Preprocess superGLUE and Finetune BERT Based on Boolq Task

1st Chengrui Zhao
Department of Statistics
University of Michigan, Ann Arbor
Ann Arbor, the United States
chengrui@umich.edu

***Abstract*—This document is a project for stats 507, which contains preprocssing datasets and finetune model in GLUE bench using Huggingface Transformer, the major used material includes superGLUE dataset and BERT model, and GLUE and distillBERT are used as comparsion. Th code results are in "bert-base-uncased-finetuned-boolq" and "distilbert-base-uncased-finetuned-cola" codebase in github**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

A. Background

This document is the final project for stats 507, it's conducted with the aim of not only obtaining good grade, but also trying to explore and learn more about deep learning.

Nowadays more and more statistical-related positions, such as data engineer, data scientist and statistician require the knowledge of preprocessing datasets and fine-tuning models in order to learn more about this, the first project is chosen instead of image processing.

B. Project Goal

By the end of project, the method of preprocessing datasets and fine-tuning models using HuggingFace transformer should have been obtained. After the training of model with the training set included in the dataset, by inputting the required sentences in the API of the model, the result can be shown correctly with the task given from the dataset.

Also, the result of two similar datasets and models will be compared together based on their metrics, accuracy, operation speed and so on, then conclude which model is better. Optimization methods will be chosen and used for the model.

C. Existing Literature

In the last year, new models and methods for pretraining and transfer learning have driven striking performance improvements across a range of language understanding tasks. The GLUE benchmark, introduced a few years ago, offers a single-number metric that summarizes progress on a diverse set of such tasks, but performance on the benchmark has recently surpassed the level of non-expert humans, suggesting limited headroom for further research.The superGLUE is invented to cope with such problems as it deals with a new set of more complex problems. [1] SuperGlue is a neural network that matches two sets of local features by jointly finding correspondences and rejecting non-matchable points. Assignments are estimated by solving a differentiable optimal transport problem, whose costs are predicted by a graph neural network. [2] The principle of how superGLUE find correspondence part in two parts can be shown as below:

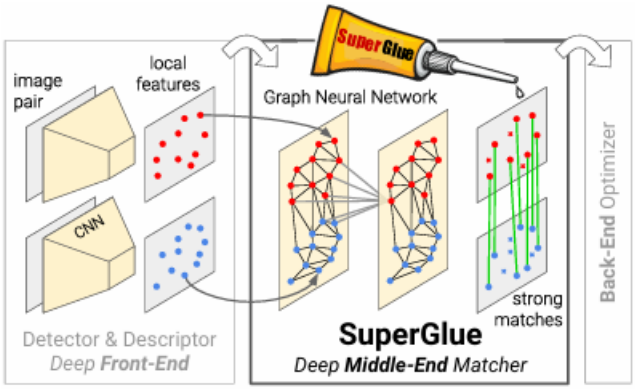

Fig. 1. part of superGLUE principle for finding correspondence parts.

There exist several different kinds of dataset in superGLUE, each corresponding to one unique task, such as judging the relationship between premise and hypothesis, answering one question based on a paragraph and explain what the pronoun represent. There exist a task called Boolq, which automatically judge the true or false of the question based on a passage. The Boolq has the biggest dataset among all tasks, which can improve the accuracy and decrease the need for optimization, so in this project it's used.

Since Boolq requires finding answers of questions from long passages, it needs a model that has great contextual comprehension of long passages, easily discriminating the keywords from the sentence of the question. Here BERT is a good choice. In contrast to earlier methods that use convolution and recurrent modules for feature extraction, BERT learns bidirectional encoder representations from Transformers, trained on large datasets as contextual language models. BERT shows

Identify applicable funding agency here. If none, delete this.

superior performance for fine-tuning it to similar small-scale tasks, making it possible to initiate rapid data processing. [3]

DistilBERT is a distilled version of the BERT (Bidirectional Encoder Representations from Trans-formers) model, designed for natural language processing tasks such as Question Answering (QA). DistilBERT is often character-ized by smaller dimensions and parameters, designed to be smaller andfaster than the original BERTmodel. However, this design choice may result in a slight loss of information compared to the origi-nal BERT model. Nevertheless, it offers advantages, especially in resource-constrained environments.Furthermore, it preserves the general learning capabilities of theoriginal BERT model. [4]

II. Method

For this section, since the methods for superGLUE with BERT and GLUE with distillBERT are the same, they are written together with superGLUE and BERT as example

A. Problem formulation

With the Boolq task, the input should be two sets of local feature, one is the question that needs to be answered, the other one is the passage that contains information to answer the question. After inputting them in the API, the model should return True or False. For example, if entering "is calgary the largest city in north America", "Calgary – The city had a population of 1,239,220 in 2016, making it Alberta's largest city and Canada's third-largest municipality. Also in 2016, Calgary had a metropolitan population of 1,392,609, making it the fourth-largest census metropolitan area (CMA) in Canada", the answer will show "False". As is shown in the example, the keyword needs to be emphasized in the passage and the passage has to be really related to the question.

There are over 15,000 data in the dataset of Boolq, about 9000 of which are training set, which trains the model (BERT) about how to find answer of question from the passage; 3000 of which are validation set, which evaluates the quality of model after each epoch of training; 3000 of which are testing set, which applies the model after training into some other data to see their accuracy. Since Boolq is a binary classification with only two answers True or False and the number of them are not equal (5892 True, 3535 False), there do exist the problem of imbalanced label distribution, to solve this, the dataset needs to be fine-tuned by adjusting the classification bias and distributing different weight to different kinds of the answer so that the total value and proportion of True and False are equal.

Discriminating the relationship between all words in the question and passage requires great contextual comprehension, which is achieved by BERT with the help of its multi-head attention. Attention is a mechanism for helping compute the embedding for a token. After the workpiece tokenizer by the HuggingFace Transformer, all words are tokenized as different numbers, then attention is used by conducting a regression with different coefficients distributed to different numbers. For example, let x1,x2,x3...xn be tokenized words, then Output ai = a weighted sum of x1 through xn Weighted by their similarity to xi, which can be presented as:

$$a_i = \sum_{j<=i} \alpha_{ij} x_j \tag{1}$$

Also, during this process multi-head attention can find the key word in question and passage which has the biggest weight (aij). First of all positional-aware encoding is conducted so that position of different words are known by the model, then extract query, key and value to he position encoding, the concrete procedure of this can be shown as:

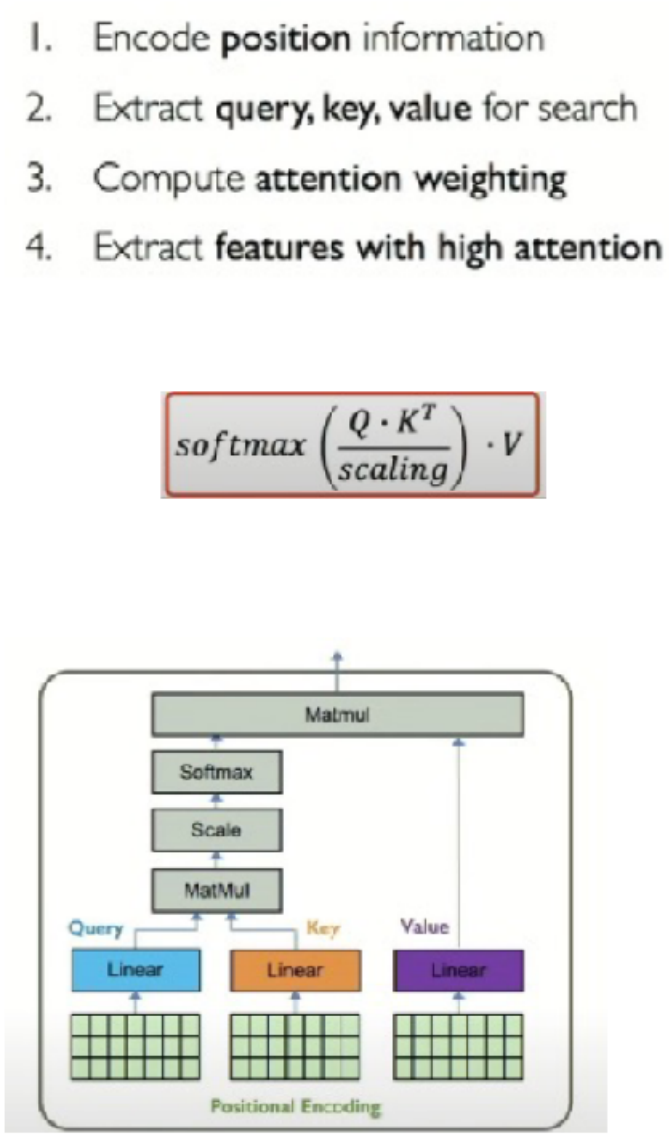

Fig. 2. procedure of finding key word by multi-head attention

Meanwhile, Feedforward Neural Network (FFN) also belongs to the formulation of BERT, after the tokenization and operation of attention, inputs will be forwarded into the hidden layers which is what's explored about deep learning in this project, there are 12 hidden layers in BERT model. After that the output can be shown, the concrete procedure of this will be shown in "data pipeline" part.

B. Walking Through Methodologies

In this subsection the concrete procedures of all involved in this project will be talked about:

First of all, Before preprocession and fine-tune, some preparation needs to be done. For example, as the huggingface transformer will be used, the login procedure is needed by installing notebook-login() from huggingface hub, Also, in order to let the final result be shown in API successfully, Git-LFS needs to be installed. Meanwhile, the task, dataset and model all needs to be confirmed.

After the confirmation, all of them needs to be downloaded: the task, model, metric used to evaluate for the result (Boolq uses accuracy), and dataset, if downloaded successfully, the following dataset can be shown:

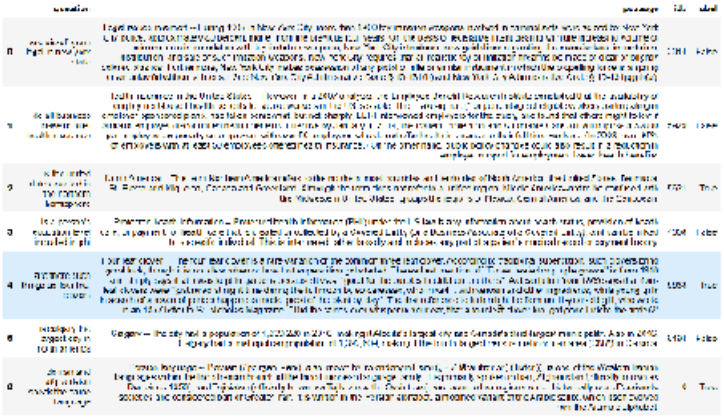

Fig. 3. examples of training datasets

Finally the preprocessing can be conducted. First define the dictionary for different tasks in superGLUE where the function of two sentences are written for each task. Then create an autotokenizer function to apply workpiece tokenization to every word in a sentence so that they are tokenized and ready to be operated, use map function to apply this function to the whole dataset. The concrete procedure will be drawn below.

After preprocessing dataset, it's time to finetune the model BERT. First apply automodelsequenceclassification to BERT to make it aware that there should be two types of answer: True and False (do this by setting num-label=2). Then set training argument to regulate how to train the model, here define that the evaluation should be done after 5 epochs of training, the batch size shall remain the same, the output of final model among all epochs will be selected with the metric and the problem of having imbalanced label distribution can be solved here by adjusting the classification bias and redistributing weight for True and False answer. Finally apply a function to the whole dataset about how to calculate metric and then simply by inputting trainer.train(), the model will be fine-tuned.

Since the transformer here is already of good quality and the output is a binomial classification, RNN isn't involved.

III. Results

A. data pipeline and model set up

For this section, since the pipeline for superGLUE with BERT and GLUE with distillBERT are the same, they are written together with superGLUE and BERT as example

Here the detailed figure of data pipeline of one example of data Based on procedure will be shown as well as the model set up of application of methods to the whole dataset. First is the detailed one:

Then is the application of procedure to the whole dataset:

B. numerical simulation result

After the completion of project, results can be shown, first is the performance of model BERT after the finetune with Boolq in preprocessed superGLUE dataset:

In comparision, the result with GLUE and distillBERT is:

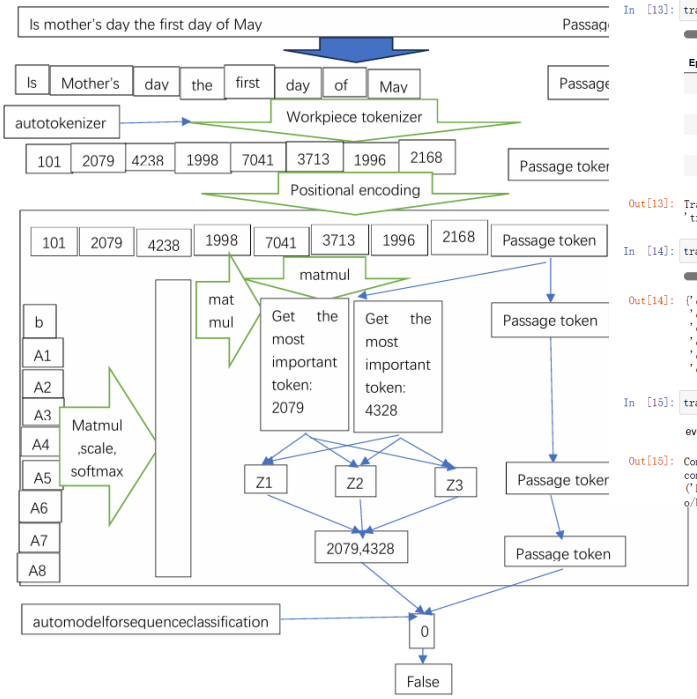

Fig. 4. concrete procedure of processing and finetune for one data

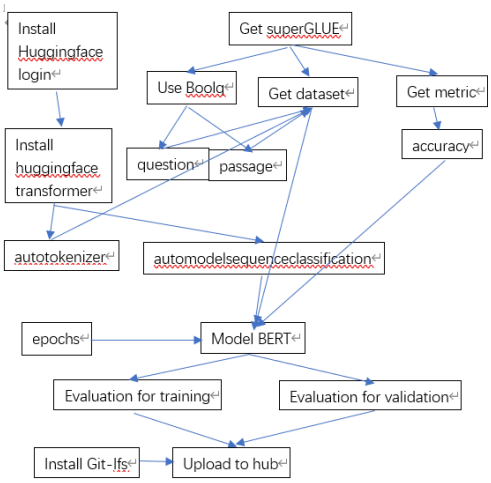

Fig. 5. model set up for the whole dataset

C. Implementation

From the figure of superGLUE dataset and BERT model it can be seen that the average loss of validation set is 0.956, which is much bigger than training loss, indicating overfit; the accuracy is 0.726, which is good; it takes 918 seconds to run this project, this is too much time which result from the great size and storage of the model, which also result in the relative slow time for 4 sample per second.

In comparison, with GLUE dataset and distillBERT model the average loss of validation set is 0.842, still indicating overfit but better compared with that in superGLUE dataset and BERT model; MCC=0.545¿0.5, indicating that the project

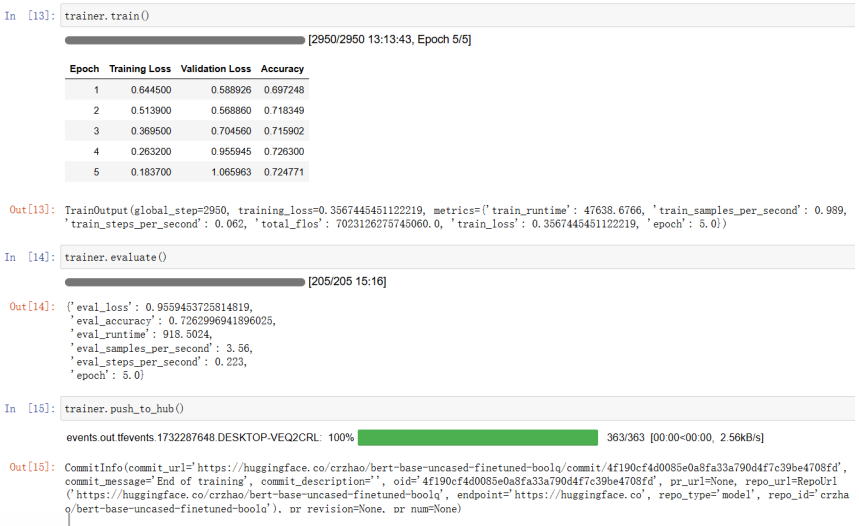

Fig. 6. the final result

{"epoch":1.0, "eval_loss":0.46771520376205444, "eval_matthews_correlation": 0.42525919625284747, "eval_runtime": 16.5354, "eval_samples_per_second": 63.077, "eval_steps_per_second": 3.991, "step": 535}
{"epoch": 2.0, "eval_loss": 0.4809904396533966, "eval_matthews_correlation": 0.509702586150945, "eval_runtime": 16.3678, "eval_samples_per_second": 63.723, "eval_steps_per_second": 4.032, "step": 1070}
{"epoch": 3.0, "eval_loss": 0.6194034814834595, "eval_matthews_correlation": 0.5155709926752544, "eval_runtime": 15.9856, "eval_samples_per_second": 65.246, "eval_steps_per_second": 4.129, "step": 1605}
{"epoch": 4.0, "eval_loss": 0.7826895713806152, "eval_matthews_correlation": 0.5126109268227362, "eval_runtime": 16.1201, "eval_samples_per_second": 64.702, "eval_steps_per_second": 4.094, "step": 2140}
{"epoch": 5.0, "eval_loss": 0.8416157960891724, "eval_matthews_correlation": 0.5451837431775948, "eval_runtime": 19.3136, "eval_samples_per_second": 54.003, "eval_steps_per_second": 3.417, "step": 2675}

Fig. 7. compared result

is meaningful; the runtime of this model is 19 seconds, much faster than the BERT due to its relatively small size, which result in 54 samples per second

IV. Conclusion

In conclusion, the distill BERT is better than BERT in its size and reduce overfit, with a bit worse in quality. So in real practice, if the quality requirement is extremely high, then use BERT, or otherwise use distillBERT to save time and storage.

References

[1] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, S. R. Bowman, "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems." NeurIPS, 2019

[2] P. Sarlin, D. DeTone, T. Malisiewicz and A. Rabinovich, "SuperGlue: Learning Feature Matching With Graph Neural Networks," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4938–4947, 2020.

[3] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang et.al, "A comprehensive survey on pretrained foundation models: a history from BERT to ChatGPT," International Journal of Machine Learning and Cybernetics, 2024.

[4] C. Ozkurt, “Comparative Analysis of State-of-the-Art Q and A Models: BERT, RoBERTa, DistilBERT, and ALBERT on SQuAD v2 Dataset,” researchsquare.com, 2024.