第9章 自然语言处理

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 人工智能基础与应用 | | 章名 | | | 自然语言处理 | |
| 教学内容 | 自然语言处理 | | | | 课时 | 2 | |
| 项目性质 | □演示性 □验证性 □设计性 √综合性 | | | | | | |
| 授课班级 |  | 授课日期 | |  | | 授课地点 |  |
| 教学目标 | 了解人工智能自然语言处理的基本概念  掌握英文语音识别项目的实现  掌握打造智能聊天机器人的实现 | | | | | | |
| 教学内容 | 1. 人工智能自然语言处理的基本概念 2. 英文语音识别 3. 打造智能聊天机器人 | | | | | | |
| 教学重点 | 基于speech\_commands实现英文语音识别 | | | | | | |
| 教学难点 | 基于speech\_commands实现英文语音识别 | | | | | | |
| 教学准备 | 装有Python的计算机  教学课件PPT  教材：《人工智能基础与应用（微课版）》 | | | | | | |
| 作业设计 |  | | | | | | |

教学过程

|  |  |
| --- | --- |
| **教学环节** | **教学内容与过程**  **（教学内容、教学方法、组织形式、教学手段）** |
| **课前组织** | 做好上课前的各项准备工作（打开计算机、打开课件、打开软件、打开授课计划、教案等），吸引学生注意力。 |
| **课程说明** | 【课前说明】  从自然语言处理的产生和发展引入本节课内容。  【目的】  使学生从了解本节课的学习目标、学习重点、考评方式等方面明确课程学习的要求和目标。 |
| **课程内容描述** | 9.1 人工智能自然语言处理介绍  自然语言处理是从20世纪50年代开始发展的，其最先在机器翻译领域得到发展。1954年的乔治敦实验（Georgetown-IBM Experiment）将60多句俄文自动翻译成英文，之后问答系统的发展也有了进展。20世纪60年代，出现了句法分析、语义分析、逻辑推理相结合的SHRDLU自然语言系统。直到20世纪80年代初期，多数自然语言处理系统都是以一套复杂的、人工制定的规则为基础形成的。  从20世纪80年代末期开始，语言处理引进了机器学习的算法，自然语言处理产生革新。近年来，深度学习技巧纷纷出炉，在自然语言处理方面获得了尖端的成果。下表列举了自然语言处理的部分范畴。  自然语言处理的部分范畴   |  |  | | --- | --- | | 技术名称 | 注释 | | 语音识别 | 机器通过识别和理解过程把语音信号转变为相应的文本或命令 | | 语音合成 | 通过机械的、电子的方法产生人造语音的技术 | | 中文自动分词 | 使用机器自动对中文文本进行词语的切分，像英文那样使得中文句子中的词之间以空格标识 | | 词性标注 | 将语料库内单词的词性按其含义和上下文内容进行标记的文本数据处理 | | 句法分析 | 对句子中的词语语法功能进行分析 | | 自然语言生成 | 使机器具有人一样的表达和写作能力 | | 文本分类 | 机器对文本集（或其他实体）按照一定的分类体系或标准进行自动分类标记 | | 问答系统 | 用准确、简洁的自然语言回答用户用自然语言提出的问题 | | 机器翻译 | 利用机器将一种自然语言（源语言）转换为另一种自然语言（目标语言）的过程 |   在自然语言处理的研究过程中，有一些难点是需要攻克的。例如，在口语中，词与词通常是连贯的，它们之间没有边界；很多词不仅仅只有一个意思；在做语音处理时会出现口音问题；文本处理时书写不规范等。  9.2 英文语音识别 9.2.1 项目介绍 英文语音识别项目以Google的speech\_commands英文单词语音识别项目为基础，并做了一些修改，以更好地使用。该项目构建可以识别10个不同单词的基本语音识别网络，10个单词分别为“yes”“no”“up”“down”“left”“right”“on”“off”“stop”“go”。实际的语音和音频识别系统要复杂得多，但就像基于MNIST数据集学习图像识别一样，这个基本语音识别网络能够帮助读者了解所涉及的一些基本技术。  可以通过多种方法构建用于处理音频的神经网络模型，其中包括递归网络或扩张（带洞）卷积。本项目基于*Convolutional Neural Networks for Small-footprint Keyword Spotting*这篇论文中介绍的架构，这种架构相对简单、可快速训练，并且易于理解。由于音频本身是一段时间内的一维连续信号，而不是二维空间信号，所以这里采用的是卷积神经网络。本项目定义了一个语音字词应该符合的时间范围，并将这段时间内的音频信号转换成图像。将传入的音频样本分成小段（时长仅为几毫秒）并计算一组频段内频率的强度，一段音频内的每组频率强度为数字向量，这些向量按时间顺序排列，形成一个二维数组，该数组可被视为单通道图像，称为声谱图。  具体做法是：将输入的语音处理成能够读取的数据，即将其转换成一组梅尔频率倒谱系数（Mel-Frequency Cepstral Coefficients，MFCC）。MFCC也是一种二维单通道表示法，因此也可将其视为图像，图像会输送到多层卷积神经网络中进行处理，并且在卷积神经网络处理后接入全连接层，再经过softmax()函数完成分类，实现区分不同词汇的功能。 9.2.2 训练模型 在Python目录下新建NLP目录，在NLP目录下新建speech目录，在speech目录下新建speech\_commands项目目录并将源码下载到该项目目录下。  在开始训练过程之前需要获取数据集。有两种方式可获取数据集：一种是手动下载语音指令数据集，下载后的数据集解压后需要放在speech\speech\_commands\tmp下；另一种是运行训练程序后脚本会自动下载该数据集。语音指令数据集中包括超过105000个WAVE音频文件，音频内容是30个不同的字词，这些数据由Google收集，并依据知识共享许可协议（Creative Commons License，CC协议）发布，读者可以提交5min的录音来帮助改进该数据集。  运行speech\_commands目录下的train.py，在训练过程中会打印出训练日志，如图所示。    打印出的训练日志  以第一步的打印日志为例：  训练步Step #1，共设置了18000个训练步，可以通过该信息观察训练进程；学习率rate 0.001000，刚开始学习率比较大，为0.001，训练后期会减小到0.000 1；准确率accuracy 5.0%，表示训练在本步中预测正确的类别数量，该值通常会有较大的波动，但会随着训练的进行总体有所提高，准确率的值会在0%～100%之间波动，始终不会超过100%；一般而言，这个值越高，训练出来的模型越好；损失函数的值cross entropy 2.560931，表示训练过程的损失函数的结果，它是一个得分，通过将当前训练运行的得分向量与正确标签进行比较计算得出，该得分应在训练期间呈下滑趋势，但是其并不一定是平滑地下滑。  在训练过程中，每100步会保存一次模型，第100步打印的日志如图所示。    第100步打印的日志  由于训练过程比较久，如果是CPU的TensorFlow训练，需要十几个小时，所以可能一次无法完成训练。当中途有其他事情的时候可以先中止训练，下次训练时先检查上次训练到哪一步了，查找上次保存的检查点，然后将“--start\_checkpoint=tmp/speech\_commands\_train/conv.ckpt-100”用作命令行参数重启该脚本，从该点继续训练，命令行参数中的100是上次中止训练的最近的步数，同时也包含在检查点文件名中。  每训练400步，会生成混淆矩阵（Confusion Matrix）。在监督学习中，混淆矩阵可作为可视化工具，在无监督学习中其一般被称为匹配矩阵。混淆矩阵是通过将实际的分类与预测分类相比较计算出来的。  由于该语音识别项目中生成的混淆矩阵相对比较复杂，所以这里先给出一个实例来进行概念理解，如现有一个猫狗分类的二分类问题，有10只猫和8只狗，在预测之后计算出的混淆矩阵如表所示。  预测之后计算出的混淆矩阵   |  |  |  |  | | --- | --- | --- | --- | | 混淆矩阵 | | 预测值 | | | 猫 | 狗 | | 真实标签 | 猫 | 7 | 3 | | 狗 | 0 | 8 |   由上表可知，混淆矩阵中数字的每一行都是真实标签（即第一行的7和3，第二行的0和8），数字的每一列是预测值（即第一列的7和0，第二列的3和8）。实际上猫的数量为7+3=10，狗的数量为0+8=8，但是预测的结果中，猫的数量为7+0=7，狗的数量为3+8=11。在每次输出混淆矩阵之后会打印该模型在验证集上的准确率（Validation accuracy），该准确率是由混淆矩阵从左上到右下的对角线上值的和除以总体数据量*N*得到的，此时的预测准确率可以计算为（7+8）÷18×100%≈83.3%。在理想情况下，该准确率应该接近于训练准确率，如果训练准确率有所提高，但验证准确率没有提高，则表明存在过拟合，模型只学习了有关训练数据集的信息，而在验证或者预测数据集上表现不佳，所以混淆矩阵也可以看作预测准确率的可视化。  下图是训练到400步时生成的混淆矩阵。    训练到400步时生成的混淆矩阵  在语音识别项目中，标签分别为“silence”“unknown”“yes”“no”“up”“down”“left”“right”“on”“off”“stop”“go”。  每一行都代表真实的标签，如第一行是“silence”（无声的）的所有音频片段，第二行是“unknown”（未知字词的）的所有音频片段，第三行是“yes”的所有音频片段，以此类推。  每一列都代表预测的值，第一列代表预测为“silence”（无声的）的所有音频片段，第二列代表预测为“unknown”（未知字词的）的所有音频片段，第三列代表预测为“yes”的所有音频片段，以此类推。  训练到18000步时的混淆矩阵如图所示，可以看出它与训练到400步时混淆矩阵的区别。    训练到18000步时的混淆矩阵  所以，完美模型生成的混淆矩阵，除了从左上到右下的对角线上的条目外，所有其他条目几乎都接近0。混淆矩阵有助于了解模型最容易在哪些方面混淆，确定问题所在后，可以通过添加更多数据或清理类别来解决问题。所以，混淆矩阵要比直接打印准确率和损失函数的值更能够准确地找到网络的问题。  在整个训练过程中，使用TensorBoard可以很好地观察训练进度。默认情况下，脚本会将事件保存到tmp/retrain\_logs，可以在命令行运行以下命令：  tensorboard --logdir tmp/retrain\_logs  模型进度的图表如图所示。  9-5  模型进度的图表  训练完成后，准确率介于85%～90%之间。在训练完成后进行模型转换，在命令行运行以下命令：  python speech\_commands/freeze.py \  --start\_checkpoint=speech\_commands/tmp/speech\_commands\_train/conv.ckpt-18000 \  --output\_file=speech\_commands/tmp/my\_frozen\_graph.pb  运行以上命令会在tmp目录下生成my\_frozen\_graph.pb模型文件。 9.2.3 测试效果 修改测试代码使其可以识别自己录制的语音，将label\_wav.py源码进行修改，修改为以下内容。  from \_\_future\_\_ import absolute\_import  from \_\_future\_\_ import division  from \_\_future\_\_ import print\_function  import argparse  import sys  import tensorflow as tf  from tensorflow.contrib.framework.python.ops import audio\_ops as contrib\_audio  FLAGS = None  def load\_graph(filename):  """Unpersists graph from file as default graph."""  with tf.gfile.FastGFile(filename, 'rb') as f:  graph\_def = tf.GraphDef()  graph\_def.ParseFromString(f.read())  tf.import\_graph\_def(graph\_def, name='')  def load\_labels(filename):  """Read in labels, one label per line."""  return [line.rstrip() for line in tf.gfile.GFile(filename)]  def run\_graph(wav\_data, labels, input\_layer\_name, output\_layer\_name,  num\_top\_predictions):  """Runs the audio data through the graph and prints predictions."""  with tf.Session() as sess:  # 将音频数据作为输入输入到图形  # 预测将包含一个二维数组，其中一个维表示输入图像的数量，而另一个维按类别分类  softmax\_tensor = sess.graph.get\_tensor\_by\_name(output\_layer\_name)  predictions, = sess.run(softmax\_tensor, {input\_layer\_name: wav\_data})  # 按置信度排序显示标签  top\_k = predictions.argsort()[-num\_top\_predictions:][::-1]  for node\_id in top\_k:  human\_string = labels[node\_id]  score = predictions[node\_id]  print('%s (score = %.5f)' % (human\_string, score))  return 0  def label\_wav(wav, labels, graph, input\_name, output\_name, how\_many\_labels):  """Loads the model and labels, and runs the inference to print predictions."""  if not wav or not tf.gfile.Exists(wav):  tf.logging.fatal('Audio file does not exist %s', wav)  if not labels or not tf.gfile.Exists(labels):  tf.logging.fatal('Labels file does not exist %s', labels)  if not graph or not tf.gfile.Exists(graph):  tf.logging.fatal('Graph file does not exist %s', graph)  labels\_list = load\_labels(labels)  # load graph, which is stored in the default session  load\_graph(graph)  with open(wav, 'rb') as wav\_file:  wav\_data = wav\_file.read()  run\_graph(wav\_data, labels\_list, input\_name, output\_name, how\_many\_labels)  def main(\_):  """Entry point for script, converts flags to arguments."""  label\_wav('./00b01445\_nohash\_0.wav','./speech\_commands/tmp/speech\_ commands\_train/\  conv\_labels.txt', './speech\_commands/tmp/my\_frozen\_graph.pb', 'wav\_data:0', 'labels\_softmax:0', 3)  def speech\_rcgnz():  tf.app.run(main=main)  识别自己的语音需要提前录制语音，安装pyaudio第三方库后可以使用Python在计算机上录音。在交互界面上输入“pip install pyaudio”安装pyaudio。  安装完成后，在speech目录下新建main.py文件来完成语音录制、语音识别的调用。  import wave  from pyaudio import PyAudio,paInt16  from sys import path  # 将存放module的路径添加进来  path.append(r'./speech\_commands/')  import label\_wav  import os  os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  framerate=16000  NUM\_SAMPLES=2000  channels=1  sampwidth=2  TIME=2  def save\_wave\_file(filename,data):  '''save the date to the wavfile'''  wf=wave.open(filename,'wb')  # 声道  wf.setnchannels(channels)  # 采样字节  wf.setsampwidth(sampwidth)  # 采样频率  wf.setframerate(framerate)  wf.writeframes(b"".join(data))  wf.close()  def my\_record():  pa=PyAudio()  stream=pa.open(format = paInt16,channels=1,  rate=framerate,input=True,  frames\_per\_buffer=NUM\_SAMPLES)  my\_buf=[]  count=0  while count<TIME\*4:# 控制录音时间  string\_audio\_data = stream.read(NUM\_SAMPLES)  my\_buf.append(string\_audio\_data)  count+=1  print('0.125s is passing！')  save\_wave\_file('00b01445\_nohash\_0.wav',my\_buf)  stream.close()  chunk=2014  def play():  wf=wave.open(r"00b01445\_nohash\_0.wav",'rb')  p=PyAudio()  stream=p.open(format=p.get\_format\_from\_width(wf.getsampwidth()),channels=  wf.getnchannels(),rate=wf.getframerate(),output=True)  while True:  data=wf.readframes(chunk)  if data=="":break  stream.write(data)  stream.close()  p.terminate()  if \_\_name\_\_ == '\_\_main\_\_':  print('可以识别的单词包括:')  print('yes,no,up,down,left,right,on,off,stop,go')  input("按Enter键开始录音:")  my\_record()  print('结果：')  label\_wav.speech\_rcgnz()  play()  执行该代码后，调用录音功能并打印提示信息，如图所示。  9-6  调用录音功能并打印提示信息  录音时间为1s，按下键盘的Enter键后说出要识别的单词，这里说了“left”这个单词，等待识别完毕，识别完成后输出识别结果最可能的3个标签，识别结果如图所示。  9-7  识别结果  “left”得分最高的识别结果为“left”，识别正确。  9.3 打造智能聊天机器人 9.3.1 seq2seq的机制原理 seq2seq模型，全称为Sequence to Sequence，它是一种通用的编码器—解码器框架，可用于机器翻译、文本摘要、会话建模、图像字幕等场景中。  前面的章节已经介绍过Sequence-to-Sequence模型，即Encoder-Decoder模型。在实际聊天系统中，解码器和编码器一般都采用RNN模型和LSTM模型，编码器和解码器之间的唯一联系就是一个固定长度的上下文向量***c***，编码器要将整个序列的信息压缩进一个固定长度的向量中去。这样做有两个弊端，一是语义向量无法完全表示整个序列的信息，二是先输入的内容携带的信息会被后输入的信息稀释，输入序列越长，这个现象就越严重，这就使得解码时没有获得输入序列足够的信息，使解码时的准确率打折扣。  为了解决上述问题，在seq2seq出现后，Attention模型被提出。该模型在产生输出的时候，会生成一个注意力范围来表示接下来输出的时候要重点关注输入序列的哪些部分，然后根据关注的区域产生下一个输出，如此反复。Attention和人的一些行为特征有一定相似之处，人在读一段话的时候，通常只会重点注意具有信息量的词，而非全部词，人会赋予每个词不同的注意力权重。Attention模型虽然提高了模型的训练难度，但提升了文本生成的效果。模型的示意图如图所示。  9-8  模型的示意图 9.3.2 实践 本小节制作一个智能聊天机器人，数据集可以在网上下载或者自己生成，网上下载的数据集可以进行修改，数据集应该是一个或多个以换行符分隔的文本文件。对话的每一行都应以“>”开头，并以换行符结尾，并将文本文件放在项目的data目录下。  由于网络比较复杂且参数量较多，该网络在GPU版的TensorFlow下需要训练几天，所以可以直接运行预训练的模型。  将模型下载下来之后，解压缩到名为“reddit”的文件夹中，将该文件夹放入该项目的“models”目录中。  执行：  python chatbot.py  等待片刻，打印出以“>”开头的行，这时需要读者自己输入想说的话，没有“>”的行是聊天机器人的响应，对话结果如下。  > Hi, how are you?  Hello there, I'm doing fine. What about you?  > I'm very good.  Fair enough.  > I love you.  I love you too.  > I love you.  You're the best.  > Where are you from?  I'm from the south.  > Goodbye!  What do you mean?  > Goodbye!  Goodbye!  很多时候，回答是不尽如人意的。这仅仅是一个对话的简单演示，感兴趣的读者可以将该数据集替换为中文，制作一个可以中文对话的智能聊天机器人。 |
| **总结评价** | 本节课主要介绍了TensorFlow在自然语言处理中的应用，先对英文的语音识别进行了分析以及实践，讲解了如何训练并实现一个单词的语音识别网络，在此基础上可以对其他单词或者中文进行识别；之后讲述了智能聊天机器人中运用的相关技术，并解释了Attention模型的优势；最后，实践了一个简单的聊天机器人。 |