Microsoft AI Challenge India 2018: Relevant Passages for Web Question Answering with RoBERTa based classification model

Priya Bhatia[1] and Akshita Parekh[2]

[1]Mtech in AI Department,IIT Hyderbad
[2]PhD in CSE Department,IIT Hyderabad

Abstract

This report describes our system for The Microsoft AI Challenge India 2018: Relevant Passages for Web Question Answering. The system uses Simple Transformers to train the model by initializing the task-specific model which mainly contains two things : model type and model name. We also incorporate hand-crafted features to improve the overall system performance. Our system achieved an overall accuracy of 65 percent when we use 0.005 percent of the given dataset.

1 Introduction

The digital world today is filled with humongous amount of data. The netizens today try to find answer to every problem on internet. In response to a single query they get multiple results within a friction of second. As a result of which users tend to get confused and end up losing their valuable time and energy searching for the appropriate answer.

An automatic web based question answering (QA) system is an important tool for improving education and e-learning. The QA system automatically returns a passage in answer to a user's query instead of returning multiple links.

To explore the various practical approaches for this problem, Microsoft India took an initiative to evaluate the ranking of passages for a given user query namely "Microsoft AI Challenge 2018".

The rest of the report is organized as follows: In section 2, we analyze the data and describe the data preparation and data pre-processing tasks. The details of the training and testing dataset is described in section 3. In section 4, we describe the details of the model. Experiments and results are presented in section 5. We conclude the report in section 6.

2 Data Preparation and Pre-Processing

2.1 Data Description

The dataset used is as provided by the mentor, with no external corpus. The dataset file size is approximately 2GB where each sample contains five fields mainly: query id, query, passage, query label and query count. For each query, we have 10 different passages in which nine are labelled as 0 and remaining one passage is labelled as 1 which denotes the relevance of that passage corresponding to the user's query.

2.2 Data Preparation

As mentioned above, the given dataset was huge in size. Processing and working on this file in local environment was challenging due to the resource limitations. So, Big Data concepts were used to systematically extract the information. Hadoop was used for storage and Spark for processing the file. It was observed that the available dataset was highly imbalanced,as the ratio of negative to positive label was 9:1. So, the given dataset was down-sampled and the ratio of negative and positive label was updated to 50:50. Then the data was divided into two parts mainly: Training and Testing Data as described in next section.

3 Training and Testing Dataset

The ratio of training to testing was maintained as 80:20. sklearn library was used for splitting the data.

Three fields namely: Query, Passage and label were extracted for binary classification purpose. The same data was used for training and testing purpose.

Notations: For each query, there were two passages in the form $< q, p^+, p^- >$. Where q is the query, p^+ and p^- are the possible passages for this query. p^+ indicates the positive label passage means relevant passage and p^- indicates the negative label passage means irrelevant passage corresponding to the query q.

```
Query = list(final['Query'])
Passage = list(final['Passage'])
label = list(final['label'])
dataset = []
```

4 Model Description

In this section, we describe the architecture of our model. Here, we use Simple Transformers for training and evaluation of our model.

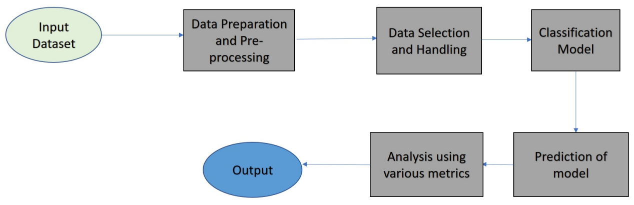

Input Dataset
Data Preparation and Pre-processing
Data Selection and Handling
Classification Model
Prediction of model
Analysis using various metrics
Output

Figure 1. Flow Diagram of the Project

4.1 Simple Transformer

Simple Transformer models are built with a particular Natural Language Processing task in mind. Each such model comes equipped with features and functionalities designed to best fit the task that they are intended to perform.The high-level process of using Simple Transformers models follows the same pattern.

Initialize a task-specific model

Train the model with train_model()

Evaluate the model with eval_model()

Make predictions on unlabelled data with predict()

4.1.1 Creating a Task-Specific Model

To create a task-specific Simple Transformers model, we will typically specify a **model_type** and a **model_name**.

model_type should be one of the model types from the supported models (e.g. bert, electra, xlnet)

model_name specifies the exact architecture and trained weights to use.

4.1.2 Importing the task-specific Model

```
from simpletransformers.classification
import ClassificationModel
```

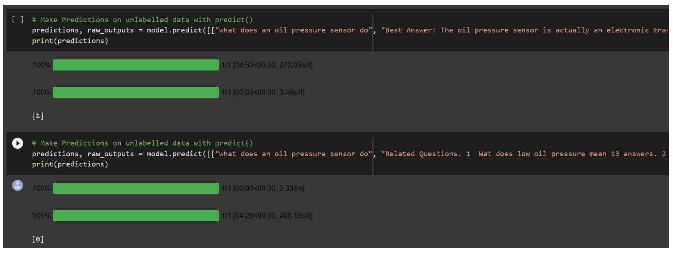
4.1.3 Loading a Pre-trained model

```
model = ClassificationModel
        ('roberta', 'roberta-base'
        )
```

4.2 RoBERTa

Roberta is a retraining of BERT with improved training methodology, 1000 percent more data and compute power. To improve the training procedure, RoBERTa removes the Next Sentence Prediction task from BERT's pre-training and introduces dynamic masking so that the masked token changes during the training epochs. Larger batch-training sizes are also found to be more useful in the training procedure.

RoBERTa uses 160 GB of text for pre-training, including 16GB of Books Corpus and English Wikipedia used in BERT.



```
[ ] # Make Predictions on unlabelled data with predict()
    predictions, raw_outputs = model.predict([["what does an oil pressure sensor do", "Best Answer: The oil pressure sensor is actually an electronic tra
    print(predictions)

    100% 1/1 [04:30<00:00, 270.05s/it]

    100% 1/1 [00:03<00:00, 3.46s/it]

    [1]

    # Make Predictions on unlabelled data with predict()
    predictions, raw_outputs = model.predict([["what does an oil pressure sensor do", "Related Questions. 1  Wat does low oil pressure mean 13 answers. 2
    print(predictions)

    100% 1/1 [00:00<00:00, 2.33it/s]

    100% 1/1 [04:26<00:00, 266.19s/it]

    [0]
```

Figure 2. Predictions

The additional data included CommonCrawl News dataset (63 million articles, 76 GB), Web text corpus (38 GB) and Stories from Common Crawl (31 GB). This together with 1024 V100 Tesla GPU's running for a day, led to pre-training of RoBERTa. As a result, RoBERTa outperforms both BERT and XLNet.

5 Experiments and Results

The above model was used to make predictions on the same query and different passages. The observations are noted below.

Query : What does an oil pressure sensor do

Output : Model predicts 1 for relevant passage and 0 for irrelevant Passage corresponding to the same query mentioned above.

Figure 2 shows that for relevant passage our model classifies it as 1 and for irrelevant passage it classifies it as 0.

6 Conclusion

Further following metrics were obtained from the test done on the above model.

Metrics	Epoch=1	Epoch=2
Accuracy	66	65
Recall	91	80
Precision	60	62
F1 Score	72	70

The test was conducted with different number of epochs and 0.005 fraction of sample data due to the resource limitations of personal computer. The data size was increased from 0.002 to 0.005 percent.

It was understood that with the increase in size of the dataset and the number of epochs, the precision would further increase and leading to better results.

7 Future Scope

It would be interesting to use context aware embeddings such as ELMo, Glove etc. in future work. It is also desired to further observe the results by replacing simple transfer models with transformer networks. Additionally, it would be interesting to explore useful hand-crafted features and ensembling methods.

References

[1] [Peters et al., 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations *In Proc. of NAACL,2018*

[2] [Mueller and Thyagarajan, 2016] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16,pages 2786–2792. AAAI Press, 2016.

[3] [Kiela et al., 2018] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP),* Brussels, Belgium, 2018.

[4] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. SpanBERT: Improving pre-training by representing and predicting spans.*arXiv preprint arXiv:1907.10529.*

[5] [Devlin et al., 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint *arXiv:1810.04805, 2018.*

[6] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In*North American Association for Computational Linguistics (NAACL).*

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems.