Tryout Work for Creating A Better System For Questions Answering

Hao Zou
Department of Computer Science and Engineering
University of Minnesota, Twin Cites
Minneapolis, MN 55414
`zou00080@umn.edu`

1 Introduction

The ideology and architecture of Dense Passage Retrieval (DPR) [1] interested me a lot. As the State-Of-The-Art baseline model, DPR achieves pretty good performances in Questions Answering (QA) domain. There are two main parts of DPR, Retriever and Reader. Retriever is for searching target subset article candidates among all document, while Reader is for reading comprehensioning the articles retrieved and find the correct answer.

As for Open-Domain QA retriever, the main methods are traditional information retriever, such as unigram/bigram TF-IDF matching, BM25 term weighting(ElasticSearch, Lucene). Here the TF-IDF and BM-25 are actually high-dimensional and sparse representations for query and doc. However, they cannot match the context information with the same/similar semantics. Hence, DPR turns to dense encoder for solving this problem. It uses dense representation for query and doc to supplement the sparse representation from TF-IDF/BM-25. It uses dense encoder to output embedding vectors and then does the high-efficient dense vector retriever by using FAISS.

2 What I have Tried

I was mainly working on two stuffs for this tryout project: DPR baseline model and Haystack toolkit which is an awesome end-to-end framework that enables people to build powerful pipelines for QA related tasks. Haystack combines a lot of the best technology from other open-source projects like Huggingface's Transformers, Elasticsearch and DPR.

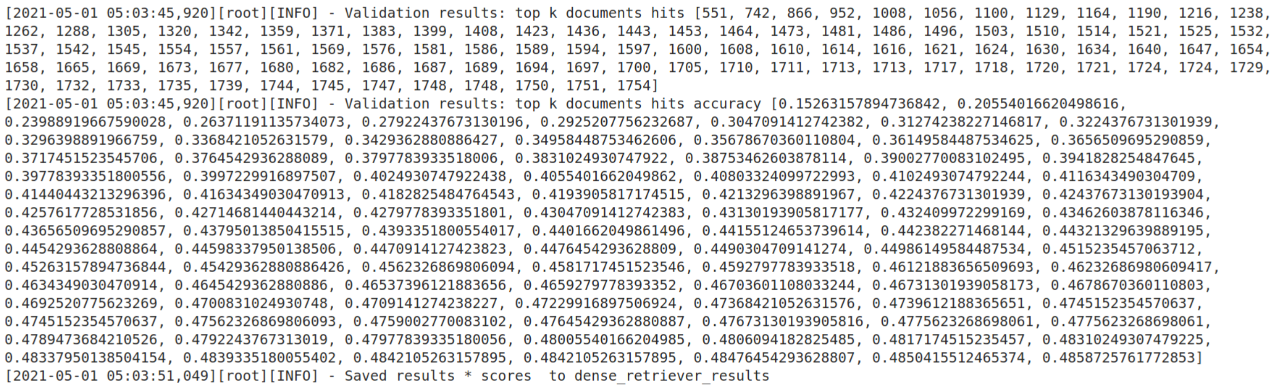
2.1 Computing Resources

- Personal computer device: 8GB, 1GPU
- Google Colab: 12GB, 1GPU
- Minnesota Supercomputing Institute (MSI):
  Tesla K40 32GB 2GPU; Tesla V100 32GB 8GPU

2.2 DPR

- Input Data Format
  The Default data format for DPR Retriever training os JSON. I just used Google's Natural Questions dataset (NQ). SQuAD dataset, however, is not suitable for Retriever training due to format issue, even if it is JSON format as well, I will illustrate that in Haystack part.

Preprint. Under review.



```
[2021-05-01 05:03:45,920][root][INFO] - Validation results: top k documents hits [551, 742, 866, 952, 1008, 1056, 1100, 1129, 1164, 1190, 1216, 1238,
1262, 1288, 1305, 1320, 1342, 1359, 1371, 1383, 1399, 1408, 1423, 1436, 1443, 1453, 1464, 1473, 1481, 1486, 1496, 1503, 1510, 1514, 1521, 1525, 1532,
1537, 1542, 1545, 1554, 1557, 1561, 1569, 1576, 1581, 1586, 1589, 1594, 1597, 1600, 1608, 1610, 1614, 1616, 1621, 1624, 1630, 1634, 1640, 1647, 1654,
1658, 1665, 1669, 1673, 1677, 1680, 1682, 1686, 1687, 1689, 1694, 1697, 1700, 1705, 1710, 1711, 1713, 1713, 1717, 1718, 1720, 1721, 1724, 1724, 1729,
1730, 1732, 1733, 1735, 1739, 1744, 1745, 1747, 1748, 1748, 1750, 1751, 1754]
[2021-05-01 05:03:45,920][root][INFO] - Validation results: top k documents hits accuracy [0.15263157894736842, 0.20554016620498616,
0.23988919667590028, 0.26371191135734073, 0.27922437673130196, 0.2925207756232687, 0.3047091412742382, 0.31274238227146817, 0.3224376731301939,
0.3296398891966759, 0.3368421052631579, 0.3429362880886427, 0.34958448753462606, 0.35678670360110804, 0.36149584487534625, 0.3656509695290859,
0.3717451523545706, 0.3764542936288089, 0.3797783933518006, 0.3831024930747922, 0.38753462603878114, 0.39002770083102495, 0.3941828254847645,
0.39778393351800556, 0.3997229916897507, 0.4024930747922438, 0.4055401662049862, 0.40803324099722993, 0.4102493074792244, 0.4116343490304709,
0.41440443213296396, 0.41634349030470913, 0.4182825484764543, 0.4193905817174515, 0.4213296398891967, 0.4224376731301939, 0.42437673130193904,
0.4257617728531856, 0.42714681440443214, 0.4279778393351801, 0.43047091412742383, 0.43130193905817177, 0.432409972299169, 0.43462603878116346,
0.43656509695290857, 0.43795013850415515, 0.4393351800554017, 0.4401662049861496, 0.44155124653739614, 0.442382271468144, 0.44321329639889195,
0.4454293628808864, 0.44598337950138506, 0.4470914127423823, 0.4476454293628809, 0.4490304709141274, 0.44986149584487534, 0.4515235457063712,
0.45263157894736844, 0.45429362880886426, 0.4562326869806094, 0.4581717451523546, 0.4592797783933518, 0.46121883656509693, 0.46232686980609417,
0.4634349030470914, 0.4645429362880886, 0.46537396121883656, 0.4659279778393352, 0.46703601108033244, 0.46731301939058173, 0.4678670360110803,
0.4692520775623269, 0.4700831024930748, 0.4709141274238227, 0.47229916897506924, 0.47368421052631576, 0.4739612188365651, 0.4745152354570637,
0.4745152354570637, 0.47562326869806093, 0.4759002770083102, 0.47645429362880887, 0.47673130193905816, 0.4775623268698061, 0.4775623268698061,
0.4789473684210526, 0.4792243767313019, 0.47977839335180056, 0.48005540166204985, 0.4806094182825485, 0.4817174515235457, 0.48310249307479225,
0.48337950138504154, 0.4839335180055402, 0.4842105263157895, 0.4842105263157895, 0.48476454293628807, 0.4850415512465374, 0.4858725761772853]
[2021-05-01 05:03:51,049][root][INFO] - Saved results * scores  to dense_retriever_results
```

Figure 1: Retriever Validation Results

Fortunately, DPR provides scripts for people to use their own data format in DPR Hydra Configuration, which I will check to see if we that can be applied to our own dataset in the next step.

- Retriever Training

  I trained the DPR Retriever from scratch using the NQ dataset, which contains the train/dev sub-datasets. The computing resource I referred to is MSI V100 8*32 GPUs. I distributed the training mode on 8 GPUs. For time consideration, I only trained it for 25 epochs. Theoretically, the more the better.

  I trained the DPR Retriever just to get more familiar with the DPR architectures. Actually, we don't need to do that since DPR already provided the pre-trained (Bi-encoder) checkpoints for training NQ Retriever dataset. Moreoever, they even provided a new checkpoints set for a new dataset combined with original NQ training set and its version where hard negatives are mined using DPR index itself, which boosted the Retriever performance.

- Retriever Inference

  This part is to get the wikipedia embedding generations.

  Due to the memory issue, I only picked one passage encoder, or say, the dense embeddings of retriever results, and increased the memory to 64GB for inference part. Figure 1 shows the validation results for document retrieval.
  The retrieved results were saved to corresponding files for Reader training stage in JSON format. Those retriever results are actually search process based on the wiki-pedia embeddings we trained. The Retriever searched and sorted the results by their similarity score and then stored them into JSON files. Figure 2 shows the examples of those retrieved documents.

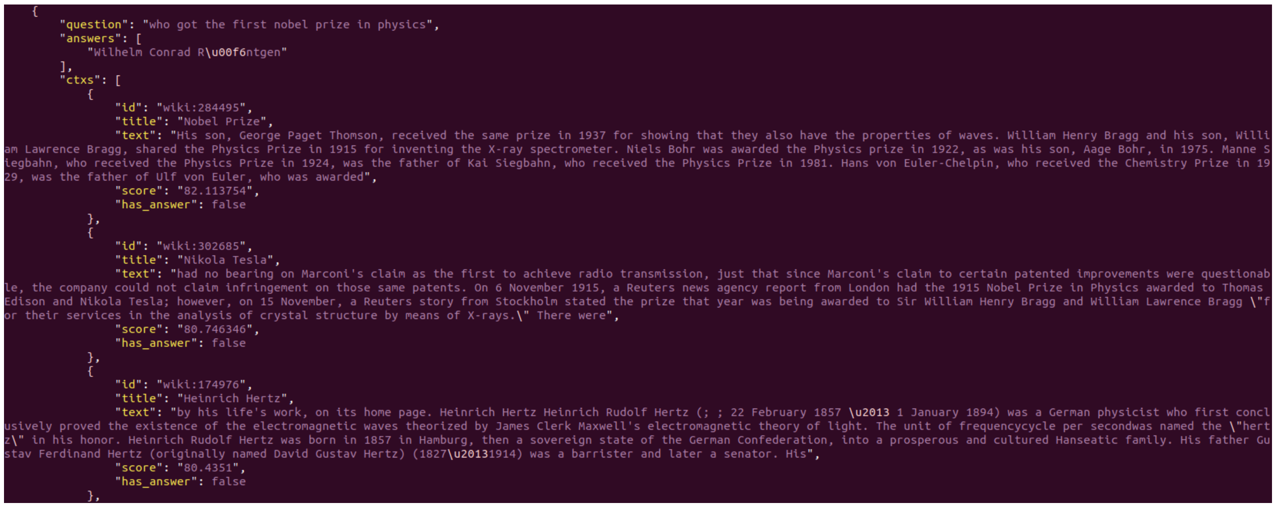
- Reader Training I trained reader model twice, one is for general NQ dataset reader training, another is with gold-passages-src and gold-passages-src-dev specified info, which can be used for special NQ only heuristics when preprocessing the data. Computing resource was V100 32GB.

- Reader Inference Results General NQ:

2.3 Haystack Toolkit

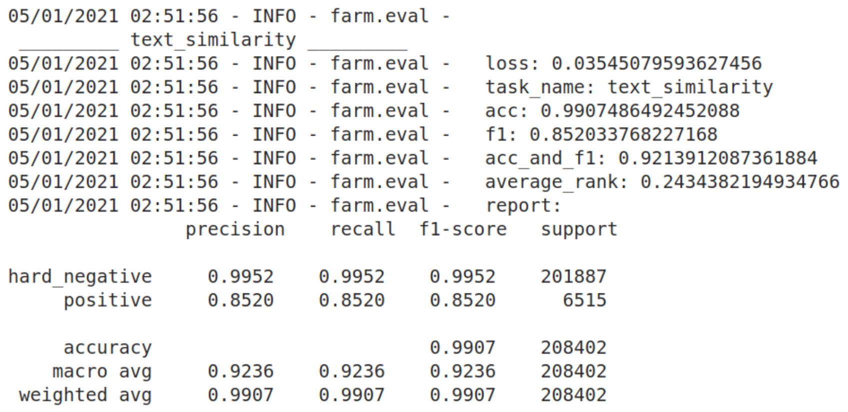
As an End-To-End toolkit, Haystack is pretty easy to use. I trained my own "Dense Passage Retrieval" model on Google's Natural Questions dataset which contained 7.4GB train set and 800MB dev set. The computing resource I referred to was V100 32*2GPU. figure 3 shows the report for NQ dataset retriever training.

Again, the dataset format is specified for DPR retrieval training. Figure 4 shows the example for the train set format. Apparently, even the SQuAD dataset is the JSON type, the format is not exactly the same, hence needed to be transformed before putting into training. I was planning to fine-tuning the DPR encoder model (domain specific question answering) by using SQuAD dataset and tried to convert SQuAD into DPR NQ format using squad-to-dpr script, however, it still ran into errors after several trials.



```
    {
        "question": "who got the first nobel prize in physics",
        "answers": [
            "Wilhelm Conrad R\u00f6ntgen"
        ],
        "ctxs": [
            {
                "id": "wiki:284495",
                "title": "Nobel Prize",
                "text": "His son, George Paget Thomson, received the same prize in 1937 for showing that they also have the properties of waves. William Henry Bragg and his son, Willi
am Lawrence Bragg, shared the Physics Prize in 1915 for inventing the X-ray spectrometer. Niels Bohr was awarded the Physics prize in 1922, as was his son, Aage Bohr, in 1975. Manne S
iegbahn, who received the Physics Prize in 1924, was the father of Kai Siegbahn, who received the Physics Prize in 1981. Hans von Euler-Chelpin, who received the Chemistry Prize in 19
29, was the father of Ulf von Euler, who was awarded",
                "score": "82.113754",
                "has_answer": false
            },
            {
                "id": "wiki:302685",
                "title": "Nikola Tesla",
                "text": "had no bearing on Marconi's claim as the first to achieve radio transmission, just that since Marconi's claim to certain patented improvements were questionab
le, the company could not claim infringement on those same patents. On 6 November 1915, a Reuters news agency report from London had the 1915 Nobel Prize in Physics awarded to Thomas
Edison and Nikola Tesla; however, on 15 November, a Reuters story from Stockholm stated the prize that year was being awarded to Sir William Henry Bragg and William Lawrence Bragg \"f
or their services in the analysis of crystal structure by means of X-rays.\" There were",
                "score": "80.746346",
                "has_answer": false
            },
            {
                "id": "wiki:174976",
                "title": "Heinrich Hertz",
                "text": "by his life's work, on its home page. Heinrich Hertz Heinrich Rudolf Hertz (; ; 22 February 1857 \u2013 1 January 1894) was a German physicist who first concl
usively proved the existence of the electromagnetic waves theorized by James Clerk Maxwell's electromagnetic theory of light. The unit of frequencycycle per secondwas named the \"hert
z\" in his honor. Heinrich Rudolf Hertz was born in 1857 in Hamburg, then a sovereign state of the German Confederation, into a prosperous and cultured Hanseatic family. His father Gu
stav Ferdinand Hertz (originally named David Gustav Hertz) (1827\u20131914) was a barrister and later a senator. His",
                "score": "80.4351",
                "has_answer": false
            },
```

Figure 2: Retriever Examples

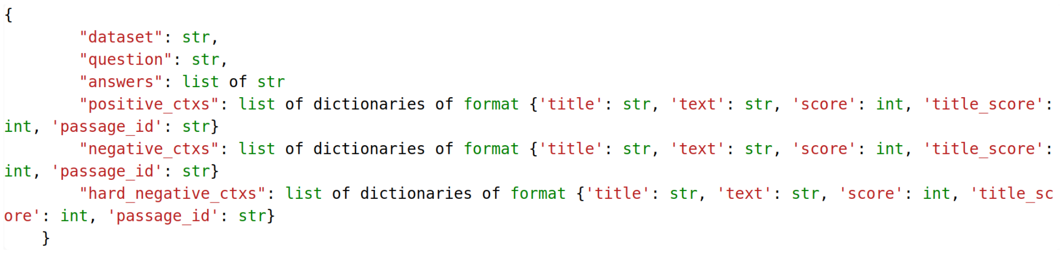


```
05/01/2021 02:51:56 - INFO - farm.eval -
 _________ text_similarity _________
05/01/2021 02:51:56 - INFO - farm.eval -   loss: 0.03545079593627456
05/01/2021 02:51:56 - INFO - farm.eval -   task_name: text_similarity
05/01/2021 02:51:56 - INFO - farm.eval -   acc: 0.9907486492452088
05/01/2021 02:51:56 - INFO - farm.eval -   f1: 0.852033768227168
05/01/2021 02:51:56 - INFO - farm.eval -   acc_and_f1: 0.9213912087361884
05/01/2021 02:51:56 - INFO - farm.eval -   average_rank: 0.2434382194934766
05/01/2021 02:51:56 - INFO - farm.eval -   report:
                precision    recall  f1-score   support

hard_negative      0.9952    0.9952    0.9952    201887
     positive      0.8520    0.8520    0.8520      6515

     accuracy                          0.9907    208402
    macro avg      0.9236    0.9236    0.9236    208402
 weighted avg      0.9907    0.9907    0.9907    208402
```

Figure 3: Report for Haystack training



```
{
       "dataset": str,
       "question": str,
       "answers": list of str
       "positive_ctxs": list of dictionaries of format {'title': str, 'text': str, 'score': int, 'title_score':
int, 'passage_id': str}
       "negative_ctxs": list of dictionaries of format {'title': str, 'text': str, 'score': int, 'title_score':
int, 'passage_id': str}
       "hard_negative_ctxs": list of dictionaries of format {'title': str, 'text': str, 'score': int, 'title_sc
ore': int, 'passage_id': str}
   }
```

Figure 4: DPR Training data format

2.4 Summary and Next Steps

1. I trained DPR Retriever from scratch with the computing resource V100 8*32GPUs for 25 epochs to get the Biencoder(Retriever). The datasets are: (train 2.0 GB/dev 815.6 MB) downloaded by DPR download-data script.
2. Validated Retriever Inference on one wikipedia passage embedding to see the Retriever performance.
3. Trained the DPR reader for NQ retriever results
4. Nowadays, there already exists a lot of good toolkits for DPR purpose, what I was doing is to be familiar with DPR architectures and get to know the feasibility of DPR through their source code by training DPR Retriever and Reader on NQ dataset.
5. Trained my own "Dense Passage Retrieval"model on NQ dataset contained 7.4GB train set and 800MB dev set using Haystack and got the result report. The datasets are: (train/dev)
`https://dl.fbaipublicfiles.com/dpr/data/retriever/biencoder-nq-train.json.gz`
`https://dl.fbaipublicfiles.com/dpr/data/retriever/biencoder-nq-dev.json.gz`
6. Although toolkits like HuggingFace, Haystack can load the pretrained DPR retriver model directly, however, the pretrained retriever/encoder model they provide is pretty limited (nq-related retriever), so if we want to fine-tuning the DPR Retriever for specific domain question answering, we still have to refer back the DPR source code, then what I was doing is like preparation for future possible DPR Retriever fine-tuning.
7. I only tried NQ dataset, next step I will focus on the Quiz Bowl dataset to see how we can play with that.

References

[1] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.