

自然语言期中作业：IMDB电影评论文本分类

一、数据预处理

1.data_processing.py

原始数据是爬虫得到的初步结果，里面包含了HTML标签和URL等与文本情感无关的噪音，需要进行数据清洗，提取出其中的IMDB影评内容。

之后再进行统一单词大小写、去除停用词以及标点符号。同时还需要对原始数据集的标签进行修改。positive改为1，negative改为0。

上述的过程在data_processing.py 中实现，代码如下，结合代码对上述过程进行具体说明。

```
1  # 读取数据集
2  file = pd.read_csv(r'D:\A.大三上学习内容\NLP\Mid-term project\IMDB_sentiment
/IMDB Dataset.csv', sep=',')
3
4  # 获取标签并修改
5  labels = file._get_column_array(1)
6  labels = [1 if i == 'positive' else 0 for i in labels]
7
8  # 取出文本并分割为训练集和测试集
9  data = file._get_column_array(0)
10 train_data = data[:30000]
11 train_label = labels[:30000]
12 validate_data = data[30000:40000]
13 validate_label = labels[30000:40000]
14 test_data = data[40000:]
15 test_label = labels[40000:]
16
17 #删除标点符号，HTML标签和URL等与文本情感无关的噪音
18 def tokenize(sentence):
19     """
20     进行文本分词
21     :param sentence: str
22     :return: [str,str,str]
23     """
24     fileters = ['!', '"', '#', '$', '%', '&', '\(', '\)', '\*', '\+', ',',
'-', '\.', '/', ':', ';', '<', '=', '>',
25         '\?', '@', '\[', '\\', '\]', '^', '_', '`', '\{', '\|',
'\}', '~', '\t', '\n', '\x97', '\x96', "'",
26         '"']
27     sentence = sentence.lower() # 把大写转化为小写
28     sentence = re.sub("<br />", " ", sentence)
29     sentence = re.sub("|".join(fileters), " ", sentence) #正则表达，代换
filters的符号
30     result = [i for i in sentence.split(" ") if len(i) > 0]
31
32     return result
```

```

33
34
35 # 分别对三个数据集进行数据预处理
36 train_set=[]
37 for sentence in train_data:
38     res = [tokenlize(sentence)]
39     train_set+=res
40
41 validate_set=[]
42 for sentence in validate_data:
43     res = [tokenlize(sentence)]
44     validate_set+=res
45
46 test_set=[]
47 for sentence in test_data:
48     res = [tokenlize(sentence)]
49     test_set+=res
50
51 #保存处理好的数据，用于后续停用词的去除
52 with open("train_set.pkl", "wb") as f:
53     pickle.dump(train_set, f)
54 with open("train_label.pkl", "wb") as f:
55     pickle.dump(train_label, f)
56 with open("validate_set.pkl", "wb") as f:
57     pickle.dump(validate_set, f)
58 with open("validate_label.pkl", "wb") as f:
59     pickle.dump(validate_label, f)
60 with open("test_set.pkl", "wb") as f:
61     pickle.dump(test_set, f)
62 with open("test_label.pkl", "wb") as f:
63     pickle.dump(test_label, f)

```

2.remove_stopwords.py

经过data_processing.py处理的数据集之中还有许多停用词，这些词汇对于后续情感分类作用不大，对其进行删除。查找资料，获取英语中常见的停用词表，对照停用词表删除停用词。

上述的过程在remove_stopwords.py中实现，代码如下，结合代码对上述过程进行具体说明。

```

1 # 读入data_processing.py清洗后的数据
2 with open("train_set.pkl", "rb") as f:
3     train_set = pickle.load(f)
4 with open("validate_set.pkl", "rb") as f:
5     validate_set = pickle.load(f)
6 with open("test_set.pkl", "rb") as f:
7     test_set = pickle.load(f)
8
9
10 # 定义删除停用词的函数
11 def remove_stopwords(sentence):
12     stopwords = []
13     with open("stopword.txt") as f:
14         lines = f.readlines()
15     for line in lines:
16         line = line.rstrip("\n")

```

```

17         stopwords.append(line)
18
19     for sw in stopwords:
20         while sw in sentence:
21             sentence.remove(sw)
22     return sentence
23
24
25 #调用函数删除停用词，并保存处理后的数据集
26 for sentence in train_set:
27     sentence = remove_stopwords(sentence)
28 with open("train_set_remove_sw.pkl", "wb") as f:
29     pickle.dump(train_set, f)
30
31 for sentence in validate_set:
32     sentence = remove_stopwords(sentence)
33 with open("validate_set_remove_sw.pkl", "wb") as f:
34     pickle.dump(validate_set, f)
35
36 for sentence in test_set:
37     sentence = remove_stopwords(sentence)
38 with open("test_set_remove_sw.pkl", "wb") as f:
39     pickle.dump(test_set, f)
40

```

经过上述的处理，可得到预处理后的数据集。

二、使用 tf 进行训练的前馈神经网络

1.数据集处理

为了将数据集用于tf神经网络训练，需要进行以下的处理：

- 1 读入文本
- 2 分词：英语的分词较为简单，以空格为依据即可
- 3 建立字典，将每个词映射到一个唯一的索引，再根据索引实现文本序列化。把文本中的每个词语和其对应的数字用字典保存，再根据字典把数据量中的句子映射为包含数字的列表。

实现上述过程，需要处理以下问题：

(1) 不同句子长度不相同，每个batch的句子需要转化成相同的长度。因此可以对短句子进行填充，填充特殊字符；对于长句子则进行截断处理

(2) 对于新出现的在词典中没有的词语，需要使用特殊字符代理

- 4 统计各个词的词频，并按照下面的公式计算出各个词语的tf值

$$tf_{t,d} = \log_{10}(\text{count}(t,d)+1)$$

防止出现为0

→ 减小不同词之间过大的波动

- 5 将文本从词的序列转换为词语tf值的序列。每一个词语有两个tf值，一个是在标签为positive的数据集中的tf值，另一个是在标签为negative的数据集中的tf值，其中，negative的数据集中的tf值取负值。由于使用的神经网络为前馈神经网络，因此虽然每个词语tf值对应一个二维向量，但是拉平为2个数据值。因此处理后的数据集中，每个句子的长度为原来的两倍。

上述过程中，1~4由transform_word2tf.py实现，5由transform_dataset_tf.py实现。另外，定义了类Vocab，调用其方法可以实现统计词频、构造词典、把句子转换成数字序列的过程，在transform_word2tf.py中调用。

上述文件的代码中均有必要的注释，不再另作呈现与解释说明。最后保存的文件train_set_tf.pkl、validate_set_tf.pkl、test_set_tf.pkl即为前馈神经网络所需的数据集。

2.前馈神经网络

前馈神经网络未使用tensorflow等库的API，由自己搭建实现。project中的前馈神经网络有两个隐藏层，输出层使用softmax。读取的数据有train_set_tf.pkl、validate_set_tf.pkl、test_set_tf.pkl，包含了处理后IMDB的影评数据，以及train_label.pkl、validate_label.pkl、test_label.pkl，包含了IMDB的影评数据对应的标签。

前馈神经网络的具体说明如下：

1. 将这个2层神经网络实现为一个名为TwoLayerNet的类，存储在two_layer_net.py中。

- 参数说明：

params['W1']是第1层的权重，params['b1']是第1层的偏置。

params['W2']是第2层的权重，params['b2']是第2层的偏置。

grads['W1']是第1层权重的梯度，grads['b1']是第1层偏置的梯度。

grads['W2']是第2层权重的梯度，grads['b2']是第2层偏置的梯度。

- 类方法说明：

__init__(self, input_size, hidden_size, utput_size)：进行初始化。参数从头开始依次表示输入层的神经元数、隐藏层的神经元数、输出层的神经元数。

predict(self, x)：进行识别（推理）。参数x是数据集数据。

loss(self, x, t)：计算损失函数的值。参数x是数据集数据，t是正确解标签。

accuracy(self, x, t)：计算识别准确率。

recall(self, x, t, pos)：计算识别召回率。

precision(self, x, t, pos)：计算识别精度。

gradient(self, x, t)：计算权重参数的梯度。

2. functions.py保存神经网络中需要使用的relu等函数。layers.py定义了神经网络中的Affine层、Relu层等，便于在two_layer_net.py中调用。
3. tf神经网络的训练在tf_net.py中实现，主要包括读取数据集，再送入two_layer_net.py中训练，为防止过拟合，采取早停的方式，设置参数earlystop，如果性能相较于earlystop个epoch前没有提升，则停止训练。并保存earlystop个epoch中在验证集上表现的模型，并在测试集上测试，获取最终测试结果。相应过程保存在 *训练过程以及测试集表现.txt* 中。
4. call_model.py恢复最终训练好的模型并在测试集上进行测试。

说明：前馈神经网络未使用tensorflow等库的API，由自己搭建实现。但是由此带来的问题是训练完成后的模型不能直接保存为“.ckpt”文件，于是保存为“.pkl”文件，可以通过call_model.py恢复模型并在测试集上测试

三、使用 tf-idf 进行训练的前馈神经网络

1.数据集处理

为了将数据集用于tf-idf神经网络训练，需要进行以下的处理：

- 1 读入文本
- 2 分词：英语的分词较为简单，以空格为依据即可
- 3 建立字典，将每个词映射到一个唯一的索引，再根据索引实现文本序列化。把文本中的每个词语和其对应的数字用字典保存，再根据字典把数据量中的句子映射为包含数字的列表。

实现上述过程，需要处理以下问题：

(1) 不同句子长度不相同，每个batch的句子需要转化成相同的长度。因此可以对短句子进行填充，填充特殊字符；对于长句子则进行截断处理

(2) 对于新出现的在词典中没有的词语，需要使用特殊字符代理

- 4 统计各个词的词频，并按照下面的公式计算出各个词语的tf值、idf值、w值。

$$tf_{t,d} = \log_{10}(\text{count}(t,d)+1)$$

防止出现为0

减小不同词之间过大的波动

$$idf_t = \log_{10} \left(\frac{N}{df_t} \right)$$

不同文本出现次数

$$w_{t,d} = tf_{t,d} \times idf_t$$

- 5 将文本从词的序列转换为词语tf值的序列。每一个词语有两个tf-idf值，一个是在标签为positive的数据集中的tf-idf值，另一个是在标签为negative的数据集中的tf-idf值。由于使用的神经网络为前馈神经网络，因此虽然每个词语tf-idf值对应一个二维向量，但是拉平为2个数据值。因此处理后的数据集中，每个句子的长度为原来的两倍。

上述过程由transform_word2tfidf.py实现。最后保存的文件train_set_tfidf.pkl、validate_set_tfidf.pkl、test_set_tfidf.pkl即为前馈神经网络所需的数据集。

2.前馈神经网络

tf-idf神经网络的训练在tfidf_net.py中实现，前馈神经网络结构与tf中的基本一致，只是超参数不同，在此不再赘述。

四、使用 word2vec 进行训练的前馈神经网络

1.数据集处理

为了将数据集用于tf-idf神经网络训练，需要进行以下的处理：

- 1 读入文本
- 2 调用gensim.models.word2vec获取预训练好的word2vec模型。一般而言，vector_size维数越高正确率越高，但是计算量也随之越大，由于笔记本计算能量有限，折衷后vector_size选择10维。

- 3.每一个词语有10个word2vec值，由于使用的神经网络为前馈神经网络，因此虽然每个词语word2vec值对应一个10维张量，但是拉平为10个数据值。因此处理后的数据集中，每个句子的长度为原来的10倍。

上述过程中，1~2由build_word2vec_model.py实现，3由transform_dataset.py实现。

上述文件的代码中均有必要的注释，不再另作呈现与解释说明。最后保存的文件train_set_w2v.pkl、validate_set_w2v.pkl、test_set_w2v.pkl即为前馈神经网络所需的数据集。

2.前馈神经网络

word2vec神经网络的训练在word2vec_net.py中实现，前馈神经网络结构与tf中的基本一致，只是超参数不同以及input_size、hidden_size不同，在此不再赘述。

五、实验结果分析

1.使用 tf 进行训练的前馈神经网络模型

```
Performance in test set:
  accuracy:0.8274  recall:0.8368  precision:0.8217

Process finished with exit code 0
```

2.使用 tf-idf 进行训练的前馈神经网络模型

```
Performance in test set:
  accuracy:0.8230  recall:0.8500  precision:0.8068

Process finished with exit code 0
```

3.使用 word2vec 进行训练的前馈神经网络模型

```
Performance in test set:
  accuracy:0.7606  recall:0.7743  precision:0.7541

Process finished with exit code 0
```

4.结果分析

可以看到，使用tf与使用tf-idf进行训练的模型在测试集上的准确率达到82%以上，而使用word2vec的准确率则在76%左右。老师课堂上说准确率处于80%~90%即为比较正常的结果。目前的实验结果与预期有一点小小的偏差，主要偏差与原因解释如下：

1. tf 的准确率略高于 tf-idf 的原因：

- 这是因为在数据预处理时已经删除了停用词了，停用词属于高频的干扰词汇，也是使用idf来进行处理的主要目标。
- 由于已经预先处理了停用词，因此tf处理后的数据集中，基本没有高频的干扰词汇，因此训练后的模型准确率已经可以近似等同于使用tf-idf的水平。
- 而在预先处理了停用词的数据集中在此使用idf，有可能会将一些出现频率比较高的同时有助于情感分析的权重进行削弱（例如，bad，good等词），反而会导致模型的准确率相较于tf下降。

2. word2vec准确率未到达80%以上的原因：

- 这是由于word2vec模型的vector_size过小导致的。一般而言，vector_size维数越高正确率越高，但是计算量也随之越大，由于笔记本计算能量有限，折衷后vector_size选择10维。再往上增加计算机很容易算不动，而且占用大量内存。
- 另一种解决方案是对vector_size进行取平均值处理，但是实验后的效果并不很理想，同时取平均后对词向量的展示效果还比不上tf，反而浪费了word2vec预处理后的model。因此还是决定将整个词向量输入。
- 虽然vector_size只有10维，但训练后的模型准确率也已经十分接近80%，并且训练时很快就收敛，因此还是可以看出word2vec模型的优势。

3. 准确率未达到85%左右的原因：

- 经过与其他同学交流，了解到他们的准确率有些可以到达85%以上，而本次实验的模型均未达到85%，主要原因还是神经网络的问题。
- 一方面，使用卷积神经网络以及循环神经网络会比前馈神经网络更有优势，准确率更高。
- 另一方面，使用前馈神经网络的同学中，一些是用TensorFlow库添加神经层，从而搭建神经网络。利用库搭建神经网络时，可以借助库实现Dropout、Batch Normalization等以优化神经网络的训练效果。但在我自己搭建的神经网络中未实现相应的内容，因此训练后模型的准确率会略差一些。

六、心得体会

- 本次project主要实现的是运用Lecture 4: Vector Semantics & Embeddings的知识，进行文本清洗步骤和实践，并学习文本的表示方法。对于tf、idf的理解原先一直停留在理论层面，通过这次project有了更深入的理解，特别印象深刻的是对tf、idf取log10从而减少词汇间词频波动过大的处理。我特地做了未取log10的数据集进行训练，得到的模型在测试集的表现如下（该模型效果一般，没有附带在作业中一并发送）：

```
Performance in test set:
  accuracy:0.5397  recall:0.5546  precision:0.5392

Process finished with exit code 0
```

使用未取log10的tf进行训练的效果

可以看到效果很一般，而取log10从而减少词汇间的波动后，训练得到的模型准确率可以直接达到82%，是一个不小的提升。

- 另外是学会了神经网络的各部分的实现与搭建。作业目标是“熟悉将常用的深度学习模型应用于文本分类问题的流程和范式”，一开始我是使用LSTM来实现，虽然准确率可以达到85%以上，但是大部分内容都是调用现有的工具。

```
Performance in test set:
  accuracy:0.8793  recall:0.8802  precision:0.8725

Process finished with exit code 0
```

使用LSTM神经网络效果

由于之前没有接触过神经网络，所以对于神经网络从输入层到输出层的一系列处理感觉如同黑盒一般，不明就里。所以最后还是决定采用比较简单些的前馈神经网络，然后不调用现有工具，自己实现，虽然耗费了更多的时间，同时最终模型的准确率相比于LSTM反而下降了，但是却对整个神经网络的训练过程有了清晰的理解，感觉是更大的收获。




- 其实本来还想用卷积神经网络来训练，从而优化模型的，但是后来身体不舒服，没有精力继续深入，所以就停留在了前馈神经网络。加上神经网络的实现不是本次project的重点，所以期中project就做到这里了。但后面我还会继续学习CNN、RNN等模型，希望在后面的project中可以使用自己实现的效果更好的神经网络。

附：提交的作业中的文件说明

1.数据集预处理文件夹中，两个.py文件在一中已经说明。

名称	修改日期	类型	大小
 data_prosessing.py	2021/11/29 15:23	JetBrains PyChar...	3 KB
 remove_stopwords.py	2021/11/26 18:50	JetBrains PyChar...	4 KB

2~4.剩余三个文件夹内容相似

 2.tf前馈神经网络	2021/11/29 19:32	文件夹	
 3.tfidf前馈神经网络	2021/11/29 20:16	文件夹	
 4.word2vec前馈神经网络	2021/11/29 20:36	文件夹	

以tf前馈神经网络为例：

名称	修改日期	类型	大小
 functions.py	2021/11/24 16:54	JetBrains PyChar...	2 KB
 layers.py	2021/11/28 15:00	JetBrains PyChar...	2 KB
 net_checkpoint.py	2021/11/29 14:49	JetBrains PyChar...	1 KB
 tf_net.py	2021/11/29 15:48	JetBrains PyChar...	4 KB
 transform_dataset_tf.py	2021/11/29 19:32	JetBrains PyChar...	2 KB
 transform_word2tf.py	2021/11/29 15:59	JetBrains PyChar...	4 KB
 two_layer_net.py	2021/11/29 15:39	JetBrains PyChar...	3 KB
 vocab.py	2021/11/29 15:43	JetBrains PyChar...	3 KB
 save_tf_net.pkl	2021/11/29 14:45	PKL 文件	24,735 KB
 test_label.pkl	2021/11/26 15:53	PKL 文件	20 KB
 test_set_tf.pkl	2021/11/27 20:44	PKL 文件	17,620 KB
 训练过程以及测试集表现.txt	2021/11/29 15:42	文本文档	2 KB

- save_set_tf.pkl保存了最后训练得到的神经网络模型，可在net_checkpoint.py中恢复并测试。
- test_label_tf.pkl、test_set_tf.pkl为前馈神经网络所需的测试集数据。_label是IMDB影评标签，_set是IMDB影评内容。
- 训练过程以及测试集表现.txt 保存了模型训练、模型测试过程中准确率等信息。
- 各个.py文件都在对应的部分中有做说明，因此不再赘述。