Quora-Question-Pairs Project

This project uses the same data set as a 5 years ago competition on the Kaggle. Although it is possible for you to find solutions on the website, we HIGHLY recommend you to construct your model by yourself.

introduction

Quora is a question-and-answer platform where question are created, answered, edited by its users. Its counterpart in China is Zhihu.

Considering the huge traffic to Quora every month, there are a number of similar questions. The goal of this project is to construct a model to judge whether two questions share the same meaning or not. You are excepted to tackle this problem by using what you've learned in the class, i.e., **the pre-trained language models in NLP**, including BERT, RoBERTa, to name but a few.

The details of the dataset are shown in Dataset. We will introduce the basic procedure for constructing a model in Getting Started. You also need to pay attention to the Submission Requirements.

Dataset

Each line in the dataset corresponds to an example, which contains 3 parts separated by `\t`:

- `question1, question2`. The full text of each question.
- `label`. The grand truth, set to `1` if `question1` and `question2` have essentially the same meaning, and `0` otherwise.

A typical example in the dataset looks like:

```
I got my Facebook account hacked. How can I recover my i.d.?    My Facebook account was
hacked. How can I recover my Facebook account?    1
```

NOTE: The original file also contains the `id` of each question. We dropped them because the utilization of this information is irrelevant to the main points of this project.

Due to we have no access to the official test file provided by Kaggle, we randomly split the train file into `train.txt`, `valid.txt`, and `test.txt`. The size of each dataset is shown as follows:

Dataset	Size
train	282,995
valid	80,855
test	40,429

Getting Started

Different from the OPPO dataset used in the class, the questions in this project have not been desensitized. So at the very first beginning, you need to consider how to tokenize the text. Luckily, transformers also provides tokenizers of pre-trained language models. The specific description of usage can be found in the corresponding documents, e.g. you can refer to the usage of BERT tokenizer here.

After that, all you need is to follow the process described in class. Maybe sometimes you need to write your own dataset for convenience, under which circumstance you will find this page is useful.

You can attempt to pre-train the Google BERT by designing a masked language model task just as we did in the class as well. Considering that HuggingFace provides an array of pre-trained language models, it is welcome to have a try with other models and compare the performance between them and BERT.

It is worth noting that we will choose the F1 score to evaluate the performance of the model. There is no doubt that a higher F1 score is better. Therefore, you are encouraged to enhance the performance of the model with any advanced technique.

In order to help you know better about your model's performance, We construct a simple 3-layers Bi-LSTM model without any further parameters adjustments as a weak baseline. After 50 epoch it can achieve:

Model	acc	precision	recall	F1
Bi-LSTM (3-layers)	74.57	62.97	74.18	68.12

Submission Requirements

You need to submit:

1. ***The best-performed model you saved*** after fine-tuning.
2. ***All*** of your python files and scripts used.
3. ***A report*** describing the methods and models your team adopted, the training procedure, final performances, and analysis.

Deadline: January 2, 2022 23:59.

E-mail: zyr527@sjtu.edu.cn

or jBox link: https://jbox.sjtu.edu.cn/v/link/view/d04eed45e6dc4c7f98ab502ec8dde8a1