# KBQA Final Report

Mengshi Ma

DICE group at University Paderborn

## 1 Task Definition

In the "Knowledge-based Question Answering"project group, we aim to develop a question answering system, which receives a natural language question from a user, then responses with the answer of this question based on the knowledge from DBpedia.

In order to answer a question, the first step is to convert the question to a SPARQL query, with which the knowledge in DBpedia can be queried as a relational database. Once we get a response from the DBpedia endpoint, the answer is extracted and presented to the user.

## 2 Approach Description

Our basis approach is the Neural SPARQL Machine (NSpM)[4], which considers SPARQL query as another natural language and employ machine translation techniques. NSpM consists of three main components: Generator, Learner and Interpreter. Generator generates a training set by fulling entities from knowledge graph into question and query templates with placeholders.

Following is an example of template and generated questions and queries:

```
Where is <A> located in?;
                SELECT ?x { <A> dbo:location ?x }

Where is London located in?;
                SELECT ?x { dbr:London dbo:location ?x }
Where is the Colosseum located in?;
                SELECT ?x { :Colosseum :location ?x }
Where is Mount Everest located in?;
                SELECT ?x { :Mount_Everest :location ?x }
```

Learner is a deep neural network based translator, which translates a sequence of tokens in natural language into a sequence which encodes a SPARQL query after training. Interpreter uses the learned model to predict a SPARQL query for received question.

There are two drawbacks in this original NSpM. Since training data is generated using templates, the number of entities are severely restricted, which leads to a bad performance in evaluation. Or the training set has to be very large that makes it hard to train with. Besides, there is no template for QALD 8 and

9 training set, which make comparisons difficult. On the other hand, learner consists only of a one-layer encoder and a one-layer decoder. The performance will be greatly improved using other translation model. To improve NSpM, we used DBpedia Spotlight and Pegasus model.

DBpedia Spotlight [2] is a tool for automatic annotating and entity linking, which performs entity extraction including entity detection and name resolution. First, we integrated DBpedia Spotlight into the interpreter to detect entities. We abandoned generator and trained with templates directly. Then in the prediction phase, when interpreter receives a question, it detects entities in the question using DBpedia Spotlight and replaces them with placeholders. Interpreter translates this modified question into a SPARQL query with placeholders and restores entities again. Integration of DBpedia Spotlight in interpreter decreases the size of training set and therefore the training effort is reduced. However, the performance is better, since it uses the entire knowledge graph while prediction and learner only needs to focus on learning predicates in SPARQL query. DBpedia Spotlight is also used for generate templates from QALD 8 and 9 training set, finding entities in quesiton and query and replacing them with placeholders.

Pegasus[6] is a pre-trained model with extracted gap-sentences for abstractive summarization. We used hugging face pegasus model, which is pre-trained on a large text corpora. Pegasus model enlarges our set of tokens and improved the quality of translation. Moreover, we pre-trained again on LC-QALD dataset and fine-tuned on QALD 8 and 9.

We also tried Convolutional Sequence-to-Sequence model (ConvS2S) [3] to improve the translation quality. ConvS2S converts an input sequence to a output sequence using convolution neural networks. This approach did not work in the end as we could not rewrite it in tensorflow 2.

For evaluation we used gerbil[1]. Gerbil is a web-based platform for comparison of QA system. User can upload test set and add a QA system via URI or upload a JSON file with answers in QALD-JSON format. We focused on QALD 8 and 9 for evaluation. At the beginning, we generated a JSON file with answers and uploaded it to gerbil. Later once we deployed our QA system on VM, we added our system via URI every time.

## 3 Evaluation

We used gerbil for evaluation. Gerbil can evaluate QA systems on a QALD dataset automatically, computing micro precision, recall and F1 score, macro precision, recall and F1 score, and F1 QALD score. Additionally, the average answering time is also calculated.

We focused on QALD 8 and QALD 9 datasets, trained and evaluated on them. The questions in QALD 9 are more complicated than in QALD 8, e.g. multiple triples and more logic. Also, QALD 9 dataset contains more questions. Therefore, QALD 9 is more challenging than QALD 8 for evaluation.

Following table shows our evaluation results from approach A during the Project Group:

| QALD-8 | | | | | |
|---|---|---|---|---|---|
| **Date** | **Model** | **Precision** | **Recall** | **F1** | **F1 QALD** |
| 06.12 | NSpM | 0 | 0 | 0 | 0 |
| 20.12 | NSpM | 0.0244 | 0.0244 | 0.0244 | 0.0476 |
| 23.01 | NSpM_SL | 0.1707 | 0.1626 | 0.1602 | 0.2777 |
| 14.06 | NSpM_PSL | 0.2561 | 0.2683 | 0.2602 | 0.4149 |
| 27.06 | NSpM_LCPSL | 0.3171 | 0.3415 | **0.3252** | **0.5025** |
| | Tebaqa | 0.4756 | 0.4878 | 0.4797 | 0.556 |

| QALD-9 | | | | | |
|---|---|---|---|---|---|
| **Date** | **Model** | **Precision** | **Recall** | **F1** | **F1 QALD** |
| 23.01 | NSpM_SL | 0.1299 | 0.1344 | 0.1312 | 0.2362 |
| 14.06 | NSpM_PSL | 0.2479 | 0.2694 | **0.2454** | **0.4127** |
| 27.06 | NSpM_LCPSL | 0.2283 | 0.2464 | 0.2237 | 0.3845 |
| | Tebaqa | 0.2413 | 0.2452 | 0.2384 | 0.3741 |

All scores are macro scores.

At the beginning of the Project group, in order to try out the original NSpM model and gerbil evaluation, we trained with QALD 8 train dataset for 8 epochs. We also implemented a python script to simulate receiving natural language question, converting it to query, sending request to DBpedia endpoint, and finally generating a dataset with answers in QALD format, since we did not have an URL endpoint for evaluation until that time.

After figuring out the functions of each component, we trained NSpM again for more epochs. This time, the evaluation result was not zero anymore, which proved that, the NSpM is a fesible approach.

By inspecting the SPARQL queries in train and test dataset, we noticed that many entities appear in test dataset are not in train dataset. Therefore, we integrated DBpedia spotlight to avoid this problem. With DBpedia spotlight and more training data, the result has been greatly improved.

For a better conversion from natural language question to SPARQL query, we used hugging face Pegasus model instead of the simple encoder and decoder neural network in the original NSpM.

At the end of the project group, we tried pre-training on LC-QALD dataset, then fine-tuning on QALD 8 or QALD 9. Interestingly, the score for QALD 8 increases with pre-training, but for QALD 9 decreases. However, we did not enough time to figure out the reason behind it.

Compare to TeBaQa [5], our approach overperforms TeBaQa on QALD 9 and performs close to TeBaQa on QALD 8.

## 4 Learned Skills

During this two semester project group, I have learned a lot in all aspects.

We used Scrum to manage our team, therefore, I have gathered some experience with Scrum, e.g. How to be a Scrum master, how to manage weekly tasks with a board, and what to do in meetings. Also, the most important thing is how to work as a team, since a usually worked alone.

I have also improved my coding style in this project. Thanks to the linter, I wrote more readable code and never forgot to add comments on each function, which make it easier for others and me later to understand.

I have also formed good habits in git, committing frequently and with meaningful comments. I practiced git functions such as new branch, merge, rebase, and also how to handle merge conflicts. Additionally, I have learned how to set up CI/CD in Github from my teammates, which simplified our deployment.

This is my first time working with network on a virtual machine with nginx, REST and docker. I think this experience can be applied on many other projects.

I got my first insight into knowledge graph in this project group and knew how SPARQL query is written.

The last but not least, I learned much about natural language processing in praxis, including encoder and decoder, tokenizer, model training. Moreover, I had some experience in building our own dataset for a NLP model.

## 5 Issues

During this project group, I have met many issues, but with the help of our teammates, they were solved quickly.

At first, I did not know what to do with our virtual machine. I asked my teammates then figured out how to deploy our software on it.

In our evaluation at beginning, we must create our dataset in QALD format with answer and upload it to gerbil, since the system was not deployed on VM at that time. We created a script to do it automatically.

Inspired by templates and generator in the original NSpM model, we used templates, questions and queries with placeholders, for training directly. For training on QALD 8 and 9, we did not have a dataset suitable for our approach. Thus, we implemented a script to replace entities in QALD 8 and 9 with placeholders and convert to the format we need. In order to include more predicates in our dataset, we also checked classes in DBpedia and wrote questions and queries on our own.

While training our model on felis Server, I encountered some issues with incompatible cuda and driver version. This was fixed quickly after some research.

At the beginning of the second semester, we found Convolutional Sequence-to-Sequence model (ConvS2S), which is written in tensorflow version 1. There is a big difference between Tensorflow version 1 and 2, we have spent a lot of time to rewrite it in tensorflow two and add additional features that we want. However, after the implementation we trained ConvS2S with our questions and queries, it could not predict any query. We tried a lot to fix this model, but it did not work in the end and we switch to Pegasus model.

At the end of project group, we wanted to pre-train on LC-QALD and fine-tune on QALD 8 and 9, which could not be done directly. After checking parameters in Pegasus model, we found a way to do it.

## 6 Self-evaluation

In this two semester I think I did a good job. I am glad to cooperate and help other teammates. If others have some problem, I am willing to help and find the solution together. I also came up with some ideas for improving our approach, we tried several of them, the results are good and bad. Sometimes we went to the wrong direction, for example with NSpM, but in the end the result of our approach is good. Due to lack of practical experience with buiding neural network and training on server, the speed of development was slowed down. Therefore, I would give myself a score of 1.7.

## References

1. Cornolti, M., Ferragina, P., Ciaramita, M.: A framework for benchmarking entity-annotation systems. In: Proceedings of the 22nd International Conference on World Wide Web, p. 249–260, WWW '13, Association for Computing Machinery, New York, NY, USA (2013), ISBN 9781450320351, `https://doi.org/10.1145/2488388.2488411`, URL `https://doi.org/10.1145/2488388.2488411`
2. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics) (2013)
3. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. CoRR **abs/1705.03122** (2017), URL `http://arxiv.org/abs/1705.03122`
4. Soru, T., Marx, E., Valdestilhas, A., Esteves, D., Moussallem, D., Publio, G.: Neural machine translation for query construction and composition (2018), URL `https://arxiv.org/abs/1806.10478`
5. Vollmers, D., Jalota, R., Moussallem, D., Topiwala, H., Ngomo, A.N., Usbeck, R.: Knowledge graph question answering using graph-pattern isomorphism. CoRR **abs/2103.06752** (2021), URL `https://arxiv.org/abs/2103.06752`
6. Zhang, J., Zhao, Y., Saleh, M., Liu, P.J.: Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In: Proceedings of the 37th International Conference on Machine Learning, ICML'20, JMLR.org (2020)