商品评论的情感识别
实验手册

华为技术有限公司

目录

1 基于 TextCNN 算法的情感分类

1.1 实验介绍

1.1.1 关于本实验

本实验介绍使用深度学习 CNN 算法实现文本分类

1.1.2 实验目的

- 理解深度学习的文本预处理方法
- 掌握使用 tensorflow 搭建 CNN 文本分类模型

1.2 实验步骤

步骤 1　导入相关库

```
import jieba
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.utils import shuffle
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import classification_report
```

步骤 2　加载数据并做语料预处理

```
data = pd.read_csv('litemall_comments.csv')
# 默认显示前五行数据内容
data.head()
```

数据样式如下图所示：

	id	value_id	type	content	admin_content	user_id	has_picture	pic_urls	star	add_time	update_time	de
0	1	2001104	0	很漂亮，外观超出我的预期，老婆很喜欢，我自己看着也很喜欢，运行舒服挺快，没拿来玩游戏，所以不...		12251	0	NaN	5.0	2020-04-24 13:23:38	2020-04-24 13:23:38	
				给妈妈换的新手机，她用旧手机拍								

图1-1

语料预处理：删除无效的评论内容、并转换数据标签，将评论的‘star’标签值转换为离散的二类别 label 值‘0’（代表负面情感）和‘1’（代表正面情感）

```
# 负面情感数据加标签‘0’
neg_filt_fn = lambda row: row['star']<3.0 and len(row['content'])>0 and row['content'] != '用户未填写评价内容'
negative = data[['content']][data.apply(neg_filt_fn, axis=1)]
negative['label'] = 0
negative = shuffle(negative)
# 打印显示处理后的数据样式
negative
```

预处理后的负面情感评论数据示例如下图所示：

	content	label
20574	戴了两天开始手会痒，过了一天之后戴手表的地方还起了个水泡，真搞不懂这是什么材质的才会造成这种...	0
3798	白条6期免费没有，手机卡领不了。给的优惠一个领不到，合着没优惠。	0
18759	辣鸡 买了一个月就降价 刚好降了你的智商税	0
10401	非常差的体验，买个电脑没有说明使用，真的非常差	0
20202	说好的预约送体脂称呢?	0
...	...	...
7262	伤心，才买1个多月就掉价了，心里有点添堵	0
6568	服务态度真的差	0
20608	扬声器有噪音，客服从来没回过我	0

图1-2

```
# 正面情感数据加标签‘1’
pos_filt_fn = lambda row: row['star']>3.0 and len(row['content'])>5 and row['content'] != '用户未填写评价内容'
positive = data[['content']][data.apply(pos_filt_fn, axis=1)]
positive['label'] = 1
positive = shuffle(positive)
# 打印显示处理后的数据样式
positive
```

预处理后的正面情感评论数据示例如下图所示：

	content	label
10754	本本不错，就是用惯了win7，win10还需要时间来适应。	1
18488	给老爸买了以后体验非常好果断又二刷给老妈买了一款。颜色选择的是樱花粉，跟银色差不多。充满电以...	1
4135	外形外观：黑色很酷\n屏幕音效：完美\n拍照效果：一般，毕竟像素摆在那里\n运行速度：不卡顿...	1
25560	买上没有几天就降价了，骗人了。不能在华为商城上买。	1
26332	不愧是华为的五星级神机，非常棒，做年一个华粉，必须拥有	1
...	...	...
26738	做工质感：盒子的质感很不错，开盖的阻尼感也很好，而且也不会太大，放在包里刚刚好。\n音质音效...	1

图1-3

```
# 选取训练数据
train_data = pd.concat(7*[negative[:2200]]+[positive[:15369]])
train_data = shuffle(train_data)
# 选取测试数据
test_data = pd.concat([negative[2200:], positive[15369:]])
test_data = shuffle(test_data)
# 训练数据，训练数据标签
x_train, y_train = train_data.content.values, train_data.label.values
# 测试数据，测试数据标签
x_test, y_test = test_data.content.values, test_data.label.values
```

步骤 3　构建词汇表

```
# 分词，把词加入词汇表
vocab = set()
cut_docs = data.content.apply(jieba.cut).values
for doc in cut_docs:
    for word in doc:
        if word.strip():
            vocab.add(word.strip())

# 将词表写入本地 vocab.txt 文件
with open('vocab.txt', 'w',encoding='utf-8') as file:
    for word in    vocab:
        file.write(word)
        file.write('\n')
```

步骤 4　设置配置参数

```
class Config():
    embedding_dim = 300 # 词向量维度
```

```
    max_seq_len = 50 #  文章最大词数
    vocab_file = 'vocab.txt' #  词汇表文件路径
config = Config()
```

步骤 5　定义预处理类，用于将文本分词并转化成 id

```
cclass Preprocessor():
    def __init__(self, config):
        self.config = config
        #  初始化词和 id 的映射词典，预留 0 给 padding 字符，1 给词表中未见过的词
        token2idx = {"[PAD]": 0, "[UNK]": 1} # {word：id}
        with open(config.vocab_file, 'r',encoding='utf-8') as reader:
            for index, line in enumerate(reader):
                token = line.strip()
                token2idx[token] = index+2

        self.token2idx = token2idx

    def transform(self, text_list):
        #  文本分词，并将词转换成相应的 id，最后不同长度的文本 padding 长统一长度，后面补 0
        idx_list = [[self.token2idx.get(word.strip(), self.token2idx['[UNK]']) for word in jieba.cut(text)] for text in
text_list]
        idx_padding = pad_sequences(idx_list, self.config.max_seq_len, padding='post')

        return idx_padding
```

预处理类测试：

```
preprocessor = Preprocessor(config)
print(list(jieba.cut('客服问也不说话，态度有问题')))
preprocessor.transform(['客服问也不说话，态度有问题'])
```

测试结果示例如下图所示：

```
preprocessor = Preprocessor(config)
print(list(jieba.cut('客服问也不说话，态度有问题')))
preprocessor.transform(['客服问也不说话，态度有问题'])
```

```
['客服', '问', '也', '不', '说话', '，', '态度', '有', '问题']

array([[ 1902,  3826,  4068, 13664, 13132, 23760,  6071,  5440,   779,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0]])
```

图1-4

步骤 6　定义 TextCNN 主类，包括模型构建、训练、评估、测试函数。

```
class TextCNN(object):
    def __init__(self, config):
        self.config = config
        self.preprocessor = Preprocessor(config)
        self.class_name = {0: '负面', 1: '正面'}

    def build_model(self):
        # 模型架构搭建
        idx_input = tf.keras.layers.Input((self.config.max_seq_len,))
        input_embedding = tf.keras.layers.Embedding(len(self.preprocessor.token2idx),# 输入的词数：
len(self.preprocessor.token2idx
                                                    self.config.embedding_dim,# 300 维的词义向量
                                                    input_length=self.config.max_seq_len,
                                                    mask_zero=True)(idx_input)
        convs = []
        for kernel_size in [3, 4, 5]:
            c = tf.keras.layers.Conv1D(128, kernel_size, activation='relu')(input_embedding)# kernel_size：3*300,
在 50*300 这个方向上滑动
            c = tf.keras.layers.GlobalMaxPooling1D()(c)# 打印下 c 的 shape（128 维的向量） 50-3+1
            print(c.shape)
            convs.append(c)
        fea_cnn = tf.keras.layers.Concatenate()(convs)
        fea_cnn_dropout = tf.keras.layers.Dropout(rate=0.4)(fea_cnn)

        fea_dense = tf.keras.layers.Dense(128, activation='relu')(fea_cnn_dropout)
        output = tf.keras.layers.Dense(2, activation='softmax')(fea_dense)

        model = tf.keras.Model(inputs=idx_input, outputs=output)
        model.compile(loss='sparse_categorical_crossentropy',
                      optimizer='adam',
                      metrics=['accuracy'])

        model.summary()

        self.model = model

    def fit(self, x_train, y_train, x_valid=None, y_valid=None, epochs=5, batch_size=128, **kwargs):
        # 训练
        self.build_model()

        x_train = self.preprocessor.transform(x_train)
        if x_valid is not None and y_valid is not None:
            x_valid = self.preprocessor.transform(x_valid)

        self.model.fit(
            x=x_train,
            y=y_train,
```

```
            validation_data= (x_valid, y_valid) if x_valid is not None and y_valid is not None else None,
            batch_size=batch_size,
            epochs=epochs,
            **kwargs
        )

#           from keras.models import load_model

        #模型的保存
        self.model.save('model4comments.h5')


    def evaluate(self, x_test, y_test):
        # 评估
        x_test = self.preprocessor.transform(x_test)
        y_pred_probs = self.model.predict(x_test)
        y_pred = np.argmax(y_pred_probs, axis=-1)
        result = classification_report(y_test, y_pred, target_names=['负面', '正面'])
        print(result)


    def single_predict(self, text):
        # 预测
        input_idx = self.preprocessor.transform([text])
        predict_prob = self.model.predict(input_idx)[0]
        predict_label_id = np.argmax(predict_prob)

        predict_label_name = self.class_name[predict_label_id]
        predict_label_prob = predict_prob[predict_label_id]

        return predict_label_name, predict_label_prob
```

步骤 7　初始化模型并训练

```
textcnn = TextCNN(config)
textcnn.fit(x_train, y_train, x_test, y_test, epochs=5) # 训练
```

训练过程输出如下所示：

```
[*]: textcnn = TextCNN(config)
     textcnn.fit(x_train, y_train, x_test, y_test, epochs=5) # 训练
```

global_max_pooling1d_11 (Global	(None, 128)	0	conv1d_11[0][0]
concatenate_3 (Concatenate)	(None, 384)	0	global_max_pooling1d_9[0][0] global_max_pooling1d_10[0][0] global_max_pooling1d_11[0][0]
dropout_3 (Dropout)	(None, 384)	0	concatenate_3[0][0]
dense_6 (Dense)	(None, 128)	49280	dropout_3[0][0]
dense_7 (Dense)	(None, 2)	258	dense_6[0][0]

```
Total params: 7,670,222
Trainable params: 7,670,222
Non-trainable params: 0
Train on 30769 samples, validate on 382 samples
17280/30769 [===============>.............] - ETA: 33s - loss: 0.2165 - accuracy: 0.9136
```

图1-5

步骤 8　测试集评估

textcnn.evaluate(x_test, y_test) # 测试集评估

测试集评估结果输出如下所示：

```
textcnn.evaluate(x_test, y_test) # 测试集评估
```

	precision	recall	f1-score	support
负面	0.93	0.77	0.85	182
正面	0.82	0.95	0.88	200
accuracy			0.87	382
macro avg	0.88	0.86	0.86	382
weighted avg	0.88	0.87	0.87	382

图1-6

步骤 9　单句评论测试

textcnn.single_predict("没有血氧检测，我买的是假的吗？") # 单句预测

textcnn.single_predict("很帅，推荐！速度够快！\n 我配了蓝牙键盘和鼠标！\n 可以办公了！")

单句评论测试结果输出如下所示：

```
textcnn.single_predict("没有血氧检测，我买的是假的吗？") # 单句预测
```

```
('负面', 0.9972187)
```

```
textcnn.single_predict("很帅，推荐！速度够快！\n我配了蓝牙键盘和鼠标！\n可以办公了！")
```

```
('正面', 1.0)
```

图1-7

1.3 实验小结

本节实验介绍了使用 tensorflow 搭建 CNN 文本分类模型，通过实验使学员了解深度学习处理文本数据的基本流程，同时通过动手搭建 CNN 网络，加深对 CNN 的理解。

2 基于词云的商品评论可视化

2.1 实验介绍

2.1.1 关于本实验

本实验介绍如何使用 wordcloud 实现商品评论的可视化。

wordcloud：词云以词语为基本单位，更加直观和艺术地展示文本，wordcloud 库把词云当作一个 WordCloud 对象，wordcloud.WordCloud( )代表一个文本对应的词云，可以根据文本中词语出现的频率等参数绘制词云。

通过 pip install wordcloud 来安装。

2.1.2 实验目的

- 掌握使用 wordcloud 实现商品评论的可视化。

2.2 实验步骤

步骤 1　导入实验所需的第三方相关库文件

```
from wordcloud import WordCloud
import jieba
import matplotlib.pyplot as plt
import pandas as pd
```

步骤 2　定义语料预处理函数

```
def content_preprocess(csv):
    '预处理：提取评论内容'
    df = pd.read_csv(csv)
    # 提取评论内容的一列；清洗数据：用户未填写评价内容
    preprocessed_data = df[df['content']!='用户未填写评价内容'][['content']]
    preprocessed_data.to_csv('./content4wordcloud.csv',header=0,index=0) #不保存行列名
# 调用预处理函数
content_preprocess('litemall_comments.csv')
```

步骤 3　定义创建词云的预处理函数

```
def word_cloud_creation(filename):
    '''创建词云，并进行分词'''
#       text = open(filename, encoding='gb18030',errors='ignore').read()
    text = open(filename, encoding='utf-8',errors='ignore').read()
    word_list = jieba.cut(text, cut_all=True)
    wl = ' '.join(word_list)
    return wl
```

步骤 4　定义设置词云属性的函数

```
def word_cloud_settings():
    wc = WordCloud( height=1200, width=1500, font_path='C:\Windows\Fonts\simfang.ttf' )
    return wc
```

步骤 5　定义生成词云并展示的函数

```
def word_cloud_implementation(wl, wc):
    '''生成词云，并展示'''
    my_words = wc.generate(wl)
    plt.imshow(my_words)
    plt.axis('off')
    wc.to_file(f'./result/word_cloud.png')
    plt.show()
```

步骤 6　封装词云生成的功能函数

```
def word_cloud_show():
    '''将商品评论转为高频词汇的词云'''
    wl = word_cloud_creation('content4wordcloud.csv')
    wc = word_cloud_settings()
    word_cloud_implementation(wl, wc)
```

步骤 7　词云可视化

```
# 展示词云生成结果
word_cloud_show()
```

示意图如下所示：

```
word_cloud_show()  # 高频词云
```

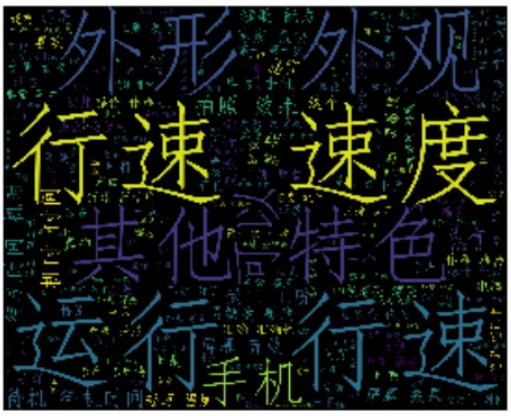

图2-1

2.3 实验小结

本实验主要是利用自然语言处理的常用工具包，利用分词、词频统计等第三方库，输出了评论内容的可视化词云。