第一次作业　选做 2：古文分类

1 任务描述

1.1 任务简介

本实验为散文-韵文二分类任务，主要为中文古文的散文和韵文的分类。本实验首先对文本进行预处理，之后采用预训练语言模型进行分类识别。

1.2 数据集

韵文数据集：https://github.com/Werneror/Poetry

散文数据集：https://disk.pku.edu.cn:443/link/94FA2F534E12B25FDD7B1F7F4B8F37F3

为减小训练时间，抽取散文、韵文各 30000 篇作为训练集。

选取了不同时代的韵文以提取共性特征。其中包括韵文数据为：训练集 3w 篇；测试集 3000 篇 ； 验证集 3000 篇。

朝代信息如下表所示：

朝代	唐	宋	近现代	总计
训练集	10000	10000	10000	30000
测试集	1000	1000	1000	3000
验证集	1000	1000	1000	3000

2 实验内容

2.1 数据预处理

韵文：

只有逗号（,）和句号（.），无其他任何特殊符号。此外还含有作者、朝代无关信息，预处理时直接删除即可。

示例：

「二城朝霁。秋阴杳，纤尘犹湿空翠。九华初服未登临，况试题糕字。谢羽客，相呼异地，云间归路殊迢递。纵彩笔堪传，锦袖冷，红笺削减，赏心潜替。候馆物序萧条，云浆谁酿，百虑惟此能制。枕衾不惯起频看，照影惊憔悴。四壁悄，浑如梦寐。年来怕见终飘坠。但晓夕、成流水，霜结疏棂，旧怀犹是。」

文件格式：csv，以朝代为单位分散在不同的 csv 文件中。**预处理时，随机抽取不同朝代的文章数据，并将其合并在一个文件中**。

散文：

有《》“”、等特殊符号。**特殊符号不应成为区分两者的标志，为控制实验变量，去掉上述特殊符号，和韵文保持一致，只留下逗号和句号**。

示例：

处理前：

「兩漢紀上漢紀
孝成皇帝紀二卷第二十五
三年春正月，〈楚王囂〉來朝。詔曰：“〔〈囂〉〕孝弟仁慈，　在國二十餘年，纖介之過未嘗聞。《書》不云乎‘用德彰厥善’。其封〈囂〉子〈勳〉爲〈廣戚侯〉。”二月丙戌，〈犍爲〉地震，山崩，擁〈江〉水逆流。　秋八月乙卯晦，日蝕。光禄大夫〈劉向〉校中祕」.....

处理后:
「兩漢紀上漢紀
孝成皇帝紀二卷第二十五
三年春正月，楚王囂來朝。詔曰：囂孝弟仁慈，　在國二十餘年，纖介之過未嘗聞。書不云乎用德彰厥善。其封囂子勳爲廣戚侯。二月丙戌，犍爲地震，山崩，擁江水逆流。　秋八月乙卯晦，日蝕。光禄大夫劉向校中祕」.....

文件格式：txt 文件，都集成在一个 txt 文件中，随机抽取即可。

2.2 模型选择
本实验参考代码：
GitHub 中文文本分类项目
链接：https://github.com/649453932/Bert-Chinese-Text-Classification-Pytorch
采用 bert，ERNIE 两种主流的预训练模型进行中文文本分类，具体模型应用如下表所示：

模型	特点
bert	采用 bert_chinese 预训练模型
ERNIE	ERNIE 是词汇级别的，在常见的中文处理任务上效果更优。
bert_CNN	bert 作为 Embedding 层，接入三种卷积核的 CNN。
bert_RNN	bert 作为 Embedding 层，接入 LSTM，用 RNN 代替全连接层进行预测，提升了长文本的记忆能力。
bert_RCNN	bert 作为 Embedding 层，通过 LSTM 与 bert 输出拼接，经过一层最大池化层。
bert_DPCNN	bert 作为 Embedding 层，经过一个包含三个不同卷积特征提取器的 region embedding 层，可以看作输出的是 embedding，然后经过两层的等长卷积来为接下来的特征抽取提供更宽的感受眼，（提高 embdding 的丰富性），然后会重复通过一个 1/2 池化的残差块，1/2 池化不断提高词位的语义，其中固定了 feature_maps,残差网络的引入是为了解决在训练的过程中梯度消失和梯度爆炸的问题。。

2.3 实验结果

BERT：

Precision, Recall and F1-Score...

	precision	recall	f1-score	support
yunwen	0.9998	1.0000	0.9999	8997
sanwen	1.0000	0.9993	0.9997	2978
accuracy			0.9998	11975
macro avg	0.9999	0.9997	0.9998	11975
weighted avg	0.9998	0.9998	0.9998	11975

ERNIE：

	precision	recall	f1-score	support
yunwen	1.0000	0.9999	0.9999	8997
sanwen	0.9997	1.0000	0.9998	2978
accuracy			0.9999	11975
macro avg	0.9998	0.9999	0.9999	11975
weighted avg	0.9999	0.9999	0.9999	11975

bert_CNN：

	precision	recall	f1-score	support
yunwen	0.9999	0.9997	0.9998	8997
sanwen	0.9990	0.9997	0.9993	2978
accuracy			0.9997	11975
macro avg	0.9994	0.9997	0.9996	11975
weighted avg	0.9997	0.9997	0.9997	11975

bert_RNN:

	precision	recall	f1-score	support
yunwen	1.0000	0.9996	0.9998	8997
sanwen	0.9987	1.0000	0.9993	2978
accuracy			0.9997	11975
macro avg	0.9993	0.9998	0.9996	11975
weighted avg	0.9997	0.9997	0.9997	11975

bert_RCNN:

	precision	recall	f1-score	support
yunwen	1.0000	0.9999	0.9999	8997
sanwen	0.9997	1.0000	0.9998	2978
accuracy			0.9999	11975
macro avg	0.9998	0.9999	0.9999	11975
weighted avg	0.9999	0.9999	0.9999	11975

bert_DPCNN:

	precision	recall	f1-score	support
yunwen	0.9956	1.0000	0.9978	8997
sanwen	1.0000	0.9866	0.9932	2978
accuracy			0.9967	11975
macro avg	0.9978	0.9933	0.9955	11975
weighted avg	0.9967	0.9967	0.9967	11975

2.3 结果分析

可以看到，进过预处理后的中文文本分类取得了非常好的分类效果，仅仅是基础的 bert 模型，准确率和召回率就已经达到 99.98%以上。其中 ERNIE 和 bert_CNN 预训练模型表现最好，最终加权精度甚至达到了 99.99%。如下表：

	precision	recall	f1-score
bert	0.9998	0.9998	0.9998
ERNIE	**0.9999**	**0.9999**	**0.9999**
bert_CNN	0.9997	0.9997	0.9997
bert_RNN	0.9997	0.9997	0.9997
bert_RCNN	**0.9999**	**0.9999**	**0.9999**
bert_DPCNN	0.9967	0.9967	0.9967

在实验过程中，遇到如下问题，及其解决思路：

1、数据格式不一致。散文为 txt 文件，韵文为 csv 文件，都通过导入到 python 程序中进行统一处理并输出 csv 文件，再另存为 txt 文件。

2、数据编码规范。另存为 txt 文件时，统一采用 utf-8 编码格式。

3、散文和韵文数据未打乱，造成学习结果差。df.sample(frac=1.0)可以将数据打乱。

实验代码及处理后的数据集已提交至：https://github.com/Sprinkle0/NLTK-Learn

欢迎各位批评指正!