting to tackle KGQA. In this setting, the input sequence corresponds to the natural language question, while the output entails a relevant SPARQL[1] query required to extract the answer from an underlying KG. This method of carrying out KGQA is also known as Semantic Parsing [20], it generates an intermediate representation (i.e. SPARQL) that is interpretable by humans, enabling them to understand how the model formulates specific answers within a multilingual context.

To overcome the limited availability of multilingual KGQA and the shortcomings of existing systems, we adopt a strategy similar to Rony et al. [8] that entails the inclusion of linguistic context as an auxiliary input to a language model. Unlike the previous methods, however, we employ a single encoder-decoder transformer model rather than creating separate encodings for the auxiliary input. We do this in order to leverage the attention mechanism embedded within the language model to facilitate the creation of an implicit understanding and representation of the provided linguistic context. Furthermore, we apply entity disambiguation tools to extract pertinent entity information, which is then also added to the auxiliary input. Our method establishes a seamless end-to-end system that can respond to multilingual queries using only the text of the input question.

Our findings demonstrate that incorporating linguistic context and entity information significantly enhances the KGQA performance. Our approach yields promising results on our benchmarking datasets. Moreover, to substantiate the efficacy of our approach, we introduce and evaluate its performance in previously unrepresented languages within the QALD datasets. Specifically, we added Japanese to the QALD-10 dataset and both Chinese and Japanese to the QALD-9-Plus dataset. The inclusion of these non-European languages introduces a novel dimension to the datasets, given their distinctive structural characteristics when compared to the existing language set. Additionally, we make our source code publicly accessible for the benefit of the research community.[2] In essence, our approach addresses the following research question: *How can the simplified integration of linguistic context and entity information into language models enhance their performance on multilingual Knowledge Graph Question Answering tasks?*

2. Related Work

The KGQA task is commonly solved by Semantic Parsing approaches, that create logical queries for given natural language questions [20]. Since SPARQL is used as a standard query language

[1] https://w3.org/TR/rdf-sparql-query/
[2] https://github.com/dice-group/MST5

for the RDF-based Knowledge Graphs,[3] we investigate the systems that adopt this particular approach to KGQA. In this section, we cover the relevant previous works that have developed systems based on Language Model (LM) for the SPARQL query generation and the KGQA systems that are multilingual.

2.1. LM-based SPARQL query generation

As one of the early works, Soru et al. [5] introduced Neural SPARQL Machines (NsPM) where SPARQL is regarded as an analogous language, it utilized Neural Machine Translation (NMT) to transform natural language questions into SPARQL queries, while avoiding language heuristics, statistics, or manual models. The architecture consists of three components: a `generator` that creates SPARQL queries from query templates for training the model, a `learner` based on Bidirectional Recurrent Neural Networks (Bi-RNNs) [21] to learn to translate input questions to encoded SPARQL, and an `interpreter` for reconstructing SPARQL queries from their encoded representation and retrieving results from a knowledge graph. Their modular design allowed integration of various machine translation models and generating SPARQL queries for diverse knowledge graphs.

In the recent years, Borroto et al. [6] produced a similar AI system for KGQA, where the architecture uses a NMT module based on Bi-RNNs [21]. They trained it in parallel to a Named Entity Recognition (NER) module, implemented using a BiLSTM-CRF network [22]. The NMT module translates the input natural language question into a SPARQL query template, whereas the NER module extracts the entities from the question. The two modules' outputs are combined to form the resulting SPARQL query. Departing from the template oriented methodology, Rony et al. [8] proposed a novel approach to tackle the challenges associated with generating SPARQL queries from natural language questions. The authors introduced a new embedding technique that encodes questions, linguistic features, and optional knowledge to allow the system to comprehend complex question patterns and graph data for SPARQL query generation. They augmented the input embedding with the extracted embeddings and then fed it to a decoder-only (GPT-2 [23]) language model to generate SPARQL query. However, it is important to highlight that the investigation conducted in the study did not assess the effectiveness of the resulting SPARQL queries in retrieving answers from a KG. This limitation stems from the fact that the generated SPARQL queries are often syntactically incorrect. Furthermore, certain other limitations are observed within this approach, particularly regarding the selection of the language model architecture and the utilization of separate embedding layers for encoding the linguistic context.

2.2. Multilingual KGQA

When it comes to the multilingual KGQA systems, Burtsev et al. [9] introduced a conversational system called DeepPavlov that operates by utilizing a suite of extensive language-dependent deep learning models. These models are used to execute a spectrum of tasks encompassing query type prediction, entity recognition, relation identification, and path ranking that can be

[3] https://w3.org/RDF/

applied to address KGQA alongside many other uses-cases. Developing it further, Zharikova et al. [24] introduced DeepPavlov-2023, which is an improved version of its predecessor.

Pellissier Tanon et al. [10] presented a KGQA methodology named Platypus, which adopts a dual-phase strategy. Initially, it leverages a semantic parser to convert questions posed in natural language into SPARQL queries. Subsequently, these SPARQL queries are executed on the Wikidata knowledge base to procure answers. The semantic parser employed is of a hybrid nature, integrating grammatical (rule-based) and template-based approaches. Grammatical approaches are utilized to extract the syntactic structure of the question and to identify any mentioned entities and relations. Following this, template-based methods are employed to populate predefined templates that are tailored to various types of questions. Building towards an end-to-end multilingual KGQA solution, Diefenbach et al. [11] introduced an approach named QAnswer that generates a SPARQL in four steps. First, it fetches the relevant resources from the underlying KGs. Then, it generates a list of possible query templates. Afterwards, the query candidates are created and ranked using the output from the previous two steps. Finally, a confidence score is computed for each of the ranked queries. QAnswer claimed to be state-of-the-art at the time, and remains a top contender in the QALD challenges. However, one of the biggest downside of this system is not being open-source.

Looking at Machine Translation (MT) as a viable alternative, Perevalov et al. [12] experimented with various MT systems to extend the supported languages of existing multilingual KGQA systems. Their findings indicate that, in most cases, translating questions to English resulted in the highest performance. Furthermore, they noted a small to moderate positive correlation between the quality of machine translation and the question-answering score. In a similar fashion, Srivastava et al. [13] adopted an entity-aware MT approach for the KGQA use-case. Where they highlighted the need for effective translation for the entity labels between languages. By using an entity-aware MT pipeline and translating each question to English, they created a multilingual KGQA approach that performs better than the traditional MT-based methods previously mentioned.

3. MST5 Approach

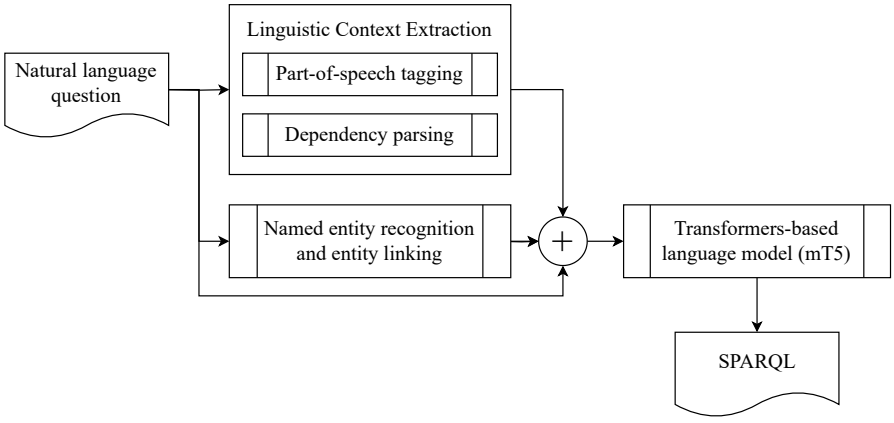
Our method, MST5, enriches a transformer-based model (mT5 [17]) with auxiliary knowledge i.e., entity-link tags and linguistic context to build a semantic-parsing KGQA system. The next subsections first give a formal mathematical definition of the task and then detail the architecture that implements it.

3.1. Problem Definition

Given a natural language query Q and auxilliary information A which consists of linguistic context and entity information, we train the transformer-based model $\mathcal{M}$ parameterized by θ, aiming to generate a syntactically and semantically correct corresponding SPARQL query $\hat{S}$ as:

$$\hat{S} = \arg\max_{S'} P(S' \mid Q, A; \theta)$$

Figure 1: An overview of the MST5 approach (from left to right). First, linguistic context and entity information is extracted from the input question. Then, the extracted information is appended to the input before being passed on to the language model. The language model generates the resulting SPARQL query.



Here P is the conditional probability distribution function and S' represents a candidate SPARQL query generated by the model.

The model is trained in a manner akin to mT5 [17], focusing on minimizing the negative log likelihood ($\mathcal{L}$) of the correct SPARQL query given the inputs:

$$\mathcal{L}(\theta) = -\log P(S \mid Q, A; \theta)$$

Here S refers to the reference (correct) SPARQL query.

3.2. MST5 Architecture

To obtain the auxiliary information from a given input, we create individual modules to perform entity recognition and disambiguation along with linguistic feature extraction. The extracted information is then used to compose the final input sequence to our language model by concatenating it with the input question. Figure 1 provides a high-level overview of our approach. Following are the detailed description of these steps:

3.2.1. Named Entity Recognition and Linking

In this step of our approach we make use of NER and Named Entity Disambiguation (NED) approaches. To fulfill this requirement, we rely on the Named Entity Aware Machine Translation (NEAMT)[4] tool introduced by Srivastava et al. [13]. While the primary purpose of this tool is to perform entity-aware machine translation, its API allows us to skip the translation part entirely and only perform entity recognition and linking tasks. Given its multilingual capabilities, NEAMT meets our requirements.

[4]https://github.com/dice-group/LFQA/tree/main/naive-eamt

3.2.2. Linguistic feature extraction

For enhancing the understanding of the language model, we explicitly include linguistic features in a manner akin to the approach described by Rony et al. [8]. Unlike their approach, we directly append the features to the input sequence. To this end, we extract Part-of-Speech (POS) tags and generate the dependency tree for the input question. For extracting these features, we rely on the spaCy tool.[5] For each distinct language, we use the respective spaCy model.[6]

3.2.3. Preprocessing the target SPARQL-queries

To generate SPARQL queries through a LM, existing works similar to Banerjee et al. [25] introduce a separate vocabulary to make it easier for a LM to predict the resulting SPARQL. However, we recognize that this incorporation introduces additional complexity. Thus, we employ minimal preprocessing during training phase to avoid the need for an expanded vocabulary. SPARQL queries often include prefixes to shorten resource identifiers (URIs). For example:

```
https://www.wikidata.org/entity/Q5  ->  wd:Q5
```

To simplify the target sequences for the mT5 model, we remove the prefix section from the query. We use the prefix form, e.g., `wd:Q5` instead of the full URIs in the query. During inference, we add all common prefixes back to the predicted query to restore its syntactic correctness. As an additional step, we replace special characters such as the question mark (`?`), which signals a variable in a SPARQL query, and curly braces (`{}`), with standardized string placeholders. This modification is due to our observation that the mT5 model struggles with accurately predicting these symbols. During the inference phase, we revert these placeholders to their original symbols to facilitate the retrieval of answers from a SPARQL endpoint.

3.2.4. Creating Input Data for the MST5 Model

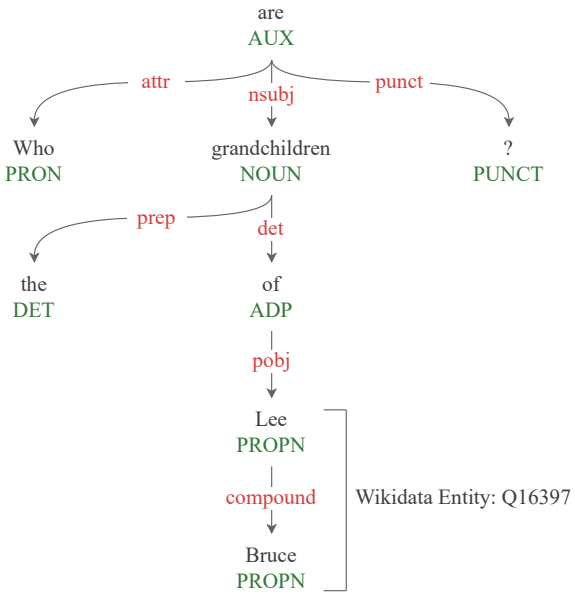
As input to our model is a token sequence, we concatenate the textual question with the linguistic features and entity links with the help of custom separator tokens. As a result, each input sequence incorporates the features listed below. Figure 2 offers an overview of these features:

1. Question, e.g., *Who are the grandchildren of Bruce Lee?*
2. POS tags, e.g., *PRON AUX DET NOUN ADP PROPN PROPN PUNCT*
3. Dependency tree tags, e.g., *attr ROOT det nsubj prep compound pobj punct*
4. Depths in the dependency tree, e.g., *2 1 3 2 3 5 4 2*
5. Entity links, e.g., *Q16397*

Padding tokens are appended to each input feature, ensuring that the tokens for every input feature begin at a consistent token index within the tokenized input sequence. This step enables the transformers-based LM to attend and contextualize the relation between the original query and the auxiliary features properly. Finally, we apply a tokenizer to the concatenated sequence

[5] https://spacy.io/
[6] https://spacy.io/usage/models

Figure 2: An example of linguistic context including dependency parsing (red) and POS-tags (green) alongisde the disambiguated entity for the text: *Who are the grandchildren of Bruce Lee?*



to produce valid inputs for MST5 model. We use the SentencePiece [26] tokenizer for mT5 [17] model as described in the model documentation.[7]

4. Experimental Setup

This section provides a comprehensive description of the training and benchmarking datasets used in our experiments, along with an in-depth explanation of our model training methodologies. We have organized our model training into two main categories: Ablation Study and Evaluation of Optimal Models. This structure helps with systematic investigation of the effects of various model configurations and enables us to concentrate on the best-performing models for additional analysis. Our current experimental setup is aimed at addressing the following research questions:

1. *Whether the simplified auxilliary input of linguistic context and entity information has any effect on the KGQA performance?*
2. *How the top-performing ablations compare to other systems?*
3. *How the optimal ablations perform in a multilingual setting?*

4.1. Training and Evaluation Datasets

To train the MST5 model, we utilize the LC-QuAD 2.0, QALD-9-Plus, and QALD-10 datasets. Below, we outline these datasets and detail any modifications applied:

[7] https://huggingface.co/docs/transformers/model_doc/mt5

LC-QuAD 2.0: Introduced by Dubey et al. [27], is the largest English-only KGQA dataset. It consists of over 30,000 natural language questions paired with their corresponding Wikidata SPARQL queries. We choose this dataset primarily to train the model on contextualizing SPARQL queries alongside natural language queries before proceeding to train on a smaller, multilingual dataset.

QALD-9-Plus: Introduced by Perevalov et al. [28], QALD-9-Plus dataset includes the questions in English, German, Russian, French, Spanish, Armenian, Belarusian, Lithuanian, Bashkir and Ukrainian. The training set for Wikidata consists of 371 questions, while the test set has 136 questions. We update the QALD-9-Plus dataset by adding Chinese and Japanese translations of each question. These translations were done by a native speaker of these languages. Since the test set contains the answers from an older (unavailable) version of Wikidata, we update the QALD-9-Plus dataset using endpoint provided for QALD-10[8] as both of these datasets were created concurrently. Subsequently, we filter out all the questions with empty answer-set. The refined test dataset now comprises 102 questions. We refer to this version as QALD-9-Plus (updated) test dataset in the upcoming sections.

QALD-10: Introduced by Usbeck et al. [2], it comprises of questions in English, German, Chinese and Russian. It uses QALD-9-Plus as its training data, and provides a test dataset of 394 questions. We update the QALD-10 dataset by adding Japanese translation for each question. These translations were done by a native speaker of the respective language.

To benchmark our models, our approach requires datasets that are multilingual and provide a reference SPARQL query alongside the retrieved answer-set for the respective natural language question. These criteria are essential to show that our models can generate SPARQL queries that are not only similar to the reference but also possess the capability to fetch the accurate answer. To fulfil these requirements, we make use of QALD-9-Plus (updated) test and QALD-10 dataset for evaluation. A key distinction among these QALD datasets lies in the evaluation metric employed for comparison. The QALD-9-Plus dataset utilizes `Macro F1` for assessing the performance across various KGQA systems, whereas the QALD10 dataset suggests `Macro F1 QALD`[9] as per community's request.[10] To conduct the evaluation and obtain performance metrics on these datasets, we make use of the GERBIL-QA [29] tool.

4.2. Model Training Details

We selected the pretrained mT5[11] as the foundational model for our work. Its pretraining involved the mC4[12] corpus, encompassing 101 diverse languages. During the fine-tuning phase, we set the maximum number of epochs to 300 while setting up an early-stop regularization

[8] https://github.com/KGQA/QALD-10#endpoint
[9] https://github.com/dice-group/gerbil/issues/320
[10] https://github.com/dice-group/gerbil/issues/211
[11] https://huggingface.co/google/mt5-xl
[12] https://tensorflow.org/datasets/catalog/c4#c4multilingual

method to prevent overfitting. Also, we use the DeepSpeed[13] tool introduced by Rajbhandari et al. [30] to optimize the computing resource usage. As for the hardware, we make use of the Nvidia-A100[14] GPU for training our models.

4.3. Ablation Study

To assess the impact of the proposed input augmentations on the end-to-end KGQA performance of our approach, we conduct an ablation study. Our study involves training and testing every potential variant that includes the following modifications:

- Fine-tuning on LC-QuAD 2.0 dataset
- Fine-tuning on QALD-9-Plus dataset
- Adding linguistic context to the input
- Incorporating entity tags into the input

We evaluate these ablations[15] on the English questions from QALD-9-Plus (updated) test dataset to compare their performance. We exclude the multilingual questions due to resource limitations, as for each additional language all the ablations would need to be evaluated. We also exclude QALD-10 dataset from the ablation study because its training data is a combination of the QALD-9-Plus training and testing data.

4.4. Optimal Models Evaluation

From the ablation study, we select the model(s) demonstrating the best performance and subject them to further evaluation, during this evaluation we cover all the supported languages. For QALD-9-Plus (updated) test dataset, we compare the performance of our optimal models with only one other KGQA system: DeepPavlov-2023 [24]. We were not able to include any other relevant system [11, 10, 9] as we couldn't extract the generated SPARQL queries for updated QALD-9-Plus dataset.[16] For QALD-10, we refine our optimal model(s) by additional training on a merged dataset of QALD-9-Plus train and test data. The further fine-tuning allowed us to perform a direct comparison with the results of the other existing models in a fair manner. We made use of the QA-System-Wrapper[17] tool to query the DeepPavlov-2023 system.

5. Results and Discussion

Our discussion of the results is structured into three sections: Initially, in subsection 5.1, we focus on the performance metrics for English. Then, in subsection 5.2, we examine the performance of our best ablations across various languages. Finally, in subsection 5.3, we discuss the challenges encountered during our evaluation with external systems and the measures we implemented to prevent our system from contributing to the same problem.

[13] https://deepspeed.ai/
[14] https://nvidia.com/en-us/data-center/a100/
[15] See appendix table 4
[16] Either unreachable public APIs, technical issues in the local deployment or no SPARQL query generation.
[17] https://github.com/WSE-research/qa-systems-wrapper

Table 1
QALD-10 macro performance comparison (English).

Approach	F1	Precision	Recall	F1 QALD
Borroto et al.	0.4538	0.4538	0.4574	0.5947
Diefenbach et al.	0.5070	0.5068	0.5238	0.5776
Shivashankar et al.	0.3215	0.3206	0.3312	0.4909
Baramiia et al.	0.4277	0.4289	0.4272	0.4281
DeepPavlov-2023 [24]	0.3241	0.3279	0.3369	0.4518
MST5_{ENT}	0.4784	0.4777	0.4848	0.6330
$\text{MST5}_{\text{LING+ENT}}$	**0.5271**	**0.5271**	**0.5317**	**0.6727**

5.1. English-based Performance

As a first step, we perform the ablation study to find out *whether the simplified auxilliary input of linguistic context and entity information has any effect on the KGQA performance.* We evaluate all our model ablations on the QALD-9-Plus (updated) English dataset.[18] Based on the results (Macro F1), we observe the following:

- Linguistic context and entities always improve the performance of the system
- In most cases, combination of linguistic context and entities leads to the best performing model
- Between the models fine-tuned on both datasets, the entity-only variant has slightly better performance

Now, to investigate *how the top-performing ablations compare to other systems*, we pick the two best performing models $\texttt{MST5}_{\text{L2+Q9+ENT}}$, $\texttt{MST5}_{\text{L2+Q9+LING+ENT}}$ and simplify their names to $\texttt{MST5}_{\text{ENT}}$, $\texttt{MST5}_{\text{LING+ENT}}$ respectively. Table 1 presents a comparative analysis of various models tested on the QALD-10 dataset. In this comparison, we find that:

- MST5 significantly outperforms the competing systems
- Model variant incorporating both linguistic context and entity information emerges as the top performer

5.2. Multilingual Performance

We proceed by examining *how the optimal ablations perform in a multilingual setting.* Table 2 provides direct comparison of our models $\texttt{MST5}_{\text{ENT}}$ and $\texttt{MST5}_{\text{LING+ENT}}$ with DeepPavlov-2023 [24] on all supported languages[19] or QALD-9-Plus (updated). The results show the following:

- MST5 variants perform better than DeepPavlov-2023 for both languages supported by it (English, Russian)

[18] See appendix table 5 for direct comparison of the model ablations.

[19] The language codes used in the datasets and our experiments are as per the ISO 639-1 standard https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

Table 2
Performance (Macro F1) comparison of MST5 with DeepPavlov-2023 on the QALD-9-Plus (updated) test dataset on all supported languages

	DeepPavlov-2023 [24]	$MST5_{ENT}$	$MST5_{LING+ENT}$
ba	-	**0.1842**	0.1579
be	-	**0.2907**	0.2807
de	-	**0.4126**	0.3851
en	0.3716	**0.4187**	0.4015
es	-	**0.3600**	0.3498
fr	-	**0.4167**	**0.4167**
ja	-	0.0654	**0.0752**
lt	-	**0.3115**	0.2792
ru	0.3117	**0.3761**	0.3467
uk	-	**0.3467**	0.3369
zh	-	**0.3344**	0.3115

- When comparing the MST5 variants to each other:
  - Performance in German and French is quite comparable to English
  - Performance in Spanish, Russian, Lithuanian, Ukrainian, Belarusian, and Chinese is a bit lower than English
  - For Bashkir and Japanese the performance is very low in comparison

For Bashkir, the low performance is due to bad quality of the entity recognition and linguistic context extraction through the third-party tools. For Japanese, the lower F1 score is because of the small size of QALD-9-Plus (updated) test set, as many resultant SPARQL queries for Japanese did not work.

Finally, in the Table 3, we have evaluation results for multilingual setting on QALD-10. We observe that our models not only outperform the DeepPavlov-2023, but also achieve comparable results on all supported languages. The performance on Japanese is lower in comparison to other languages, but still significantly better than the QALD-9-Plus (updated) test dataset. We attribute this improvement in performance to the larger size (≈4x) of the QALD-10 test dataset.

5.3. Discussion

Beyond the overall scarcity of multilingual SPARQL-generation systems, our evaluation tables reveal gaps in the reported results of other KGQA approaches. We mainly face this issue with our QALD-9-Plus (updated) dataset. For the QALD-10 dataset, the issue is with the lack of performance data on non-English languages.

This shortfall stems from several contributing factors:

1. KGQA systems [4, 9] not providing a SPARQL, making it hard to run on a custom dataset and SPARQL endpoint;
2. Unresponsive endpoints [11, 10] or source code that doesn't work;

Table 3
QALD-10 performance (Macro F1 QALD) comparison for all supported languages.

	DeepPavlov-2023 [24]	$MST5_{ENT}$	$MST5_{LING+ENT}$
de	-	0.5908	**0.6048**
en	0.5092	0.6330	**0.6727**
ja	-	0.4224	**0.4964**
ru	0.4118	0.6296	**0.6600**
zh	-	0.5877	**0.6398**

3. No maintained knowledge-base endpoint for the evaluation dataset [28];
4. Absence of multilingual performance numbers for the evaluated systems [2].

Due to these factors, reproducing or generating new results on existing multilingual systems becomes impossible. To mitigate this, it is beneficial to have open-source and well-documented systems, complemented by datasets that offer a maintained SPARQL endpoint or the relevant knowledge-base data dumps.

In our approach, we offer an open-source and thoroughly documented codebase. Additionally, our experimental framework, as detailed in Section 4, enables the replication and straightforward comparison of results, serving as a valuable reference for future research work.

6. Limitations

One major limitation of MST5 is its dependence on third-party tools for entity recognition, disambiguation, and linguistic-context extraction, which vary across languages. This hampers scalability when new languages are added and yields poor performance on low-resource languages such as Armenian and Bashkir, obstructing truly multilingual support. To overcome this, we will adopt a multi-task framework that employs a single LM to jointly extract context and perform entity recognition + disambiguation, eliminating external tools and improving generalization to low-resource languages. Future work will enrich the auxiliary input with entity types and relations and explore NLP techniques like Semantic Role Labeling [33] to further boost SPARQL query quality.

7. Conclusion

In this paper we introduce a streamlined multilingual KGQA approach built on the pretrained multilingual language model mT5 [17]. By feeding auxiliary linguistic and entity information directly into the model, we let the LM learn the necessary representations autonomously, eliminating the need for separate encoders. An extensive ablation study shows that this auxiliary input markedly boosts end-to-end performance on the latest QALD benchmarks, and our comparisons with existing KGQA systems highlight its superior results. We also emphasize the lack of multilingual KGQA systems and the paucity of cross-lingual evaluation data, which hampers comprehensive benchmarking. To mitigate these issues we discuss common pitfalls and propose a multi-task extension as a foundation for future research.

Acknowledgements

This work has been supported by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) through the project SAIL (grant no. NW21-059D). It has been supported by the German Federal Ministry of Education and Research (BMBF) through funding for the EuroStars project E! 114154 PORQUE (grant no. 01QE2056C) and the project KI-OWL (grant no 01IS24057B). It has also received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070305.

References

[1] D. Diefenbach, V. Lopez, K. Singh, P. Maret, Core techniques of question answering systems over knowledge bases: a survey, Knowledge and Information Systems 55 (2018) 529–569. URL: https://doi.org/10.1007/s10115-017-1100-y. doi:10.1007/s10115-017-1100-y.

[2] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. N. Ngomo, M. Saleem, A. Both, QALD-10 — The 10th Challenge on Question Answering over Linked Data, Under review in the Semantic Web Journal (2023). URL: https://www.semantic-web-journal.net/system/files/swj3357.pdf.

[3] A. Both, D. Diefenbach, K. Singh, S. Shekarpour, D. Cherix, C. Lange, Qanary – a methodology for vocabulary-driven open question answering systems, 2016. doi:10.1007/978-3-319-34129-3_38.

[4] S. Hu, L. Zou, J. X. Yu, H. Wang, D. Zhao, Answering natural language questions by subgraph matching over knowledge graphs, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 824–837. URL: https://api.semanticscholar.org/CorpusID:4569766.

[5] T. Soru, E. Marx, A. Valdestilhas, D. Esteves, D. Moussallem, G. Publio, Neural machine translation for query construction and composition, 2018. URL: https://arxiv.org/abs/1806.10478.

[6] M. A. Borroto, F. Ricca, B. Cuteri, A system for translating natural language questions into sparql queries with neural networks: Preliminary results $discussionpaper$, in: SEBD 2021: Italian Symposium on Advanced Database Systems, RWTH Aachen, Aachen, Germany, 2021, pp. 226–234. URL: https://www.tib.eu/de/suchen/id/TIBKAT%3A181641011X.

[7] D. Vollmers, R. Jalota, D. Moussallem, H. Topiwala, A.-C. N. Ngomo, R. Usbeck, Knowledge graph question answering using graph-pattern isomorphism, in: Studies on the Semantic Web, IOS Press, 2021. URL: https://doi.org/10.3233%2Fssw210038. doi:10.3233/ssw210038.

[8] M. R. A. H. Rony, U. Kumar, R. Teucher, L. Kovriguina, J. Lehmann, Sgpt: A generative approach for sparql query generation from natural language questions, IEEE Access 10 (2022) 70712–70723. doi:10.1109/ACCESS.2022.3188714.

[9] M. Burtsev, A. Seliverstov, R. Airapetyan, M. Arkhipov, D. Baymurzina, N. Bushkov, O. Gureenkova, T. Khakhulin, Y. Kuratov, D. Kuznetsov, A. Litinsky, V. Logacheva, A. Lymar, V. Malykh, M. Petrov, V. Polulyakh, L. Pugachev, A. Sorokin, M. Vikhreva, M. Zaynutdinov, DeepPavlov: Open-source library for dialogue systems, in: Proceedings of ACL 2018, System Demonstrations, Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 122–127. URL: https://aclanthology.org/P18-4021. doi:10.18653/v1/P18-4021.

[10] T. Pellissier Tanon, M. D. de Assunção, E. Caron, F. M. Suchanek, Demoing platypus – a multilingual question answering platform for wikidata, in: A. Gangemi, A. L. Gentile, A. G. Nuzzolese, S. Rudolph, M. Maleshkova, H. Paulheim, J. Z. Pan, M. Alam (Eds.), The Semantic Web: ESWC 2018 Satellite Events, Springer International Publishing, Cham, 2018, pp. 111–116.

[11] D. Diefenbach, A. Both, K. Singh, P. Maret, Towards a question answering system over the semantic web, Semantic Web 11 (2020) 421–439. doi:10.3233/SW-190343.

[12] A. Perevalov, A. Both, D. Diefenbach, A.-C. Ngonga Ngomo, Can machine translation be a reasonable alternative for multilingual question answering systems over knowledge

graphs?, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 977–986.
[13] N. Srivastava, A. Perevalov, D. Kuchelev, D. Moussallem, A.-C. Ngonga Ngomo, A. Both, Lingua franca – entity-aware machine translation approach for question answering over knowledge graphs, in: Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 122–130. URL: https://doi.org/10.1145/3587259.3627567. doi:10.1145/3587259.3627567.
[14] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities, ScientificAmerican.com (2001).
[15] M. Kejriwal, Knowledge graphs: A practical review of the research landscape, Information 13 (2022). URL: https://www.mdpi.com/2078-2489/13/4/161. doi:10.3390/info13040161.
[16] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, 2020. arXiv:1906.08237.
[17] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, C. Raffel, mt5: A massively multilingual pre-trained text-to-text transformer, CoRR abs/2010.11934 (2020). URL: https://arxiv.org/abs/2010.11934. arXiv:2010.11934.
[18] S. Black, L. Gao, P. Wang, C. Leahy, S. Biderman, GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021. URL: https://doi.org/10.5281/zenodo.5297715. doi:10.5281/zenodo.5297715, If you use this software, please cite it using these metadata.
[19] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, et collab, Bloom: A 176b-parameter open-access multilingual language model, 2023. arXiv:2211.05100.
[20] P. Wu, X. Zhang, Z. Feng, A survey of question answering over knowledge base, in: Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding, Springer Singapore, Singapore, 2019, pp. 86–97.
[21] M. Schuster, K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing 45 (1997) 2673–2681. doi:10.1109/78.650093.
[22] Z. Huang, W. Xu, K. Yu, Bidirectional lstm-crf models for sequence tagging, 2015. arXiv:1508.01991.
[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.
[24] D. Zharikova, D. Kornev, F. Ignatov, M. Talimanchuk, D. Evseev, K. Petukhova, V. Smilga, D. Karpov, Y. Shishkina, D. Kosenko, M. Burtsev, DeepPavlov dream: Platform for building generative AI assistants, in: D. Bollegala, R. Huang, A. Ritter (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 599–607. URL: https://aclanthology.org/2023.acl-demo.58. doi:10.18653/v1/2023.acl-demo.58.
[25] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, C. Biemann, Modern baselines for sparql semantic parsing, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, ACM, 2022. URL: http://dx.doi.org/10.1145/3477495.3531841. doi:10.1145/3477495.3531841.

[26] T. Kudo, J. Richardson, SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, in: E. Blanco, W. Lu (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 66–71. URL: https://aclanthology.org/D18-2012. doi:10.18653/v1/D18-2012.

[27] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 69–78.

[28] A. Perevalov, D. Diefenbach, R. Usbeck, A. Both, Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers, in: 2022 IEEE 16th International Conference on Semantic Computing (ICSC), IEEE, 2022, pp. 229–234.

[29] R. Usbeck, M. Röder, M. Hoffmann, F. Conrad, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler, C. Unger, Benchmarking Question Answering Systems, Semantic Web 10 (2019) 293–304. URL: http://www.semantic-web-journal.net/system/files/swj1578.pdf. doi:10.3233/SW-180312.

[30] S. Rajbhandari, J. Rasley, O. Ruwase, Y. He, Zero: Memory optimizations toward training trillion parameter models, 2020. arXiv:1910.02054.

[31] K. Shivashankar, K. Benmaarouf, N. Steinmetz, From graph to graph: Amr to sparql, in: Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022), 2022.

[32] N. Baramiia, A. Rogulina, S. Petrakov, V. Kornilov, A. Razzhigaev, Ranking approach to monolingual question answering over knowledge graphs, in: Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022), 2022.

[33] D. Jurafsky, J. H. Martin, Speech and language processing (3rd edition draft), https://web.stanford.edu/~jurafsky/slp3/, 2024. Accessed on: 15.02.2024.

Acronyms

KG Knowledge Graph. 2, 3, 4

KGQA Knowledge Graph Question Answering. 1, 2, 3, 4, 8, 9, 11, 12

LM Language Model. 3, 6, 12

MT Machine Translation. 4

NEAMT Named Entity Aware Machine Translation. 5

NED Named Entity Disambiguation. 5

NER Named Entity Recognition. 3, 5

NLP Natural Language Processing. 12

NMT Neural Machine Translation. 3

NsPM Neural SPARQL Machines. 3

SPARQL SPARQL. 2, 3, 4, 5, 6, 8, 9, 11, 12

A. General Appendix

Table 4
All covered MST5 ablations based on the auxiliary features (linguistic context and entity information) and fine-tuning dataset used during the model training.

Name	Ling. Ctxt.	Ent. Info.	LC-QuAD 2.0	QALD-9-Plus
-	No	No	No	No
$MST5_{Q9}$	No	No	No	Yes
$MST5_{L2}$	No	No	Yes	No
$MST5_{L2+Q9}$	No	No	Yes	Yes
-	No	Yes	No	No
$MST5_{Q9+ENT}$	No	Yes	No	Yes
$MST5_{L2+ENT}$	No	Yes	Yes	No
$MST5_{L2+Q9+ENT}$	No	Yes	Yes	Yes
-	Yes	No	No	No
$MST5_{Q9+LING}$	Yes	No	No	Yes
$MST5_{L2+LING}$	Yes	No	Yes	No
$MST5_{L2+Q9+LING}$	Yes	No	Yes	Yes
-	Yes	Yes	No	No
$MST5_{Q9+LING+ENT}$	Yes	Yes	No	Yes
$MST5_{L2+LING+ENT}$	Yes	Yes	Yes	No
$MST5_{L2+Q9+LING+ENT}$	Yes	Yes	Yes	Yes

Table 5
Macro performance metrics of MST5 ablations on the QALD-9-Plus (updated) test dataset (English): divided into three groups based on fine-tuning data, separated by horizontal lines

Ablation name	F1	Precision	Recall	F1 QALD
$MST5_{L2+ENT}$	0.1886	0.2001	0.1982	0.322
$MST5_{L2+LING}$	0.2291	0.2355	0.2353	0.367
$MST5_{L2+LING+ENT}$	0.2622	0.2747	0.2698	0.4088
$MST5_{L2}$	0.1353	0.134	0.1373	0.2359
$MST5_{Q9+ENT}$	0.3173	0.3318	0.3164	0.4554
$MST5_{Q9+LING}$	0.206	0.2188	0.2043	0.2759
$MST5_{Q9+LING+ENT}$	0.3203	0.3424	0.3216	0.4624
$MST5_{Q9}$	0.0098	0.0098	0.0098	0.0194
$MST5_{L2+Q9+ENT}$	**0.4187**	**0.4366**	**0.4242**	**0.572**
$MST5_{L2+Q9+LING}$	0.3407	0.3571	0.3493	0.4745
$MST5_{L2+Q9+LING+ENT}$	0.4015	0.4192	0.4046	0.5563
$MST5_{L2+Q9}$	0.2628	0.2792	0.2631	0.3663