Ranking Approach to Monolingual Question Answering over Knowledge Graphs

Nikita Baramiia[1], Alina Rogulina[1], Sergey Petrakov[1], Valerii Kornilov[1], and Anton Razzhigaev[1]

Skolkovo Institute of Science and Technology (Skoltech)
Bolshoy Boulevard 30, bld. 1, 121205, Moscow, Russia
{Nikita.Baramiia, Alina.Rogulina, Sergey.Petrakov, Valerii.Kornilov, Anton.Razzhigaev}@skoltech.ru

Abstract. In this paper we describe our solution to the task 1 of Question Answering over Linked Data (QALD) challenge: multilingual QALD over Wikidata. We propose the method where we learn to rank items and properties to find suitable SPARQL query. With our approach we achieve 0.4281 Macro F1-score in QALD system.

Keywords: Question answering · transformers · approximate nearest neighbors

GitHub: https://github.com/roguLINA/NNLP_project

1 Introduction

Question Answering (QA) is one of rapidly developed fields in natural language processing (NLP), covering many different problems from search engines to dialogue systems. One of the most common tasks is to answer a question by making a query to an RDF data repository (an RDF dataset is the unit that is queried by a SPARQL query). In our case, we should train the model to make right SPARQL queries to retrieve answers from Wikidata.

2 Data description

In this challenge Wikidata was chosen as the main RDF dataset for answers search. Our preprocessing procedure of the training data consists of the parsing queries which have the form shown in the example 1. As a result, our train data is reduced from 412 to 145 samples with items (Q) and properties (P).

We also have embeddings for items (4 106 847) and properties (5 927) from Wikidata[1]. Then SPARQL queries are generated from this pool.

We use only one language (English) for all samples, both train and test. The choice of this approach is justified by the fact that each question is represented by its own set of languages necessarily included English.

[1] These embeddings were prepared via TransE algorithm from PyTorch-BigGraph (PGD) system from Facebook

```
{
    "id": "99",
    "question": [...],
    "query": {
        "sparql": "SELECT DISTINCT
        ?o1 WHERE {
        <http://www.wikidata.org/
        entity/Q23337>
        <http://www.wikidata.org/
        prop/direct/P421>  ?o1 .  }"
    },
    "answers": [...]
}
```

Listing 1: QALD_JSON format

3 Description of the approach

Our approach consists of several parts which we describe step-by-step in the Sections 3.1, 3.2, 3.3. Then we discuss training process of the proposed model and its usage on inference step.

3.1 Learning to rank

Our solution is closely connected with a task of ranking: in traditional statement we want our model to give a score for each observation according to which we can sort them from the most relevant to the least. Our core idea is that our model predicts embeddings of Q and P which are as close as possible to the relevant ones (connected with correct query) and as far as possible to the others. We use triplet margin loss and transformer language model BART for this purpose.

3.2 BART model

We use pre-trained BART [3] model from Hugging Face Hub[2]. Basically, BART is a transformer sequence-to-sequence model with a bidirectional encoder and an autoregressive decoder. This model performs well in such tasks as summarizing, translation, classification, and what is especially important for us, a task of answering a question (after fine-tuning). For this reason, BART is the basis for our research.

The authors of [5] demonstrate that BART and RoBERTa [4] are the best models according to F1-score at the task of extractive question answering. Since our target metric is F1-score it is additional argument to work with these models.

[2] `https://huggingface.co/facebook/bart-base`

We compare the performance of RoBERTa and BART and we do not receive any improvements from RoBERTa, that is why we concentrate on BART.

Moreover, we compare BART model to other models like DistilBERT [6], and even with XLM-RoBERTa [1]. Also, we conducted experiments with multilingual BART, BERT, multilingual BERT, and multilingual XLM-RoBERTa. The comparison of DistilBERT and BART shows that BART has lower loss within all process of training. Theoretically, DistilBERT has faster inference, however, in practice, duration of training did not differ a lot.

We made one modification of the model to adapt it to our task: we average last hidden state embeddings getting output final embedding with size (batch_size, 768), to which we apply a linear map $\mathbf{R}^{768} \to \mathbf{R}^{Q\_embed\_size+P\_embed\_size}$. This final model was fine-tuned to predict embeddings for Q and P. In our approach we use batch_size equals to 128.

3.3 Approximate neighbours search with ScaNN

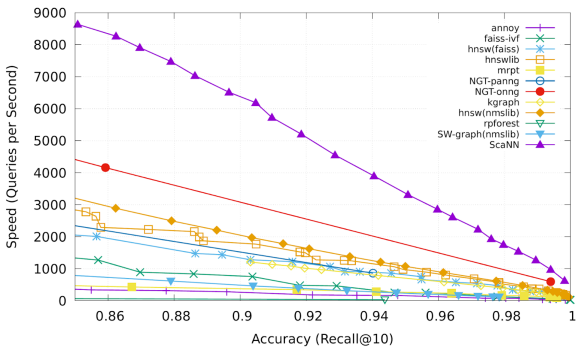

Fig. 1. ScaNN outperforms other methods significantly on glove-100-angular benchmark (Recall@10 – fraction of true nearest neighbors found among 10 returned by algorithm: averaged over all queries)

Scalable Nearest Neighbors (ScaNN) – one of the latest methods for efficient neighbours search achieved by using the new *score-aware quantization loss function* proposed in the paper [2]. It lets the authors to achieve state-of-the-art results in most of benchmarks[3]: on Fig. 1 you can see one of prime examples on the glove-100-angular dataset.

In the Section 3.2 we explain how we get embeddings for P and Q, but with high probability we will never get exact matches, so the idea is to find nearest ones from the prepared pool of Q-items and P-properties via ScaNN approach with the dot product distance metric. In the Section 1 we describe how it works during training and model inference.

[3] `http://ann-benchmarks.com`

3.4 Train and inference procedures

Training process is presented in Algorithm 1 below:

Algorithm 1 Training process

Require: pool_of_Q_and_P, samples, model f_w
  for epoch *in* num_of_epochs **do**
    shuffled_samples $\leftarrow$ $shuffle$(samples)
    batches $\leftarrow$ $split$(shuffled_samples)

    for (sentences_batch, queries_batch) *in* batches **do**
      queries_anchor = f_w(sentences_batch)
      queries_positive = true queries values
      queries_negative = nearest to anchor false queries

      $L = triplet\_loss$(queries_anchor, queries_positive, queries_negative)
      $w \leftarrow w - \nabla_w L$
    end for
  end for=0

It is a typical metric learning procedure with hard negative mining strategy.

Within training procedure we use Adam optimizer. There is a widespread practice of changing learning rate using scheduler since it is worth changing learning rate value during training. The main idea is to decrease it while training because we need smaller steps of the gradient when we come closer to optimum. In our case, we implement StepLR with warm-up during the first two epochs, initial and minimal learning rates 10^{-5}, step size 13, gamma 0.9 (every 13 epochs current learning rate multiplied by 0.9, and it could not be less than 10^{-5}). Initial learning rate equals to minimal learning rate since we use warm-up technique. This mean that first two epochs has minimal learning rate. This is a good practice that allows adaptive optimizers such as Adam to better calculate loss function gradients. Thus, it helps the optimizer to choose a more optimal and stable direction of optimization. Batch size of the final model equals 128 since it is a power of 2 and it is big enough to represent data in one iteration.

We additionally use early stopping as a useful method of regularization, which prevents overfitting of the model. Also, it gives significant reduction of training time in case when there is a large number of training epochs. We set the parameter patience to 6 (if current loss higher than minimal previously achieved loss within 6 epochs then training stops). Thus, we use only 40 epochs out of 100 originally declared.

Furthermore, we use triplet loss with default margin. It is a good idea to take this type of loss because we want to generate a vector that will be close to the vector of correct answers, and far from the vectors of wrong answers. This is exactly what triple loss does.

In the Algorithm 2 we provide steps for getting response with our model:

Algorithm 2 Inference procedure

Require: pool_of_Q_and_P, trained model f_w, test_sentence
0: predicted_query = f_w (test_sentence)
0: 3_nearest_Qs_and_Ps = *find_nearest*(predicted_query)
  for Q *in* 3_nearest_Qs_and_Ps **do**
    for P *in* 3_nearest_Qs_and_Ps **do**
      if exist_query(Q, P) == True **then**
        success = True
        return Q, P
      else
        success = False
    end for
  end for

    if success == False **then**
      return None

We decided to consider more than one nearest neighbour (exactly 3) because not all queries are able to get a response from Wikidata, but probably some combinations of the nearest neighbours can. In our opinion, it is better than not to provide the answer at all.

4 Results and discussion

After iterations of our experiments we received that fine-tuned BART with ScaNN showed the best result. We could reach 0.4281 Macro F1 QALD score. We achieved this score on 3 millions of embeddings. Additionally, we tried 4 millions of embeddings, however, the results were comparable. Thus, we can claim that results are robust to the number of embeddings.

Speaking about extensions to the multilingual case, it is possible with using transformers trained on several languages. Then, the pipeline is the same but we get *num_of_languages* times more data where we have the same answers (queries) for *num_of_languages* samples. Another way is to rotate embeddings: it is a well-known method when we train the rotation matrix to translate embeddings from one language to another. In this case, the main model will be trainable (probably English variant is the most suitable) and models for other languages will be used as is, without fine-tuning. For the last case we can utilize our trained model and we will need only to train rotation matrices: it is a rough but relatively fast way to get baseline extension for multilingual case.

After the deadline of QALD competition submission we conducted one more experiment to understand the quality of current result. We used only 500 000 embeddings of queries and 2 epochs of train. This leads to increase of quality up to 0.5133.

Acknowledgment

We express our deep gratitude to Skoltech NLP Lab headed by Professor Alexander Panchenko for embeddings prepared via TransE algorithm from PyTorch-BigGraph (PGD) system from Facebook. Also, we thank organizers of the competition for the opportunity to work on this interesting problem and conduct this research Professor Ricardo Usbeck, Xi Yan, QALD and NLIWOD team.

References

1. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019)
2. Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., Kumar, S.: Accelerating large-scale inference with anisotropic vector quantization. In: International Conference on Machine Learning (2020), https://arxiv.org/abs/1908.10396
3. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
4. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
5. Pearce, K., Zhan, T., Komanduri, A., Zhan, J.: A comparative study of transformer-based language models on extractive question answering. arXiv preprint arXiv:2110.03142 (2021)
6. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)