# **登封造极团队方案 复现文档**

**赛题：**CCF2019 BDCI 金融信息负面及主体判定

**队伍**：登峰造极

**成员：**Chevalier，42Kjs，zhkkk，队友好棒棒，Wizare

如果有任何其他问题也可以直接联系我们：scut\_chengfeng@163.com ，[scut\_kongjunsheng@qq.com](mailto:scut_kongjunsheng@qq.com)，我们会积极配合。

## 一．代码运行环境

* python3 (最好是python3.6.7)
* pytorch-transformers 1.2.0
* torch 1.1.0
* tensorflow-gpu 1.12.0
* numpy 1.16.3
* tqdm 4.31.1
* scikit-learn 0.20.3
* pandas 0.24.2
* 系统: Linux version 4.4.0-138-generic (buildd@lcy01-amd64-006) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10) ) #164-Ubuntu SMP Tue Oct 2 17:16:02 UTC 2018
* CPU: Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz
* GPU: 4\*2080Ti 11G
* CUDA Version 10.0.130
* cudnn 7.5.0

## 二．方案概述：

1. 我们将本赛题简化成一个任务——基于金融实体的情感分析，在3种预训练模型上进行微调，包括3个不同种子的bert、1个在比赛数据集再次预训练的bert、1个roberta\_wwm\_ext模型。模型选型上考虑互补性，并考虑模型效率，只使用了base模型，使其更适合在真实环境中落地使用。
2. 数据预处理新颖，将样本转化为“["[CLS]"]文本["[SEP]"]实体n-10;...;<实体n>;...;实体n+10["[SEP]"]”的格式，使其能够考虑相邻实体之间相关性。
3. 预训练模型简洁有效，考虑到bert每一层能够捕捉到输入文本不同维度的特征，我们通过attention集成每一层的输出以得到更好的文本表示。
4. 同时我们通过Multi-Sample Dropout提升模型的泛化性。
5. 最终我们平均融合5个预训练模型，融合后的模型在线上已经能取得较好的成绩。
6. 另外，考虑到模型不能解决所有问题，因此我们在模型融合的基础上进行了后处理，提升了总体性能。

## 三．代码框架：

* datasets/: 存放原始的数据集，以及预处理后的数据集
  + preprocess\_round\_1\_2\_train\_data.csv: 对初赛和复赛合并之后的训练集进行预处理后的文件
  + preprocess\_round2\_test.csv: 对复赛测试集进行预处理后的文件
  + pretrain\_data.txt: 用于further pretrain的训练文件
  + round\_1\_2\_train\_data.csv: 初赛和复赛合并之后的训练集
  + round2\_test.csv: 复赛提交的测试集
  + Round2\_train.csv: 复赛提供的训练集
  + Train\_Data.csv: 初赛提供的训练集
* transformers: 用于将tensorflow的预训练权重转换为pytorch的预训练权重
* Further\_pretraining/: 根据现有数据集，对bert模型进行pretrain训练
* pretrain\_weight/: 预训练模型权重
  + bert
  + bert\_further\_pretrain
  + roberta\_wwm\_ext
* model\_save/:模型训练后的权重存放
* log/: 日志存放
* Fusion\_model.py: 模型融合脚本
* model\_1\_bert\_att\_drop\_42.py: 在bert模型的基础上，添加attention和dropout层作为整体训练模型，以随机种子为42进行训练
* model\_2\_bert\_att\_drop\_further\_pretrain.py: 先根据现有数据集对bert模型进行further pretrain，得到新bert的模型权重。在bert模型的基础上，添加attention和dropout层作为整体训练模型
* model\_3\_roberte\_wwm\_ext\_att\_drop\_42.py: 在roberte\_wwm\_ext模型的基础上，添加attention和dropout层作为整体训练模型，以随机种子为42进行训练
* model\_4\_bert\_att\_drop\_420.py: 在bert模型的基础上，添加attention和dropout层作为整体训练模型，以随机种子为420进行训练
* model\_5\_bert\_att\_drop\_1001001.py: 在bert模型的基础上，添加attention和dropout层作为整体训练模型，以随机种子为1001001进行训练
* predict\_model\_1\_bert\_att\_drop\_42.py: 无须训练，加载最优模型直接预测
* predict\_model\_2\_bert\_att\_drop\_further\_pretrain.py: 无须训练，加载最优模型直接预测
* predict\_model\_3\_roberte\_wwm\_ext\_att\_drop\_42.py: 无须训练，加载最优模型直接预测
* predict\_model\_4\_bert\_att\_drop\_420.py: 无须训练，加载最优模型直接预测
* predict\_model\_5\_bert\_att\_drop\_1001001.py: 无须训练，加载最优模型直接预测
* preprocess.py: 数据预处理脚本
* postprocess.py: 模型预测结果后处理脚本

## 四.复现：

我们提供三种方式复现：（1）使用我们之前得到的概率文件，进行融合（2）从头开始训练模型复现我们的结果（3）使用我们的模型文件复现我们的结果

### （1）One Step:

\* 因为训练模型比较久而且模型比较大，所以我们提供了所有模型对OOF和测试集的预测结果(./submit/train\_prob和./submit/test\_prob)，只需要简单的做一下概率平均,然后运行一下后处理就可以得到我们提交的最好结果。

执行以下脚本：

```

python Fusion\_model.py

python postprocess.py

```

最后生成的./submit/best\_result.csv即可用于提交。

\* 当然如果想要从头复现，可以看下面的说明：

### （2）step by step：

**1. 预处理：**

\* 该文件为预处理文件，主要进行以下几个预处理：

1.清除无用的信息

2.如果待预测实体不在文本的前512中，将预测实体所在的文本提前到前512中

3.将文本中出现的实体，添加上“<”，“>”，来突出实体

4.将含有多条实体的数据切分成多条只预测一个实体的数据

5.截断文本（取前512）

得到"./datasets/preprocess\_round\_1\_2\_train\_data.csv"和"preprocess\_round2\_test.csv"

这里我们使用的是初赛和复赛合并之后的训练集数据集，完全复现请使用合并后的数据集（"./datasets/round\_1\_2\_train\_data.csv"）。

```

python preprocess.py

```

如果是使用新数据集（更改对应参数），使用以下：

```

python preprocess.py ./datasets/round\_1\_2\_train\_data.csv ./datasets/round2\_test.csv

```

**2. 预训练权重**

**\*Ps: 如果嫌检查预训练权重麻烦，可以跳过该步骤，我们已经提供了pytorch版本的bert权重、再次预训练的bert权重、roberta\_wwm\_ext权重**

"./pretrain\_weight"下有三个预训练权重：

1. bert-base（2）roberta\_wwm\_ext（3）bert\_further\_pretrain。

我们已经放在该文件下，文件来源如下：

1.[BERT-Base, Chinese](<https://github.com/google-research/bert#pre-trained-models>)，这里只提供tensorflow版本，还需转换成pytorch版本。

2.[roberta\_wwm\_ext](<https://github.com/ymcui/Chinese-BERT-wwm>)，通过讯飞云下载pytoch版本。

3.bert\_further\_pretrain，其中bert\_further\_pretrain预训练权重为bert-base通过在该比赛数据集再次预训练得到。由于训练时间比较长，我们提供已经further-pretrain好的权重供下载。

\* 如果你想Ferther pretrain Bert, 可以执行一下脚本：

```

sh ./shell/get\_pretrain\_data.sh

sh ./shell/run\_pretrain.sh

```

\*Ps:从官网下载的BERT-Base, Chinese和通过脚本再次预训练得到的bert-base-further-pretrain，得到的是tensorflow的权重，还需要转换为pytorch的bert权重，可以执行以下脚本或者参考[tensorflow-bert权重转pytorch](<https://www.lizenghai.com/archives/32772.html>)

```

cd transformers

export BERT\_BASE\_DIR=#tensorflow权重的绝对路径#

python convert\_bert\_original\_tf\_checkpoint\_to\_pytorch.py --tf\_checkpoint\_path $BERT\_BASE\_DIR/bert\_model.ckpt --bert\_config\_file $BERT\_BASE\_DIR/bert\_config.json --pytorch\_dump\_path $BERT\_BASE\_DIR/pytorch\_model.bin

```

Ps:还需要把bert\_config.json文件重命名为config.json

**3. 模型训练**

\* 该模块是主要的模型训练及模型在测试集上的预测。

\* 模型采用七折交叉训练。

\* 首先需要从百度云下载预训练权重，copy到"./pretrain\_weight/"下

\* 执行脚本训练模型，每个模型训练的时间在15个小时左右。

\* 各个模型的权重在训练完后将保存在"./model\_save"下，概率文件将保存在"./submit/train\_prob"和"./submit/test\_prob"下。

依次执行代码训练五个模型如下：

```

python model\_1\_bert\_att\_drop\_42.py

python model\_2\_bert\_att\_drop\_further\_pretrain.py

python model\_3\_roberte\_wwm\_ext\_att\_drop\_42.py.py

python model\_4\_bert\_att\_drop\_420.py

python model\_5\_bert\_att\_drop\_1001001.py

```

Ps:如果GPU指定报错，在脚本中可以修改GPU参数

Ps:如果嫌模型训练时间过长，可执行以下代码直接预测

```

python predict\_model\_1\_bert\_att\_drop\_42.py

python predict\_model\_2\_bert\_att\_drop\_further\_pretrain.py

python predict\_model\_3\_roberte\_wwm\_ext\_att\_drop\_42.py.py

python predict\_model\_4\_bert\_att\_drop\_420.py

python predict\_model\_5\_bert\_att\_drop\_1001001.py

```

**4.模型融合**

该模块将五个模型的概率文件平均融合。该结果在线上已经能取得一个不错的成绩。

```

python Fusion\_model.py

```

**5.后处理**

\* 该模块主要根据训练集中的一些实体共现频率提取的规则，处理了下并列实体的情况，以及根据训练集的先验知识，补充部分短实体。

\* 运行得到最终的提交文件best\_result.csv。

```

python postprocess.py

```

**6.提交：**

\* 在submit目录下, 提交best\_result.csv。

**Concat:**

scut\_chengfeng@163.com