NLP第四次大作业实验报告

陈远腾 2020k8009929041

一、选题

5.利用公开的情感或情绪分析评测语料，对比分析统计学习方法和神经网络方法的性能差异，并进行错误分析。

二、环境搭建

1.容器环境准备

由于我们希望训练基于transformer的文本分类模型，对显存要求较高，本机的GPU无法满足，因此我在autodl平台上租赁了4 x RTX3090 的服务器容器，作为本次实验的平台。（实际上是沿用了第三次大作业所使用的环境）

2.依赖库与相关工具安装

本次实验在神经网络方法方面，我使用hugging face提供的transformer框架：

链接：<GitHub - huggingface/transformers: 🤗 Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX.>

简要介绍（具体内容请见以上仓库的说明文档）：

Hugging Face Transformers是一个用于自然语言处理（NLP）的开源库，旨在提供一种简单而快速的方式来构建、训练和部署各种NLP模型。该库的核心是使用PyTorch和TensorFlow 2.0实现的预训练模型，这些模型可以用来进行各种NLP任务，例如文本分类、命名实体识别、语言翻译和问答等。

Transformers库提供了一个高级别的API，使得使用这些预训练模型变得异常简单，同时也提供了低级别的API，使得用户可以对模型进行更加细粒度的控制。在使用预训练模型时，用户可以选择使用已经训练好的模型，也可以通过微调已有的模型或使用自己的数据进行训练。

除了预训练模型之外，Hugging Face Transformers还提供了各种NLP任务的数据集和评估指标，这些可以帮助用户更好地理解和评估他们的模型在特定任务上的表现。

环境搭建命令：

```
# 创建新的虚拟环境bert
conda create -n bert python=3.8
# 激活虚拟环境
conda activate bert
# 安装pytorch（我的服务器上CUDA版本为11.3，如其他版本请参考Pytorch官网进行安装）
conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1
cudatoolkit=11.3 -c pytorch
# 安装transformers库方法1（pip）
pip install transformers
# 安装transformers库方法2（conda）
conda install -c huggingface transformers

# 安装数据集所需库datasets
pip install datasets
# 安装评估库evaluate
pip install evaluate
# 升级accelerate库
pip install --upgrade accelerate

# 安装传统方法所用的sklearn库
pip install -U scikit-learn
```

三、实验过程

1.数据集准备

```
from datasets import load_dataset
dataset = load_dataset("yelp_review_full")
```

这里我们直接使用Hugging Face Datasets库中的`load_dataset`方法加载Yelp Review Full数据集：

`yelp_review_full`数据集是一个关于餐厅评论的数据集，包含来自Yelp网站的超过500,000条英文评论。这些评论被分为五个类别（1-5星），其中1星表示非常差，5星表示非常好。数据集中的评论具有不同长度和主题，包括食物、服务、价格等。这个数据集通常用于文本分类和情感分析任务。加载后返回一个字典对象，其中包含“train”、“test”和“validation”三个键，每个键对应着一个数据集分割。每个分割都有“text”和“label”两个字段，其中“text”字段包含评论文本，而“label”字段包含评论的星级评分。

这里我们截取训练集的前三条文本及其对应标签展示如下：

```
(open-mmlab) C:\Users\cyt>python
Python 3.7.16 (default, Jan 17 2023, 16:06:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from datasets import load_dataset
>>> dataset = load_dataset("yelp_review_full")
Using the latest cached version of the module from C:\Users\cyt\.cache\huggingface\modules\datasets_modules\datasets\yel
p_review_full\e8e18e19d7be9e75642fc66b198abadb116f73599ec89a69ba5dd8d1e57ba0bf (last modified on Wed Jun 14 15:03:26 202
3) since it couldn't be found locally at yelp_review_full., or remotely on the Hugging Face Hub.
Found cached dataset yelp_review_full (C:/Users/cyt/.cache/huggingface/datasets/yelp_review_full/yelp_review_full/1.0.0/
e8e18e19d7be9e75642fc66b198abadb116f73599ec89a69ba5dd8d1e57ba0bf)
100%|██████████████████████████████████████████████████████████████████████████████| 2/2 [00:00<00:00, 13.29it/s]
>>> dataset['train'][0]
{'label': 4, 'text': "dr. goldberg offers everything i look for in a general practitioner.  he's nice and easy to talk t
o without being patronizing; he's always on time in seeing his patients; he's affiliated with a top-notch hospital (nyu)
 which my parents have explained to me is very important in case something happens and you need surgery; and you can get
 referrals to see specialists without having to see him first.  really, what more do you need?  i'm sitting here trying
to think of any complaints i have about him, but i'm really drawing a blank."}
>>> dataset['train'][1]
{'label': 1, 'text': "Unfortunately, the frustration of being Dr. Goldberg's patient is a repeat of the experience I've
had with so many other doctors in NYC -- good doctor, terrible staff.  It seems that his staff simply never answers the
phone.  It usually takes 2 hours of repeated calling to get an answer.  Who has time for that or wants to deal with it?
 I have run into this problem with many other doctors and I just don't get it.  You have office workers, you have patien
ts with medical needs, why isn't anyone answering the phone?  It's incomprehensible and not work the aggravation.  It's
with regret that I feel that I have to give Dr. Goldberg 2 stars."}
>>> dataset['train'][2]
{'label': 3, 'text': "Been going to Dr. Goldberg for over 10 years. I think I was one of his 1st patients when he starte
d at MHMG. He's been great over the years and is really all about the big picture. It is because of him, not my now form
er gyn Dr. Markoff, that I found out I have fibroids. He explores all options with you and is very patient and understan
ding. He doesn't judge and asks all the right questions. Very thorough and wants to be kept in the loop on every aspect
```

可以看到，训练集的前三条文本的标签分别为4,1,3。

> 后续我们将基于Yelp Review Full数据集训练五分类的文本分类模型。

2.数据预处理

与之前的任务相同，在训练模型前，首先需要对原始数据集做数据预处理，例如分词：

这里我们直接使用transformers库提供的AutoTokenizer工具对原始数据集进行分词处理，选取的训练好的分词模型为hugging face提供的 "bert-base-cased"。

```python
# 导入AutoTokenizer库
from transformers import AutoTokenizer
# 加载预训练分词模型tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

# 定义分词函数，为后续map做准备
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True)

# 对原始数据集进行分词预处理
tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

分词后效果如下：

```
[7]  tokenized_datasets["train"]                                                    Python

...  Dataset({
         features: ['label', 'text', 'input_ids', 'token_type_ids', 'attention_mask'],
         num_rows: 650000
     })

[8]  tokenized_datasets["test"]                                                     Python

...  Dataset({
         features: ['label', 'text', 'input_ids', 'token_type_ids', 'attention_mask'],
         num_rows: 50000
     })
```

可以看到，训练集上有650000条文本数据，而测试集上有50000条文本数据。

由于我们的计算资源有限，因此这里只截取训练集上前10000条文本作为我们的训练集，截取测试集上前2000条文本作为我们的测试集：

```
small_train_dataset =
tokenized_datasets["train"].shuffle(seed=42).select(range(10000))
small_eval_dataset =
tokenized_datasets["test"].shuffle(seed=42).select(range(2000))
```

3.传统方法

(1)基于决策树的文本分类

基于传统方法的文本分类，我们首先想到了决策树的方法：

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from datasets import Dataset

# 获取训练集和测试集的特征和标签
train_features = small_train_dataset['text']
train_labels = small_train_dataset['label']
test_features = small_eval_dataset['text']
test_labels = small_eval_dataset['label']

# 特征提取
vectorizer = CountVectorizer(max_features=1000)
train_features_vectorized = vectorizer.fit_transform(train_features)
test_features_vectorized = vectorizer.transform(test_features)

# 训练决策树模型
clf = DecisionTreeClassifier()
clf.fit(train_features_vectorized, train_labels)

# 在测试集上进行预测并评估模型性能
pred_labels = clf.predict(test_features_vectorized)
accuracy = accuracy_score(test_labels, pred_labels)
report = classification_report(test_labels, pred_labels)
print('Accuracy:', accuracy)
print('Classification report:\n', report)
```

分类结果及错误分析如下：

```
Accuracy: 0.3381
Classification report:
              precision    recall  f1-score   support

           0       0.45      0.44      0.45      4049
           1       0.28      0.28      0.28      3950
           2       0.27      0.27      0.27      3983
           3       0.29      0.29      0.29      4038
           4       0.40      0.40      0.40      3980

    accuracy                           0.34     20000
   macro avg       0.34      0.34      0.34     20000
weighted avg       0.34      0.34      0.34     20000
```

可以看到使用决策树的方法进行文本分类的准确率非常低，只有33.81%，在错误分析中，我们发现：

- 对于较极端的评价文本(label = 0 或 label = 4)，模型的准确率和召回率都在40%以上，因为这些文本的情感比较强烈，比较容易辨识。
- 而对于较中立的评价文本(label = 1 或 label = 2 或 label = 3)，模型的准确率和召回率都在30%以下，因为这个文本的情感较模糊，不易辨识。

(2)基于贝叶斯的文本分类

其次，我想到了使用贝叶斯分类模型：

```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# 获取训练集和测试集的特征和标签
train_features = small_train_dataset['text']
train_labels = small_train_dataset['label']
test_features = small_eval_dataset['text']
test_labels = small_eval_dataset['label']

# 特征提取
vectorizer = CountVectorizer(max_features=1000)
train_features_vectorized = vectorizer.fit_transform(train_features)
test_features_vectorized = vectorizer.transform(test_features)

# 训练朴素贝叶斯模型
clf = MultinomialNB()
clf.fit(train_features_vectorized, train_labels)

# 在测试集上进行预测并评估模型性能
pred_labels = clf.predict(test_features_vectorized)
accuracy = accuracy_score(test_labels, pred_labels)
report = classification_report(test_labels, pred_labels)
print('Accuracy:', accuracy)
print('Classification report:\n', report)
```

分类结果及错误分析如下：

```
Accuracy: 0.5022
Classification report:
              precision    recall  f1-score   support

           0       0.60      0.66      0.63      4049
           1       0.42      0.34      0.38      3950
           2       0.42      0.43      0.42      3983
           3       0.45      0.45      0.45      4038
           4       0.58      0.63      0.60      3980

    accuracy                           0.50     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       0.50      0.50      0.50     20000
```

可以看到使用贝叶斯的方法进行文本分类的准确率较决策树而言有一定提升，达到了50.22%，在错误分析中，我们发现现象与决策树比较相似：

- 对于较极端的评价文本(label = 0 或 label = 4)，模型的准确率和召回率都在60%左右，因为这些文本的情感比较强烈，比较容易辨识。
- 而对于较中立的评价文本(label = 1 或 label = 2 或 label = 3)，模型的准确率和召回率都在40%左右，因为这个文本的情感较模糊，不易辨识。

4.神经网络方法

神经网络方法方面，我以hugging face提供的Bert预训练模型为基础，在Yelp Review Full数据集上进行fine-tune训练五分类的文本分类模型：

- 预训练模型加载：

```
from transformers import AutoModelForSequenceClassification
model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased", num_labels=5)
```

- 优化器：Adam
- 评估标准：loss + accuracy
- fine-tune轮数：epoch = 10

训练结果：

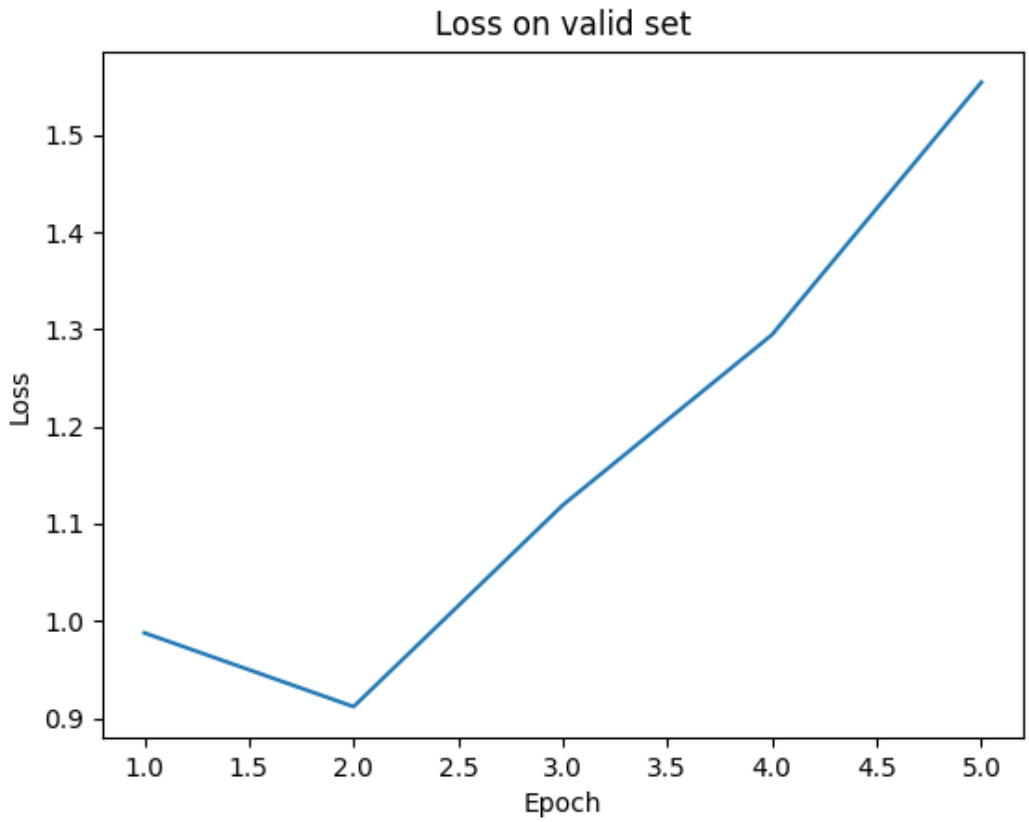
```
trainer.train()
```

```
/root/autodl-tmp/conda/envs/bert/lib/python3.8/site-packages/transformers/optimization.py:391: FutureWarning: This implementation of AdamW is deprecated and will be re
  warnings.warn(
/root/autodl-tmp/conda/envs/bert/lib/python3.8/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather along dimension 0, but all input tens
  warnings.warn('Was asked to gather along dimension 0, but all '
```

[3130/3130 43:41, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	0.987765	0.563550
2	0.963300	0.911858	0.602000
3	0.963300	1.119259	0.599500
4	0.564700	1.294533	0.594700
5	0.265800	1.554174	0.591950
6	0.265800	1.823240	0.605750
7	0.131700	2.107365	0.598950
8	0.060500	2.393476	0.603450
9	0.060500	2.584165	0.601500
10	0.022200	2.614256	0.604250

测试集上的loss曲线如下：



可以看到，在第2个epoch处模型在测试集上的loss就基本到达了最低，准确率达到60%左右。第二个epoch后出现了过拟合现象，在测试集上的loss逐渐升高。

> 可以看到经过fine-tune，模型在测试集上的准确率由56%提升到了60%左右。表现明显好于传统方法(决策树和贝叶斯分类)。

四、结果对比

根据以上实验结果，我们将传统方法和神经网络方法的结果左以下对比：

分类方法	准确率	rank
决策树	33.81%	3
贝叶斯	50.22%	2
Bert预训练 + fine-tune	60.20%	1

可以看到，基于神经网络的文本分类模型获得了最好的表现。