Reproducibility of the Experimental Result of BERT for Evidence Retrieval and Claim Verification

Thummaoon Kunauntakij
MSc Data Science Student,
Technische Universität Wien,
e12122522@student.tuwien.ac.th

András Bonifác Kónya
MSc Data Science Student,
Technische Universität Wien,
e01502933@student.tuwien.ac.th

Branimir Raguž
MSc Data Science Student,
Technische Universität Wien,
e12123474@student.tuwien.ac.th

ABSTRACT

We attempt to reproduce the result of BERT for Evidence Retrieval and Claim Verification [1]. The original paper use BERT for the task of evidence-based claim verification using FEVER dataset 50K Wikipedia pages [2] and it achieves a new state of the art recall of 87.1 for retrieving evidence sentences the dataset, and scores second in the leaderboard with the FEVER score of 69.7. We discuss their experiment design, metric used and attempt to reproduce their result. By reviewing the process describe by the original paper, we conclude that their experiment design is questionable, and the result might not be able to generalize well. Although we are not able to confirm the number due to various difficulties encountered for recreating the dataset and the time frame limitation, we document the list of problem and our effort we have done to resolve them in the process.

CCS CONCEPTS • Evidence retrieval • Claim verification • BERT

1 INTRODUCTION

1.1 SUMMARY OF THE ORIGINAL PAPER

BERT for evidence retrieval and claim verification is a method that utilize *BERT*, Bidirectional Encoder Representations from Transformers, which is a natural language processing (NLP) pre-training machine learning technique developed by *Google*. Two *BERT* models are proposed in the paper, one for retrieving evidence sentences supporting or rejecting claims, and another for verifying claims based on the retrieved evidence sentences. To train the *BERT* retrieval system, they used pointwise and pairwise loss functions and examined the effect of HNM (Hard Negative Mining). They applied the method to a shared task, *FEVER: a Large-scale Dataset for Fact Extraction and VERification* [2] which provided a benchmark dataset. The resulting model scored second in the leaderboard with the *FEVER* score of 69.7.

Their methods can be divided into 3 discrete modules.

1.1.1Document retrieval

In this step, a claim is given, the module finds English Wikipedia articles containing information about this claim. The paper uses a module found in *UKP-Athene* [3] to collect a set of top documents.

1.1.2Sentence retrieval

The sentence retrieval step extracts the top five evidence sentences. *BERT* is trained for predicting the next sentence so it can be used for finding top relevant sentences for the documents retrieved in the first module. They use it by adding a softmax at the last hidden state of the classification token and trained together with the pre-trained layers. They fine-tune it using two different pointwise and pairwise approaches. To prevent training on trivial samples, they applied online hard negative mining (HNM) to focus the training on harder pair of claim and evidence.

1.1.3Claim verification

Each claim is compared against the top sentences from the previous module and the final claim classification is determined by aggregating the five individual decisions. The default label is not enough information (NEI) unless there is any supporting evidence to predict the claim as supported. If there is at least one rejecting evidence and no supporting fact, the label is refuted. They trained another *BERT* as a three-class classifier to do the task.

2 EXPERIMENT SETUP

By reviewing the original paper and their *GitHub* repository [4]. We can summarize their experiment setup as the following.

2.1 Experiment Design

The summarization is listed below.

2.1.1 FEVER dataset [2] release 3 sets of data, training, development, and testing data.

2.1.2 The original authors preprocessed the data using document retrieval system from the *UKP-Athene* Project and reformat the result into a specific format.

2.1.3. They used 10 different combination of sentence retrieval and claim verification techniques as candidate methods. All of them utilize *BERT* in the sentence retrieval step and either *BERT* or *UKP-Athene* in the claim verification step.

2.1.4. Train all the combination using the shuffled training set.

2.1.5. Run the model against the development set and compare the result various metrics; Precision, Recall, F1, label accuracy and on the shared task metrics, *FEVER* Score.

2.1.6. Finally, the authors run the model against the test dataset and submit it for evaluation.

2.2 Discussion of the Experiment Design

Because the paper aimed at participating in a shared task of a workshop. The experiment process of the paper is rather simple, straight forward and probably was not designed for verifying the result scientifically. The setup used all the training data to train all the candidate methods and the result models are validated against the same development set. The only randomness feature in the process is in the shuffling of the training data. There is no mention of repeated trials at all, thus there is no report for the variance of the evaluation metrics. Also, because they run multiple methods through the same process and select the best one without repeating them on a different spilt of the data, we could question about the generalization of the result on different subset of data and unseen data. We can see that the *FEVER* score and label accuracy dropped from the development set, at 72.42% and 74.59% respectively to 69.66%, 71.86% respectively. It beat the next couple of the best methods by less than 0.5%.

3 REPRODUCIBILITY

3.1 Code

As mentioned, the authors of the original paper released the source code they used in a *GitHub* repository [4]. Also the source code of *UKP-Athene* is available as well.

3.2 Hyperparameter

The authors explicitly stated the hyper parameter they used for training the model in the paper including batch size, learning rate, the number of epoch and specification an additional method, *Hard Negative Mining (HNM).*

3.3 Dataset

3.2.1 Raw Data

The authors use the dataset from a shared task, *FEVER* dataset [2]. The task contains 4 sets of data; training set, development set, testing set and preprocessed Wikipedia pages.

3.2.2 Preprocessed Data

The authors utilize part of the code from UKP-Athene Project [3], specifically the output of the document retrieval module of the project. They use this to select a list of most relevant documents for the claim. Then, they reformat the output into 3 TSV (tab separated values) files; *train_sentences_pos.tsv, train_sentences_neg_32.tsv* and *dev_sentences.tsv* to be used as an input in sentence retrieval step. Lastly, in the claim verification step, they use *train.tsv, train_negative.tsv* and *dev.tsv* as an input. It is unclear what is the structure of the files for this step.

3.4 Attempt to reproduce the result

The main obstacle for reproducing the result of this project is to acquire the data. The raw data from FEVER dataset are freely available on its webpage. But to preprocess the data as an input to this project, we need to use a script from *UKP-Athene.* However, *UKP-Athene* depended a lot on outdated dependencies. As we have no prior experience in deep learning and natural language processing, it is not possible for us to develop the software by ourselves. Thus, we invested almost all of the time we have for debugging the *UKP-Athene* dependency and still unable to acquire the required data. As a result, we could not be able to reproduce the result within the given time frame. However, we documented the issues we experience and how we resolved them to show the effort that we have done.

3.4.1Setting up the environment for UKP-Athene: Project Dependencies

There are several issues we've experience with UKP-Athene. As we had not many experiences in related technology. These problems costed us an enormous amount of time to figure out how to resolve them.

It is developed in an end-of-life version-3.6 *Python* and contain multiple deprecated software dependencies. First, we tried to resolve the issues by *Pipenv*, a Python package manager. We found that a specific *Pytorch* version is used (version 0.4.0) and made it unable to be install by a general package manager like Pip or *Pipenv* as it was not hosted on

Pypi repository anymore. The solution suggested by the *Pytorch* website is to install it via *Conda*, which is another package manager that has its own repository. By using *Conda*, we were also able to specify a specific version of *Python* as well.

They were multiple of deprecated software package that were used. *Pyfasttext* was already deprecated and the developer pull the package down from the repositories. They suggest us to migrate to the official *Fasttext Python*-binding package called *Fasttext* which required us to migrate the original code to a new library.

Some dependencies are not compatible with current build tools. We must update them up from the required version specified in *requirements.txt*, a file containing the list *Python* dependencies with their exact version. These included *MarkupSafe, Spacy* and *Sanic.*

UKP-Athene also relied on two packages that is hosted in *GitHub*; *DrQA* and *FeverScorer*. These packages contained some outdate build script. We resolved them by forking both source code and fix the error. Then, we installed them from our *Github* repository instead.

After installing the dependencies, we encountered with another issues when we tried to run the code on our local machine which is operated by *Windows 10*. We run the code using *Ubuntu* on *Windows Subsystem for Linux version 2 (WSL2)*. It turned out that *WSL2* cannot reliably use a graphic processing unit, GPU, of the host system. We found that by using *Docker* the *Linux* inside it can access the GPU. However, we still could not run the script from UKP-Athene. This was caused by incompatible between dependency packages, specifically *Pytorch* and *Tensorflow* with a recent *CUDA* driver, a parallel computing platform and application programming interface (API) for *Nvidia* GPU. The problem was resolved by using an old version of *Docker* container, *nvidia/cuda:9.0-cudnn7-devel*, hosted by *Nvidia* containing *CUDA* version 9.0 and *CUDNN* 7.0, a deep neural network library for *CUDA*.

However, the problem still existed. This time it was the incompatible of the *CUDA* driver with the GPU in our laptop. Specifically, we have a laptop with *Nvidia RTX 3080*, a GPU that had released in January 2021. It had a *CUDA* compute capability at 8.6 but *CUDA* version 9.0 could only operate a GPU with compute capability up to 7.0. To resolve this, we migrate the whole system to *Google Compute Engine*, a cloud service which offer *NVIDIA V100*, a GPU with compute capability at 7.0. At this point, we could run the script from UKP-Athene.

3.4.2Running UKP-Athene Document Retrieval

We run the document retrieval script from *UKP-Athene*. The unresolved problem we face was that it tried to pull the millions of *Wikipedia* pages from an online service but at halfway, after a couple of hours, the service always rejected our request due to the traffic at the server. Thus, we could not get the data required.

3.4.3Another Possible Source of dataset

Later, we found that *UKP-Athene* host some intermediate dataset that was already processed by the document retrieval script. We hoped that it could be reformat into a TSV file and use them for this project. However, we found that the dataset used a different *Wikipedia* document. We could not join them with preprocessed *Wikipedia* data from FEVER dataset as there are many documents retrieved in the pre-processed data that were not found.

3.4.4 Imprecise Specification of the input data

Even if we could get the data from *UKP-Athene*. The authors still were too imprecise about how to format the input data. As we found the instruction quoted from the project GitHub repository, *"We cannot provide the dataset but you should download it from the FEVER website and use document retrieval codes like Athene's code to get the sentences and then make a TSV file."*. They only provide a sample data file for sentence retrieval without explaining the structure of it and they did not give any example for the claim verification step at all. We could only guess what each of the columns in the file mean and do some trial and error to check if it is accepted by the script.

4 CONCLUSION

The paper we attempted to reproduce was a simple and straight forward. However, because of the simpleness of their experiment design and metric used, it is questionable whether the methods could perform as good if we apply them with new dataset. Also, even if the authors open source their code for publicly used and the usage of open dataset from the FEVER shared task, the impreciseness instruction of how to get the processed data required by the script make it difficult to reproduce the result. Thus, we believe that it is hard for other researchers, especially one with less software development experience, who interest in the same task to use the result from this paper as a benchmark in the future.

REFERENCES

[1] Soleimani A., Monz C., Worring M.: BERT for Evidence Retrieval and Claim Verification. In: Jose J. et al. (eds) Advances in Information Retrieval. ECIR 2020. Lecture Notes in Computer Science, vol 12036. Springer, Cham. https://doi.org/10.1007/978-3-030-45442-5_45

[2] Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A.: FEVER: a large-scale dataset for fact extraction and verification. arXiv preprint arXiv:1803.05355 (2018)

[3] Hanselowski, A., et al.: UKP-Athene: multi-sentence textual entailment for claim verification. In: Proceedings of the First Workshop on Fact Extraction and VERification (FEVER), pp. 103–108. Association for Computational Linguistics, Brussels, November 2018. https://doi.org/10.18653/v1/W18-5516

[4] Soleimani A.: ASoleimaniB/BERT_FEVER, Github.com, https://github.com/ASoleimaniB/BERT_FEVER/tree/d630e7150554c72319b37729f0522b462b63603c (2020)

A APPENDICES

A.1 The final Docker file for creating the environment for UKP-Athene.

```
FROM nvidia/cuda:9.0-cudnn7-devel
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8
ARG PATH="/root/miniconda3/bin:${PATH}"
RUN apt-get update
RUN apt-get install -y wget && rm -rf /var/lib/apt/lists/*
RUN wget \
  https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh \
  && mkdir /root/.conda \
  && bash Miniconda3-latest-Linux-x86_64.sh -b \
  && rm -f Miniconda3-latest-Linux-x86_64.sh
RUN conda create -y -n myenv python=3.6
ENV PATH /opt/conda/envs/myenv/bin:$PATH
RUN apt update
RUN apt-get install -y software-properties-common
RUN add-apt-repository ppa:deadsnakes/ppa
RUN apt install -y git
RUN apt install -y vim
RUN apt install -y unzip gzip
RUN conda install -n myenv pytorch=0.4.0 cuda90 -c pytorch
COPY ./requirements.txt /workspace/requirements.txt
RUN /root/miniconda3/envs/myenv/bin/python -m pip install -r /workspace/requirements.txt
RUN /root/miniconda3/envs/myenv/bin/python -c "import nltk; nltk.download('punkt')"
WORKDIR workspace
```

A.2 Change Made in requirement.txt for the dependencies of UKP-Athene.

Table 1: Change made in requirement.txt for the dependencies of UKP-Athene

From	To
cymem==1.31.2	cymem==1.31.2
cytoolz==0.8.2	cytoolz==0.9.0.1
MarkupSafe==1.0	MarkupSafe==2.0
numpy==1.14.5	numpy>=1.14.5
numpy>=1.14.5	numpy>=1.14.5
pyfasttext==0.4.5	fasttext==0.9.2
fasttext==0.9.2	sanic==0.7.0
sanic==0.7.0	removed
tensorboard==1.8.0	removed
tensorboard-pytorch==0.7.1	removed
tensorboardX==1.2	removed
tensorflow-hub==0.1.0	removed
thinc==6.10.2	thinc==6.12.1
torch==0.4.0	removed
torchvision==0.2.1	removed
git+git://github.com/j6mes/drqa@parallel	git+git://github.com/gentlerainsky/DrQA
git+git://github.com/sheffieldnlp/fever-scorer@master	git+git://github.com/gentlerainsky/fever-scorer