《自然语言处理》

-情感分析



华为技术有限公司

目录

[1 TextCNN情感分析实验 2](#_Toc55587230)

[1.1 实验简介 2](#_Toc55587231)

[1.1.1 实验背景 2](#_Toc55587232)

[1.1.2 实验目的 2](#_Toc55587233)

[1.1.3 预备知识 2](#_Toc55587234)

[1.1.4 实验环境 3](#_Toc55587235)

[1.2 实验步骤 3](#_Toc55587236)

[1.3 实验小结 11](#_Toc55587237)

# TextCNN情感分析实验

## 实验简介

### 实验背景

情感分析是自然语言处理最基本的应用，情感分类较为简单，实用性也较强。常见的购物网站、电影网站都可以采集到相对高质量的数据集，也很容易给业务领域带来收益。例如，可以结合领域上下文，自动分析特定类型客户对当前产品的意见，可以分主题分用户类型对情感进行分析，以作针对性的处理，甚至基于此进一步推荐产品，提高转化率，带来更高的商业收益。

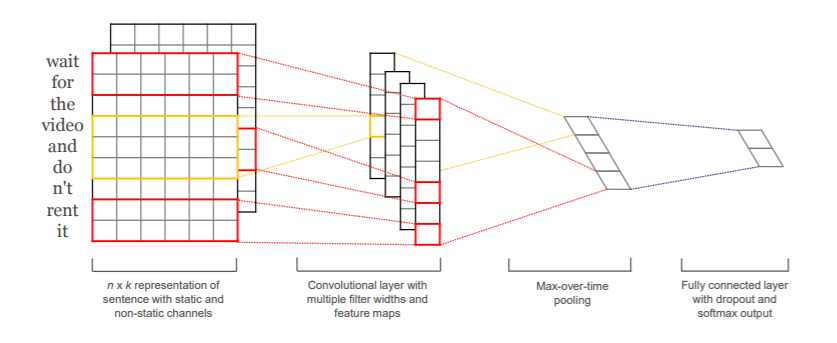
本实验主要基于卷积神经网络对互联网社区、商城的评论信息进行情感分析，判断其情感倾向。

### 实验目的

* 理解文本分类的基本流程
* 理解CNN网络在文本任务中的用法
* 掌握tensorflow搭建文本分类模型的方法

### 预备知识

与传统的应用在图像上的CNN网络不同，CNN应用在文本上时，卷积核的宽度需与词向量的维度保持一致，卷积在文本上提取的是类似N-gram的局部特征。



### 实验环境

* ModelArts Tensorflow2.1.0 CPU/GPU环境

## 实验步骤

安装依赖库

安装jieba分词依赖库

输入：

! pip install jieba

输出：



导入依赖库

输入：

import jieba

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from sklearn.metrics import classification\_report

数据加载

输入：

train\_data = pd.read\_csv('data/train.tsv', sep='\t')

valid\_data = pd.read\_csv('data/dev.tsv', sep='\t')

test\_data = pd.read\_csv('data/test.tsv', sep='\t')

x\_train, y\_train = train\_data.text\_a.values, train\_data.label.values # 训练集

x\_valid, y\_valid = valid\_data.text\_a.values, valid\_data.label.values # 验证集

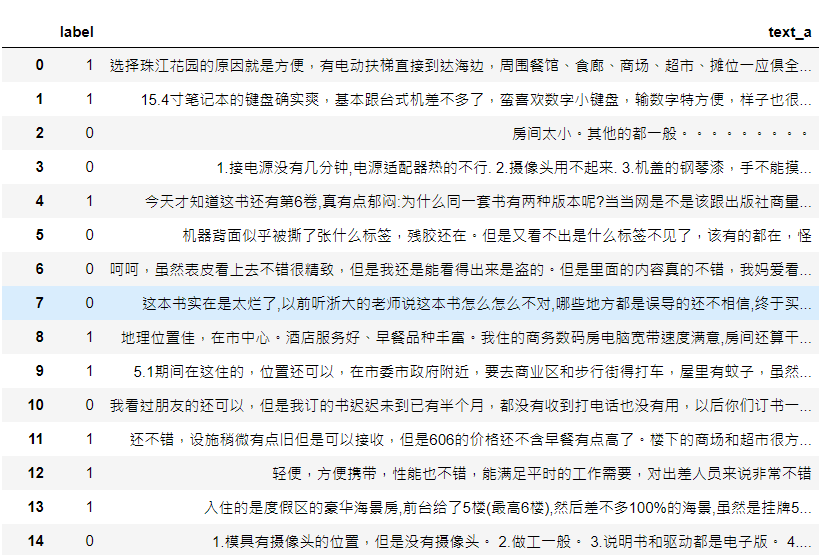
x\_test, y\_test = test\_data.text\_a.values, test\_data.label.values # 测试集

查看原始数据

输入：

train\_data

输出：

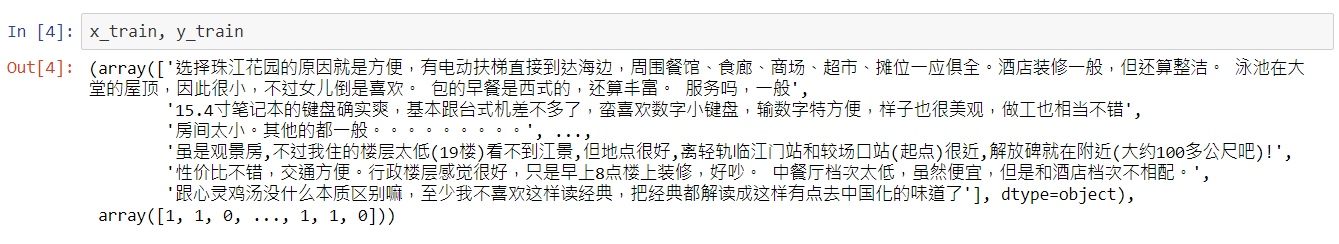


查看解析后的训练数据

输入：

x\_train, y\_train

输出：



构建词汇表

词汇表用于将文字映射成对应的索引ID.

输入：

vocab = set()

cut\_docs = train\_data.text\_a.apply(lambda x: jieba.cut(x)).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('data/vocab.txt', 'w') as file:

for word in vocab:

file.write(word)

file.write('\n')

定义配置参数

定义词向量维度，最大填充长度等参数

输入：

class Config():

embedding\_dim = 300 # 词向量维度

max\_seq\_len = 200 # 文章最大词数

vocab\_file = 'data/vocab.txt' # 词汇表文件路径

config = Config()

定义预处理类

预处理类用于将输入文本转换成模型输入需要的ID矩阵

输入：

class 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') 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

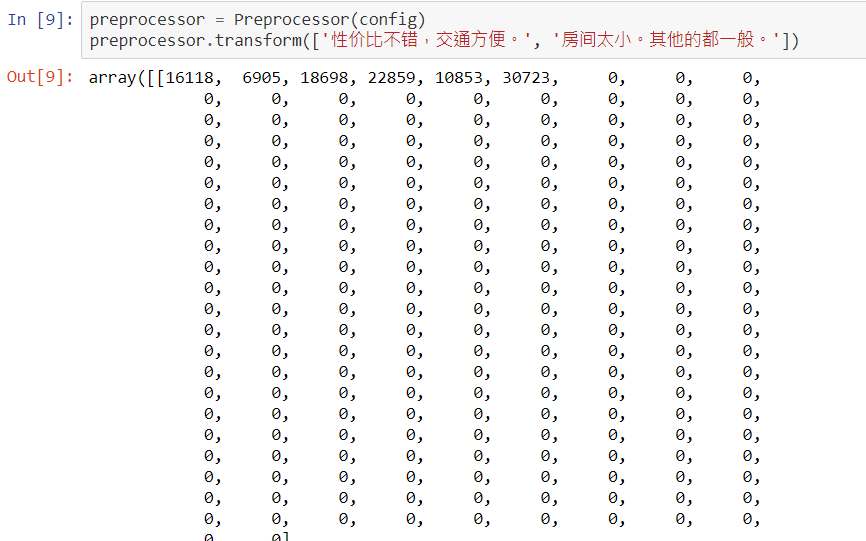
测试预处理输出：

输入：

preprocessor = Preprocessor(config)

preprocessor.transform(['性价比不错，交通方便。', '房间太小。其他的都一般。'])

输出：



定义模型类

模型类定义了模型结构搭建、训练、评估、加载离线模型、在线推理函数。

输入：

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),

self.config.embedding\_dim,

input\_length=self.config.max\_seq\_len,

mask\_zero=True)(idx\_input)

# 请同学们补全CNN模型

# 代码补全结束

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, callbacks=None, \*\*kwargs):

# 训练

self.build\_model()

x\_train = self.preprocessor.transform(x\_train)

valid\_data = None

if x\_valid is not None and y\_valid is not None:

x\_valid = self.preprocessor.transform(x\_valid)

valid\_data = (x\_valid, y\_valid)

self.model.fit(

x=x\_train,

y=y\_train,

validation\_data= valid\_data,

batch\_size=batch\_size,

epochs=epochs,

callbacks=callbacks,

\*\*kwargs

)

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，返回predict\_label\_name(预测类别名), predict\_label\_prob（预测概率）

# 请同学们补全对text情感类别预测代码

# 代码补全结束

return predict\_label\_name, predict\_label\_prob

def load\_model(self, ckpt\_file):

self.build\_model()

self.model.load\_weights(ckpt\_file)

启动训练

定义early\_stop早停回调函数，checkpoint保存回调函数，传入训练数据，启动训练

输入：

# 定义early stop早停回调函数

patience = 6

early\_stop = tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=patience)

# 定义checkpoint回调函数

checkpoint\_prefix = './checkpoints/textcnn\_imdb\_ckpt'

checkpoint\_callback=tf.keras.callbacks.ModelCheckpoint(

filepath=checkpoint\_prefix,

save\_weights\_only=True,

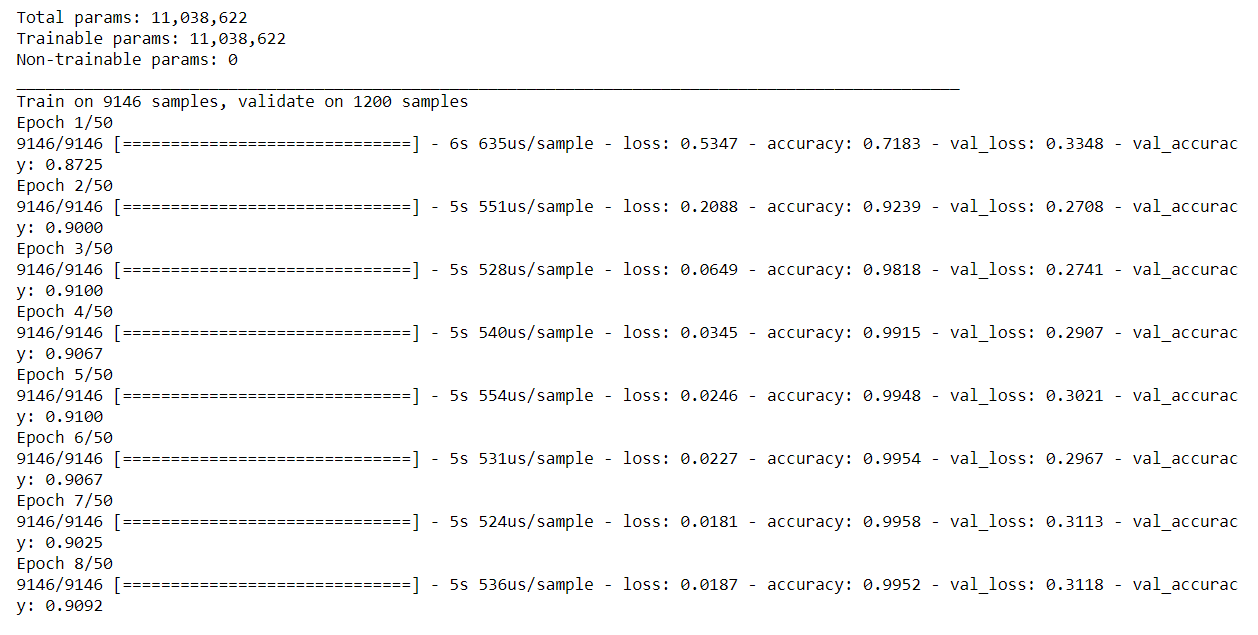
save\_best\_only=True)

# 初始化模型类，启动训练

textcnn = TextCNN(config)

textcnn.fit(x\_train, y\_train, x\_valid, y\_valid, epochs=50, callbacks=[early\_stop, checkpoint\_callback]) # 训练

输出：



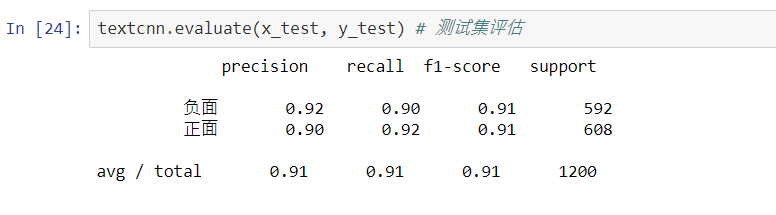
测试评估

评估测试集上的效果指标

输入：

textcnn.evaluate(x\_test, y\_test) # 测试集评估

输出：



离线加载预测

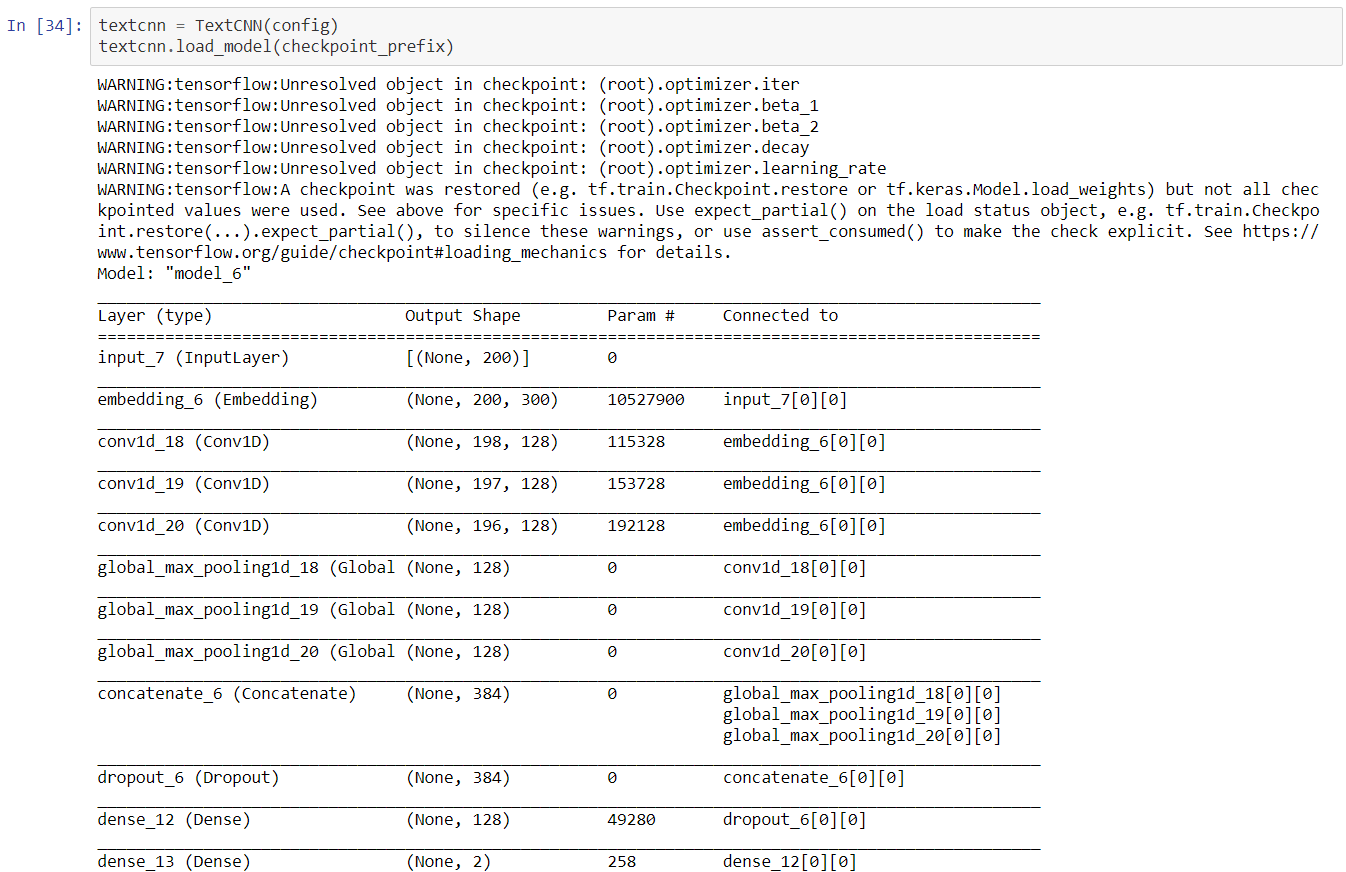
加载保存好的离线模型

输入：

textcnn = TextCNN(config)

textcnn.load\_model(checkpoint\_prefix)

输出：



在线测试

取单条评论文本数据，进行测试，输出情感类别及其概率

测试正面样例

输入：

textcnn.single\_predict("外观很漂亮，出人意料地漂亮，做工非常好") # 单句预测

输出：



测试负面样例

输入：

textcnn.single\_predict("书的内容没什么好说的，主要是纸张、印刷太差，所用的纸非常粗糙比一般的盗版书还要差，裁的也不好。") # 单句预测

输出：



## 实验小结

本实验介绍了如何使用tensorflow搭建用于文本分类的CNN模型，通过实验，使学员了解文本分类任务的基本流程，同时理解卷积网络在文本任务中的使用方法，通过实验也加深了对CNN网络的理解，同时提升了代码实践能力。