实训案例

新闻分类

1. 题目介绍

1.1 题目背景

自媒体的出现，让网络上各类文章的数量大大增加。现在有一家自媒体平台收到了大量投稿，试开发一个机器学习模型来对这些新闻稿件进行分类，降低工作人员的负担。

1.2 题目要求

1) 任务提供包括数据读取、基础模型、模型训练等基本代码

2) 参赛选手需完成核心模型构建代码，并尽可能将模型调到最佳状态

3) 模型单次推理时间不超过 10 秒

1.3 开发环境

可以使用基于 Python 分词库进行文本分词处理，使用 Numpy 库进行相关数值运算，使用 sklearn 中的机器学习模型，使用 Tensorflow，Keras 等框架建立深度学习模型等。

1.4 注意事项

- Python 与 Python Package 的使用方式，可在右侧 `API` 文档中查阅。

- 当右上角的『Python 3』长时间指示为运行中的时候，造成代码无法执行时，可以重新启动 Kernel 解决（左上角『Kernel』-『Restart Kernel』）。

2. 题目内容

2.1 介绍数据集

数据集总共有 5 种类型共 50000 条新闻文本，分别是：科技，社会，娱乐，财经，体育。

- 数据集路径为"./datasets/5fbcdfa05005208e83d1ede4-momodel/news"

[1]:
```
# 导入相关包
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
import time
import random
import jieba as jb
import numpy as np
import jieba.analyse
import tensorflow as tf
import tensorflow.keras as K
from matplotlib import pyplot as plt
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation
from tensorflow.keras.utils import to_categorical
```

读取数据集，保存在字典中

[2]:
```
dataset = {}
dataset_path = "./datasets/5fbcdfa05005208e83d1ede4-momodel/news"
files= os.listdir(dataset_path)
for file in files:
    path = os.path.join(dataset_path, file)
    if not os.path.isdir(path) and not file[0] == '.': # 跳过隐藏文件和文件夹
        f = open(path, 'r',  encoding='UTF-8') # 打开文件
        for line in f.readlines():
            dataset[line] = file[:-4]
```

现在分别展示其中的每个类别中的 1 条新闻。

[ ]:
```
types = ("科技", "社会", "娱乐", "财经", "体育")
for t in types:
    for k, v in list(dataset.items()):
        if v == t:
            print(k, "Type:", v, '\n')
            break
```

2.2 数据集预处理

在做文本挖掘的时候，首先要做的预处理就是分词。
英文单词天然有空格隔开容易按照空格分词，但是也有时候需要把多个单词做为一个分词，比如一些名词如”New York”，需要做为一个词看待。
而中文由于没有空格，分词就是一个需要专门去解决的问题了。
这里我们使用 jieba 包进行分词，使用**精确模式**、**全模式**和**搜索引擎模式**进行分词对比。
更多方法参考：https://github.com/fxsjy/jieba

[ ]:
```
article = next(iter(dataset))
```

[ ]:
```
# 精确模式分词
article_split = [" ".join(jb.cut(article, cut_all=False))]
print("精确模式分词结果:\n",article_split)
```

[ ]:
```
# 全模式分词
article_split = [" ".join(jb.cut(article, cut_all=True))]
```

```
print("全模式分词结果:\n",article_split)
```

[ ]:
```
# 搜索引擎模式分词
article_split = [" ".join(jb.cut_for_search(article))]
print("搜索引擎模式分词结果:\n",article_split)
```

使用 TF-IDF 算法统计各条新闻的关键词频率

TF-IDF（term frequency–inverse document frequency，词频-逆向文件频率）是一种用于信息检索与文本挖掘的常用加权技术。
* TF-IDF 是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。
* TF-IDF 的主要思想是：如果某个单词在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。
这里我们使用 jieba 中的默认语料库来进行关键词抽取，并展示每位作者前 5 个关键词

[ ]:
```
# 将新闻进行词频统计
str_full = {}
for t in types:
    str_full[t] = ""

for k, v in dataset.items():
    str_full[v] += k

for k, v in str_full.items():
    print(k,":")
    for x, w in jb.analyse.extract_tags(v, topK=5, withWeight=True):
        print('%s %s' % (x, w))
```

2.3 采用 Tensorflow Keras 建立一个简单的深度神经网络模型

通过 Tensorflow Keras 构建深度学习模型的步骤如下：+ 定义模型——创建一个模型并添加配置层 + 编译模型——指定损失函数和优化器，并调用模型的 compile() 函数，完成模型编译。+ 训练模型——通过调用模型的 fit() 函数来训练模型。+ 模型预测——调用模型的 evaluate() 或者 predict() 等函数对新数据进行预测。

2.3.1 读取数据集

首先需要读取数据集，记录每个片段的作者并保存。

[ ]:
```
# 读取数据和标签
def load_data(dataset_path):
    """
    :param dataset_path:数据集文件夹路径
    :return:返回读取的片段和对应的标签
    """
    sentences = [] # 新闻文本
    target = [] # 类别

    # 定义 label 到数字的映射关系
    labels = {'科技': 0, '社会': 1, '娱乐': 2, '财经': 3, '体育': 4}

    files = os.listdir(dataset_path)
    for file in files:
        path = os.path.join(dataset_path, file)
        if not os.path.isdir(path) and not file[0] == '.':
            with open(path, 'r', encoding='UTF-8') as f: # 打开文件
                for line in f.readlines():
                    sentences.append(line)
                    target.append(labels[file[:-4]])

    target = np.array(target)
    encoder = LabelEncoder()
    encoder.fit(target)
    encoded_target = encoder.transform(target)
    dummy_target = to_categorical(encoded_target)
    return sentences, dummy_target
```

对读取的片段进行分词，由于分词后的片段任然为中文词语组成的序列，需要创建词汇表，将每个中文词映射为一个数字。这里使用 Tensorflow 的 Tokenizer 创建词汇表。

[ ]:
```
def padding(text_processed, max_sequence_length):
    """
    数据处理，如果使用 lstm，则可以接收不同长度的序列（这里不用 lstm，只做等长处理）
```

```
    :text_processed: 不定长的 Token 化文本序列，二维 list
    :path: 数据集路径
    :max_sequence_length: padding 大小，长句截断短句补 0
    :return 处理后的序列，numpy 格式的二维数组
    """
    res = []
    for text in text_processed:
        if len(text) > max_sequence_length:
            text = text[:max_sequence_length]
        else:
            text = text + [0 for i in range(max_sequence_length-len(text))]
        res.append(text)
    return np.array(res)
```

[ ]:
```
# 查看我们创建词汇表的结果
sentences,target = load_data(dataset_path)

# 定义是文档的最大长度。如果文本的长度大于最大长度，那么它会被剪切，反之则用 0 填充
max_sequence_length = 80

# 使用 jieba 机精确模式分词
sentences = [" ".join(jb.cut(t, cut_all=False)) for t in sentences]
print(sentences[0])
```

Tokenizer 类允许使用两种方法向量化一个文本语料库：将每个文本转化为一个整数序列（每个整数都是词典中标记的索引）；或者将其转化为一个向量，其中每个标记的系数可以是二进制值、词频、TF-IDF 权重等。

[ ]:
```
# 构建词汇表
vocab_processor = tf.keras.preprocessing.text.Tokenizer(num_words=100,
                                                        filters='!"#$%&()*+,-./:
 ↪;<=>?@[\]^_`{|}~ ',
                                                        oov_token='<UNK>')

# 要用以训练的文本列表
vocab_processor.fit_on_texts(sentences)
```

将词汇表保存为 json，后续可以直接读取，读取方式为 **tf.keras.preprocessing.text.tokenizer_from_json(json_string)**，可以获得 vocab_processor 相同参数的对象

[ ]:
```
# 将词汇表保存到路径
vocab_keras_path = "results/vocab_keras.json"

vocab_json_string = vocab_processor.to_json()
file = open(vocab_keras_path,"w")
file.write(vocab_json_string)
file.close()
```

[ ]:
```
# 如果之前已经生成了词汇表，就不用在重复生成了，直接从文件中加载词汇表
vocab_json_string = ""
vocab_keras_path = "results/vocab_keras.json"
with open(vocab_keras_path) as f:
    vocab_json_string = f.read()
vocab_processor = tf.keras.preprocessing.text.
↪tokenizer_from_json(vocab_json_string)
```

[ ]:
```
# 词序列的列表，将 sentences 文本序列化
text_processed = vocab_processor.texts_to_sequences(sentences)

# 将句子 padding 为固定长度，如果使用 lstm 则不需要 padding 为固定长度
text_processed = padding(text_processed, max_sequence_length)
```

打乱并切分数据集，取 30% 数据作为验证集

[ ]:
```
# 验证集比例
val_split = 0.3

# 打乱顺序
text_target = list(zip(text_processed, target))
# print(text_target)
random.shuffle(text_target)
text_processed[:], target[:] = zip(*text_target)
text_processed[:], target[:] = zip(*text_target)

# 验证集数目
```

```
val_counts = int(val_split*len(text_target))

# 切分验证集
val_X = text_processed[-val_counts:]
val_y = target[-val_counts:]
train_X = text_processed[:-val_counts]
train_y = target[:-val_counts]
```

2.3.2 常见的创建模型方式介绍

Keras 的核心数据结构是 model，一种组织网络层的方式。最简单的模型是 Sequential 顺序模型，它由多个网络层线性堆叠。对于更复杂的结构，你应该使用 Keras 函数式 API，它允许构建任意的神经网络图。下面先来看看 Sequential 顺序模型：

[ ]:
```
# 方式一：使用 .add() 方法将各层添加到模型中
# 导入相关包
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation

# 选择模型，选择序贯模型（Sequential()）
model = Sequential()

# 构建网络层
# 添加全连接层
model.add(Dense(64, input_shape=(max_sequence_length,)))

# 添加激活层，激活函数是 relu
model.add(Activation('relu'))

# 打印模型概况
model.summary()
```

[ ]:
```
# 方式二：网络层实例的列表构建序贯模型
# 导入相关的包
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
```

```
# 选择模型，选择序贯模型（Sequential()）
# 通过将网络层实例的列表传递给 Sequential 的构造器，来创建一个 Sequential 模型
model = Sequential([
    Dense(64, input_shape=(max_sequence_length,)),
    Activation('relu')
])

# 打印模型概况
model.summary()
```

[ ]:
```
# 方式三：函数式模型
# 导入相关的包
from tensorflow.keras.layers import Input, Dense,Activation
from tensorflow.keras.models import Model

# 输入层，返回一个张量 tensor
inputs = Input(shape=(max_sequence_length,))

# 全连接层，返回一个张量
output_1 = Dense(64)(inputs)

# 激活函数层
predictions= Activation(activation='relu')(output_1)

# 创建一个模型，包含输入层、全连接层和激活层
model = Model(inputs=inputs, outputs=predictions)

# 打印模型概况
model.summary()
```

2.3.3 建立深度学习模型

- 创建一个简单的 CNN 神经网络模型用于文本分类

[ ]:
```
def dnn_model(train_X, train_y, val_X, val_y, model_save_path='results/demo.h5',
              log_dir="results/logs/"):

    # 选择模型，选择序贯模型（Sequential()）
    model = K.Sequential()
    # 全连接神经网络
    model.add(K.layers.Dense(64, input_shape=(80,), activation="relu"))
    # 全连接神经网络，采用 softmax 作为激活函数
    model.add(K.layers.Dense(5, activation="softmax"))
    # 编译模型
    model.compile(loss="categorical_crossentropy",
                  optimizer="adam",
                  metrics=["accuracy"])
    # 训练模型
    history = model.fit(train_X.astype(np.float64),
                        train_y,
                        epochs=2,
                        validation_data=(val_X, val_y) )

    # 保存模型
    model.save(model_save_path)

    return history, model
```

- 模型训练过程和模型概况

[ ]:
```
# 开始时间
start = time.time()

# 数据预处理
data_path = "./datasets/5fbcdfa05005208e83d1ede4-momodel/news"

# 训练模型，获取训练过程和训练后的模型
history,model = dnn_model(train_X, train_y, val_X, val_y)

# 打印模型概况和模型训练总数长
model.summary()
```

```
print("模型训练总时长：",time.time()-start)
```

- 模型训练过程图形化

[ ]:
```
def plot_training_history(res):
    """
    绘制模型的训练结果
    :param res: 模型的训练结果
    :return:
    """
    # 绘制模型训练过程的损失和平均损失
    # 绘制模型训练过程的损失值曲线，标签是 loss
    plt.plot(res.history['loss'], label='loss')

    # 绘制模型训练过程中的平均损失曲线，标签是 val_loss
    plt.plot(res.history['val_loss'], label='val_loss')

    # 绘制图例，展示出每个数据对应的图像名称和图例的放置位置
    plt.legend(loc='upper right')

    # 展示图片
    plt.show()

    # 绘制模型训练过程中的的准确率和平均准确率
    # 绘制模型训练过程中的准确率曲线，标签是 acc
    plt.plot(res.history['accuracy'], label='accuracy')

    # 绘制模型训练过程中的平均准确率曲线，标签是 val_acc
    plt.plot(res.history['val_accuracy'], label='val_accuracy')

    # 绘制图例，展示出每个数据对应的图像名称，图例的放置位置为默认值。
    plt.legend()

    # 展示图片
    plt.show()
```

[ ]:
```
# 绘制模型训练过程曲线
plot_training_history(history)
```

- 加载模型和模型评估

[ ]:
```
def load_and_model_prediction(val_X, val_y, model_path = 'results/demo.h5'):
    """
    加载模型和模型评估，打印验证集的 loss 和准确度
    :param validation_generator: 预测数据
    :return:
    """
    # 加载模型
    model = K.models.load_model(model_path)
    # 获取验证集的 loss 和 accuracy
    loss, accuracy = model.evaluate(val_X, val_y)
    print("\nLoss: %.2f, Accuracy: %.2f%%" % (loss, accuracy * 100))
```

[ ]:
```
load_and_model_prediction(val_X, val_y)
```

2.4 加载模型并进行预测

[ ]:
```
# 预测某一篇新闻
def predict(text):
    """
    :param text: 中文字符串
    :return: 字符串格式的类型，比如:' 科技'
    """
    labels = {0: '科技', 1: '社会', 2: '娱乐', 3: '财经', 4: '体育'}
    sen_prosessed = " ".join(jb.cut(text, cut_all=False))
    max_sequence_length = 80
    sen_prosessed = vocab_processor.texts_to_sequences([sen_prosessed])[0]
    sen_prosessed = padding([sen_prosessed], max_sequence_length)[0]
    sen_prosessed = np.array(sen_prosessed).reshape(1, -1)

    # 加载模型进行预测
    result = model.predict(sen_prosessed)
```

```
    prediction = labels[list(result[0]).index(max(result[0]))]
    # ------------------------------------------------------------------------

    return prediction

# 使用一篇新闻做测试
text = "跨国公司欲抄底中国芯片设计产　孙燕　在全球半导体行业困境面前，国际半导体巨头将矛头瞄准了本土半导体设计企业　日前，《第一财经日报》从可靠途径获悉，美国 Aptina 近日已经悄然并购了上海智多微电子公司的手机软件平台设计部门　上海智多微电子公司一位内部员工证实了这一消息，该内部员工表示，由于自己也正在办理离职手续，交易金额并未宣布　成立于 2003 年 9 月的智多微电子主要从事移动多媒体应用处理芯片和手机平台解决方案的研发，目前已经开发出 9 款多媒体应用处理器。智多微电子董事长兼 CEO 胡祥在去年 10 月还曾对媒体表示，智多微电子正在试图降低智能手机入市门槛。龙旗、希姆通、天宇朗通、夏新、天时达等 16 家国产手机厂家都曾是智多微电子的客户　而 Aptina 公司是美光科技有限公司 2008 年初才成立的子公司，为手机制造商提供 200 万、300 万和 500 万像素 CMOS 图像传感器，是 CMOS 成像行业的领先企业　“智多手机软件平台部门的员工数大概在 30 人至 40 人，这部分员工应该会全部转到 Aptina 公司。”知情人士透露，此前智多微电子的员工数多达 200 人，今年逐步缩减到 100 人左右　该知情人士指出，智多微电子竞争对手联发科技（MTK）手机芯片功能越来越强大，蚕食了智多微电子的客户和其生存空间　“目前国内很多半导体芯片企业都很缺钱，受金融风暴影响，融资渠道基本关闭。”业内一位资深分析人士指出，虽然中国集成电路产业发展 20 多年，但至今营业额达到 1 亿美元的公司很少“从 2007 年 10 月到 2008 年的 10 月，总共有 4 家本土 IC 设计公司被外国公司收购，2009 年这种并购案例还会增加。”iSuppli 中国半导体行业分析师顾文军指出，随着现在资本市场的低迷，而中国半导体上市公司在美国纳斯达克表现均不佳，并购则成了公司的一种出路。被并购的 4 家本土 IC 设计公司为：上海杰脉、杭州晶圆微芯、深圳原核、成都威斯达。　相关报道　寒流袭来：中国芯片业熬　2011 年中国芯片市场将达 2000 亿　美股评论：全球芯片业濒临绝"
print(predict(text))
```

3. 题目

题目内容：根据一段新闻，预测这段新闻的类别。

3.1 创建并训练模型

深度学习模型训练流程, 包含数据处理、创建模型、训练模型、模型保存、评价模型等。
如果对训练出来的模型不满意, 你可以通过调整模型的参数等方法重新训练模型, 直至训练出你满意的模型。
如果你对自己训练出来的模型非常满意, 则可以提交了!

注意:

1. 你可以在我们准好的接口中实现深度学习模型（若使用可以修改除 predict 外的函数接口）, 也可以自己实现深度学习模型实现，但需要满足 predict 函数的输入输出符合格式要求!
2. 写好代码后可以在 Py 文件中使用 GPU 进行模型训练。

========== 实现自己的深度学习模型代码答题区 ==========

双击下方区域开始编写 **数据处理**、**创建模型**、**训练模型**、**保存模型**和 **评估模型**等部分的代码，如果使用其他框架训练模型可以自行实现。

[ ]:
```
# 导入相关包
import copy
import pandas as pd
import numpy as np
import tensorflow.keras.backend  as K
from matplotlib import pyplot as plt
from sklearn import preprocessing
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM,GRU
from tensorflow.keras.callbacks import TensorBoard,ModelCheckpoint
from tensorflow.keras import optimizers


def processing_data(data_path, validation_split = 0.3 ):
    """
    数据处理
    :data_path: 数据集路径
    :validation_split: 划分为验证集的比重
    :return: train_X,train_y,val_X,val_y
    """
```

```
    # --------------- 在这里实现中文文本预处理，包含分词，建立词汇表等步骤␣
↪------------------------------
    train_X, train_y, test_X, test_y = None,None,None,None
    pass

    #␣
↪-----------------------------------------------------------------------------------------

    return train_X, train_y, test_X, test_y

def model(train_X, train_y, val_X, val_y, save_model_path):
    """
    创建、训练和保存深度学习模型
    :param train_X: 训练集特征
    :param train_y: 训练集 target
    :param test_X: 测试集特征
    :param test_y: 测试集 target
    :param save_model_path: 保存模型的路径和名称
    """
    # ---------------------- 实现模型创建、训练和保存等部分的代码␣
↪---------------------
    pass
    # 保存模型（请写好保存模型的路径及名称）

    #␣
↪-----------------------------------------------------------------------------------------

    return

def evaluate_mode(val_X, val_y, save_model_path):
    """
    加载模型和评估模型
    可以实现，比如：模型训练过程中的学习曲线，测试集数据的 loss 值、准确率及混淆矩阵
等评价指标!
    主要步骤：
        1.加载模型（请填写你训练好的最佳模型），
```

```
        2.对自己训练的模型进行评估

    :param test_X: 测试集特征
    :param test_y: 测试集 target
    :param save_model_path: 加载模型的路径和名称，请填写你认为最好的模型
    :return:
    """
    # ---------------------- 实现模型加载和评估等部分的代码␣
↪----------------------
    pass

    #␣
↪--------------------------------------------------------------------------------


def main():
    """
    深度学习模型训练流程，包含数据处理、创建模型、训练模型、模型保存、评价模型等。
    如果对训练出来的模型不满意，你可以通过调整模型的参数等方法重新训练模型，直至训
练出你满意的模型。
    如果你对自己训练出来的模型非常满意，则可以提交了！
    :return:
    """
    data_path = "./dataset"  # 数据集路径
    save_model_path = "results/model.h5"  # 保存模型路径和名称
    validation_split = 0.2 # 验证集比重

    # 获取数据、并进行预处理
    train_X, train_y, val_X, val_y = processing_data(data_path,␣
↪validation_split = validation_split)

    # 创建、训练和保存模型
    model(train_X, train_y, val_X, val_y, save_model_path)

    # 评估模型
    evaluate_mode(val_X, val_y, save_model_path)
```

```
if __name__ == '__main__':
    main()
```

3.2 模型预测

注意：1. 点击左侧栏提交结果后点击生成文件则只需勾选 predict() 函数的 cell，即**【模型预测代码答题区域】**的 cell。**注意不要勾选训练模型的代码**。
2. 请导入必要的包和第三方库 (包括此文件中曾经导入过的)。3. 请加载你认为训练最佳的模型，即请按要求填写模型路径。4. predict() 函数的输入和输出请不要改动。5. 测试时记得填写你的模型路径及名称，如果采用 离线任务 请将模型保存在 **results** 文件夹下。

========== **提交 Notebook 训练模型结果数据处理参考示范** ==========

[ ]:
```
# 导入相关包
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
import numpy as np
import jieba as jb
import tensorflow as tf
from tensorflow.keras.models import load_model
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical



# ----------------------------------------------------------------------------
# 本 cell 代码仅为 Notebook 训练模型结果进行平台测试代码示范
# 可以实现个人数据处理的方式，平台测试通过即可提交代码
#  ----------------------------------------------------------------------------

vocab_json_string = ""
vocab_keras_path = "results/vocab_keras.json"
with open(vocab_keras_path) as f:
    vocab_json_string = f.read()
```

```
vocab_processor = tf.keras.preprocessing.text.
 ↪tokenizer_from_json(vocab_json_string)



def padding(text_processed, max_sequence_length=80):
    """
    数据处理，如果使用 lstm，则可以接收不同长度的序列。
    :text_processed: 不定长的 Token 化文本序列，二维 list
    :max_sequence_length: padding 大小，长句截断短句补 0
    :return 处理后的序列，numpy 格式的二维数组
    """
    res = []
    for text in text_processed:
        if len(text) > max_sequence_length:
            text = text[:max_sequence_length]
        else:
            text = text + [0 for i in range(max_sequence_length - len(text))]
        res.append(text)
    return np.array(res)
```

========== **模型预测代码答题区域** ==========

在下方的代码块中编写 **模型预测**部分的代码，请勿在别的位置作答

[ ]:
```
# ------------------------- 请加载您最满意的模型 -------------------------
# 加载模型（请加载你认为的最佳模型）
# 加载模型，加载请注意 model_path 是相对路径，与当前文件同级。
# 如果你的模型是在 results 文件夹下的 demo.h5 模型，则 model_path = 'results/demo.
 ↪h5'
model_path = None

# 加载模型，如果采用 keras 框架训练模型，则 model=load_model(model_path)
model = load_model(model_path)

# ---------------------- 模型预测，注意不要修改函数的输入输出␣
 ↪------------------------
```

```
def predict(text):
    """
    :param text: 中文字符串，新闻片段
    :return: 字符串格式的类型，如'科技'
    """

    # ----------- 实现预测部分的代码，以下样例可代码自行删除，实现自己的处理方式-----------
    labels = {0: '科技', 1: '社会', 2: '娱乐', 3: '财经', 4: '体育'}
    sen_prosessed = " ".join(jb.cut(text, cut_all=False))
    max_sequence_length = 80
    sen_prosessed = vocab_processor.texts_to_sequences([sen_prosessed])[0]
    sen_prosessed = padding([sen_prosessed], max_sequence_length)[0]
    sen_prosessed = np.array(sen_prosessed).reshape(1, -1)

    # 加载模型进行预测
    result = model.predict(sen_prosessed)
    prediction = labels[list(result[0]).index(max(result[0]))]
    # ------------------------------------------------------------------------
    return prediction
```