课程设计报告

唐倩倩，语言学及应用语言学，201921198622

一、数据处理

1、数据初步处理：

首先对两个数据集进行以下处理：

第一步：遍历数据集，获取每个样例的rating值和文本；

第二步：将rating值处理为0-7的分类标签；

第三步：剔除文本中
；

第四步：使用NLTK对文本进行tokenize和Lemmatize (词形还原)；

第五步：将处理好的标签和文本按照以下格式存入Dataframe中（共两列，第一列为标签，第二列为文本）：

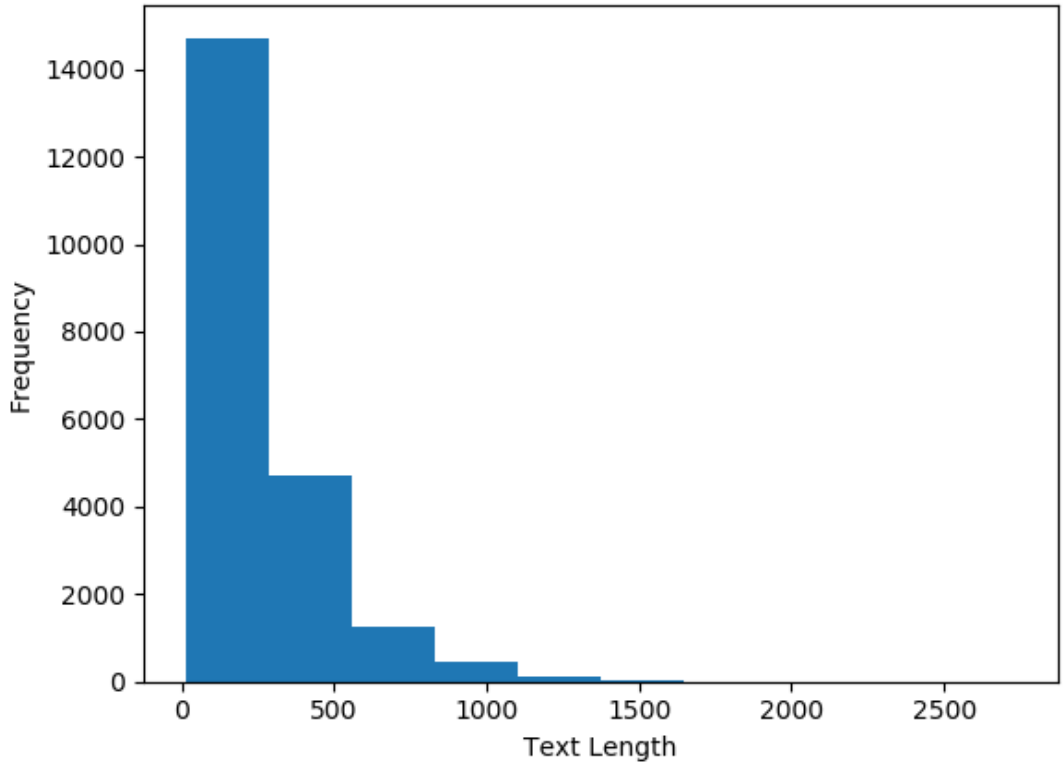
	labels	text
0	4	The movie never claim to be something spectacu...
1	6	Just the kind of movie I love . Some very good...
2	7	That 70 Show be the best TV show ever , period...
3	2	Basically , `` Caprica '' be the Cylon origin ...
4	0	I do n't believe there ha ever be a more evil ...
5	7	This be one of the most brilliant movie that I...
6	2	This movie wa a real torture fest to sit throu...
7	6	This film remind me of 42nd Street star Bebe D...
8	0	With title like this you know you get pretty m...
9	5	It ha be say that Deanna Durbin invent teenage...
10	1	If 1977 's `` Exorcist II : The Heretic '' do ...

第六步：从训练集中分出15%的数据作为验证集。

第七步：将处理好的训练集、验证集、测试集数据分别输出到csv文件中，并储存在\data目录下，以便后续使用。

2、数据处理对比实验

统计训练集中文本的长度和长度所出现的频次并画图，结果如下：



根据统计结果可知，数据中存在大量长度过长(>512)的文本，由于后续要用到的预训练模型BERT、Roberta、XLNet对输入数据的长度有限制，因此决定尝试对文本进行hierarchical分割，看结果是否会有提升。

同时，我还将尝试用多分类数据进行训练，再将预测得到的标签转化成二分类标签，以此来实现二分类，看看结果是否会比直接使用二分类数据进行训练更好。

因此，我对数据进行了进一步处理，得到了以下四种数据：

A. 多分类未分割数据 (_multi_raw.csv)

B. 多分类分割数据 (_multi_split.csv)

C. 二分类未分割数据 (_bi_raw.csv)

D. 二分类分割数据 (_bi_split.csv)

（1）数据进一步处理

第一步：Hierarchical分割

Hierarchical数据分割本质上就是对数据进行有重叠(overlap)的分割，这样一来分割后的短文本之间仍然保留了一定的关联信息。

A. 首先将过长的文本切分为有重叠的短文本，在本次实验中，设置的分割长度为300, overlap长度为50.

```
#将过长的文本分割成短文本(有重叠的分割)
def split_text(text, split_len, overlap_len):
    text_piece=[]
    tokens = text.split(" ")
    num = len(tokens)//split_len
    window = split_len - overlap_len
```

```
    for i in range(num):
        if i == 0: #第一次分割,直接按分割长度取文本
            piece = tokens[:split_len]
        else: # 否则, 往回退overlap长度后继续往后按分割长度取文本
            piece = tokens[(i * window ):(i * window + split_len)]
        text_piece.append(piece)
    last_piece = tokens[(num * window - 1):]
    if last_piece:
        text_piece.append(last_piece)
    return text_piece
```

B. 接着给分割后的短文本加上原文本的标签，按照上文提过的格式储存到csv文件中以备后续实验（第一列为标签，第二列为文本内容）。

通过对初步数据处理获得的**多分类未分割数据(multi_raw.csv)**进行Hierarchical分割，我们得到了**多分类分割数据(multi_split.csv)**。

第二步：将多分类数据转化成二分类数据

在多分类的数据中，0-3的标签代表为负例，4-7的标签代表为正例。根据这个规则，我们可以将多分类数据转化成标签为0和1的二分类数据，代码如下：

```
# 将多分类数据处理成二分类数据
def get_binary_data(path,type,mode):
    df = pd.read_csv(path,skip_blank_lines = True)
    for index,row in df.iterrows():
        if row["labels"] > 3:
            df.at[index,"labels"] = 1
        else:
            df.at[index, "labels"] = 0
    df.to_csv(f"data/{type}_bi_{mode}.csv",index=None)
    return df
```

通过处理多分类未分割数据 (*multi_raw.csv)和多分类分割数据 (*multi_split.csv)，可以得到二分类未分割数据 (*bi_raw.csv)和二分类分割数据 (*bi_split.csv)。

```
#将多分类未分割数据转化成二分类未分割数据
get_binary_data("data/train_multi_raw.csv","train","raw")
get_binary_data("data/test_multi_raw.csv","test","raw")
get_binary_data("data/validation_multi_raw.csv","validation","raw")

#将多分类分割数据转化成二分类分割数据
get_binary_data("data/train_multi_split.csv","train","split")
get_binary_data("data/test_multi_split.csv","test","split")
get_binary_data("data/validation_multi_split.csv","validation","split")
```

（2）对比实验结果

所有数据的实验都是在roberta-large上进行的，具体预训练模型的选择会在下面详细介绍。使用四种数据进行实验时，模型的参数保持一致，具体如下：

```
model_args = {
    'output_dir': 'roberta-roberta-large-outputs',
    'max_seq_length': 512,
    'num_train_epochs': 3,
```

```
    'train_batch_size': 16,
    'eval_batch_size': 16,
    'gradient_accumulation_steps': 1,
    'learning_rate': 4e-5,
    'save_steps': 3000,
    "evaluate_during_training": True,
    "evaluate_during_training_steps": 3000,
    'reprocess_input_data': True,
    "save_model_every_epoch": True,
    'overwrite_output_dir': True,
    'no_cache': True,
    'use_early_stopping': True,
    'early_stopping_patience': 3,
    'manual_seed': 1,
    'n_gpu': 8
}
```

最终在验证集上的结果如下：

data	accuracy	f1 score
multi_raw	0.951	0.951
multi_split	0.947	0.947
bi_raw	0.957	0.958
bi_split	0.955	0.953

根据结果可知，在当前参数下，二分类数据呈现出比多分类数据更好的结果，使用未分割的数据也比分割过的数据有更好的效果，因此，本次实验最终使用**二分类未分割数据**来进行最终的结果评估。

二、模型设计

1、预训练模型的选取

Table 1. Performance of deep learning based text classification models on sentiment analysis datasets (in terms of classification accuracy), evaluated on the IMDB, SST, Yelp, and Amazon datasets. Italic indicates the non-deep-learning models.

Method	IMDB	SST-2	Amazon-2	Amazon-5	Yelp-2	Yelp-5
Naive Bayes [157]	-	81.8	-	-	-	-
LDA [189]	67.4	-	-	-	-	-
BoW+SVM [12]	87.8	-	-	-	-	-
tf.Δ idf [190]	88.1	-	-	-	-	-
Char-level CNN [31]	-	-	94.49	59.46	95.12	62.05
Deep Pyramid CNN [30]	-	84.46	96.68	65.82	97.36	69.40
ULMFiT [191]	95.4	-	-	-	97.84	70.02
BLSTM-2DCNN [22]	-	89.5	-	-	-	-
Neural Semantic Encoder [76]	-	89.7	-	-	-	-
BCN+Char+CoVe [192]	91.8	90.3	-	-	-	-
GLUE ELMo baseline [193]	-	90.4	-	-	-	-
BERT ELMo baseline [4]	-	90.4	-	-	-	-
CCCapsNet [57]	-	-	94.96	60.95	96.48	65.85
Virtual adversarial training [145]	94.1	-	-	-	-	-
Block-sparse LSTM [194]	94.99	93.2	-	-	96.73	
BERT-base [4, 90]	95.63	93.5	96.04	61.6	98.08	70.58
BERT-large [4, 90]	95.79	94.9	96.07	62.2	98.19	71.38
ALBERT [86]	-	95.2	-	-	-	-
Multi-Task DNN [92]	83.2	95.6	-	-	-	-
Snorkel MeTaL [195]	-	96.2	-	-	-	-
BERT Finetune + UDA [196]	95.8		96.5	62.88	97.95	62.92
RoBERTa (+additional data) [85]	-	96.4	-	-	-	-
XLNet-Large (ensemble) [5]	96.21	96.8	97.6	67.74	98.45	72.2

参考一篇名为“Deep Learning Based Text Classification: A Comprehensive Review”这篇综述中使用预训练模型进行情感分析的结果(Table1)以及前人的一些经验，本次实验决定选择以下三个预训练模型来进行对比实验：

(1) bert-large-cased

(2) xlnet-large-cased

(3) roberta-large

在对比实验的过程中，三个模型所使用的参数保持一致，具体如下：

```
model_args = {
    'output_dir': f'{model_type}-{model_name}-outputs',
    'max_seq_length': 350,
    'num_train_epochs': 3,
    'train_batch_size': 16,
    'eval_batch_size': 16,
    'gradient_accumulation_steps': 1,
    'learning_rate': 4e-5,
    'save_steps': 3000,
    "evaluate_during_training": True,
    "evaluate_during_training_steps": 3000,
    'reprocess_input_data': True,
    "save_model_every_epoch": True,
    'overwrite_output_dir': True,
    'no_cache': True,
    'use_early_stopping': True,
    'early_stopping_patience': 3,
    'manual_seed': 1,
    'n_gpu': 8
}
```

在验证集上得到的结果如下表：

Baseline	accuracy	f1 score
bert-large-cased	0.934	0.935
xlnet-large-cased	0.945	0.945
roberta-large	0.950	0.951

可见，roberta-large在三个模型中表现最佳，因此本次实验选取**roberta-large**作为最终的预训练模型。

三、最终结果评估

根据上文对数据和模型的结果对比，本实验最终选取**二分类未分割测试集**数据在 **roberta-large**上进行最终结果的评估，最终结果如下：

```
>>> print("accuracy", metrics.accuracy_score(bi_targets,bi_preds))
accuracy 0.96336
>>> print("recall", metrics.recall_score(bi_targets,bi_preds))
recall 0.96696
>>> print("f1 score", metrics.f1_score(bi_targets,bi_preds))
f1 score 0.9634914308489437
```

accuracy: 0.963

f1 score: 0.963