从代码学AI ——情感分类(LSTM)

Jerry_wl (/u/2c7269783d3f) ＋关注
2017.08.24 09:21* 字数 2193 阅读 2212 评论 0 喜欢 4
(/u/2c7269783d3f)

前言

本篇文章会从代码的角度说明如何基于TFlearn使用LSTM进行文本的情感分类。如果对于TFLearn和LSTM都不熟悉，没有关系，先硬着头皮将代码看下(使用LSTM对IMDB数据集进行情感分类 (https://link.jianshu.com?t=https://github.com/tflearn/tflearn/blob/master/examples/nlp/lstm.py))。从代码的角度看都是很简洁的，所以即使不熟悉，多看看代码，当代码已经熟练于心了，后面如果有一天你漠然回首理解了其中的不解后，你的记忆更加深刻。所以不懂、不熟悉没关系，坚持下去就回明白的。

由于实例代码使用的是IMDB数据集，所以这里会优先介绍 一下这个数据集。

IMDB数据集

该数据集包含了电影的评论以及评论对应的情感分类的标签(0,1分类)。作者的初衷是希望该数据集会成为情绪分类的一个基准。这里介绍该数据集如何生成的以及如何使用提供的文件。

核心数据集包含了5万条评论数据，这些数据被均分成训练集和测试集(训练和测试集各2.5万)。标签也是均衡分布的(正负样本各2.5万)。也提供了5万条无标签数据，以用于无监督学习。

在数据集中，每个电影最多收集30条评论，因为同一个电影的评论往往具有相关性。同时训练集和测试集采集的是不同的电影，所以尝试去记住和电影强相关的词汇以及相关的标签是不会取得显著的提升效果的。

在训练和测试集中，负面结果的分值<=4,正面结果的分值>=10.中性的评论没有包含在测试和训练集合中。在无监督的数据集中包含任意评分的评论。

对于下载下来的数据集的文件结构大致如下：

有两个顶级文件夹[train/, test/],对应训练集和测试集。每个都包含了[pos/, neg/]目录，在这些文件夹中，评论数据以如下方式存储：[[id]_[rating].txt]。这里id表示唯一性ID，rating表示评分，例如[test/pos/200_8.txt]表示正面评论，id是200，评分是8分。
无监督数据集中[train/unsup/]所有的评分都是0，因为所有的评分都被省略了。

数据集中也包含了每个评论对应电影的评论页面的URL，由于电影的评论数据是动态变化的，所以不能指定评论的URL，只能指定电影评论页面的URL。评论文件在如下文件中：
[urls_[pos, neg, unsup].txt]

对于评论的数据文件，数据集中已经包含了训练好的词袋模型(BoW).这些数据存储在.feat文件中。

每个.feat文件都是LIBSVM格式，一种用于标记数据的ascii的稀疏向量格式。
这些文件中的特征索引从0开始，且特征索引对应的词汇对应着[imdb.vocab]中相应的词汇。所以一个在.feat文件中以0:7的形式表示[imdb.vocab]中的第一个单词,在该评论中出现7次

LIBSVM相关资料参见:LIBSVM (https://link.jianshu.com?t=http://www.csie.ntu.edu.tw/~cjlin/libsvm/)

数据集中也包含了一个[imdbEr.txt]文件，这里存储了[imdb.vocab]中每个词的情感评分。预期评级是了解数据集中单词的平均极性的好方法。

数据集介绍就到这里，下面开始代码解读。

代码解读

```
# -*- coding: utf-8 -*-
"""

https://www.tensorflow.org/versions/master/programmers_guide/embedding

https://github.com/tflearn/tflearn/blob/master/tflearn/datasets/imdb.py

采用LSTM进行情感分类的实例，数据集采用IMDB数据集


LSTM论文链接:

    http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf

IMDB数据集链接

    http://ai.stanford.edu/~amaas/data/sentiment/

"""
```

引入相关的模块和方法

```
from __future__ import division, print_function, absolute_import

import tflearn
from tflearn.data_utils import to_categorical, pad_sequences
from tflearn.datasets import imdb
```

获得测试和训练数据

```
# 导入IMDB的数据集,n_words表示构建词向量的时候，考虑最常用的10000个词,valid_portion表示训练过程中
# load_data的结果是train[0][0:]表示训练数据,train[1][0:]表示对应的标签，即train[0]是训练矩阵，tr
train, test, _ = imdb.load_data(path='imdb.pkl', n_words=10000,valid_portion=0.1)
# 获得训练集对应的数据和标签
trainX, trainY = train
# 获得测试集对应的数据和标签
testX, testY = test
```

数据预处理

```
# 进行数据处理，补充长度，长度全为100，不足的0补位,每条训练数据都变成100位的向量
trainX = pad_sequences(trainX, maxlen=100, value=0.)
testX = pad_sequences(testX, maxlen=100, value=0.)
# 将数据的打标转化为向量 原来是0->[1,0],原来是1->[0,1]
trainY = to_categorical(trainY, nb_classes=2)
testY = to_categorical(testY, nb_classes=2)
```

网络构建

```
# 构建网络
# 1.先指定输入数据数据量大小不指定，和placeholder类似，在运行时指定，每个向量100维  <tf.Tensor 'I
net = tflearn.input_data([None, 100])
# 2.进行词嵌套，相当于将离散的变为连续的，输入词词有10000个ID，每个ID对应一个词，将每个词变为一个12
# <tf.Tensor 'Embedding/embedding_lookup:0' shape=(?, 100, 128) dtype=float32>

#Embedding可以将离散的输入应用于机器学习处理方法中。传统的分类器和神经网络一般来讲更适合处理连续的向
#如果有些离散对象自然被编码为离散的原子，例如独特的ID，它们不利于机器学习的使用和泛化。
#可以理解embedding是将非矢量对象转换为利于机器学习处理的输入。
net = tflearn.embedding(net, input_dim=10000, output_dim=128)
# 3.LSTM，输出<tf.Tensor 'LSTM/LSTM/cond_199/Merge:0' shape=(?, 128) dtype=float32>
net = tflearn.lstm(net, 128, dropout=0.8)
# 4.全连接层  输出<tf.Tensor 'FullyConnected/Softmax:0' shape=(?, 2) dtype=float32>
# 就是将学到的特征表示映射到样本标记空间
net = tflearn.fully_connected(net, 2, activation='softmax')
# 5.回归层 指定优化方法、学习速率(步长)、以及损失函数
net = tflearn.regression(net, optimizer='adam', learning_rate=0.001,loss='categorical_crosse
```

构建模型并训练

```
# 构建深度模型，tensorboard需要的日志文件存储在/tmp/tflearn_logs中
model = tflearn.DNN(net, tensorboard_verbose=0)
# 训练模型，指定训练数据集、测试数据集
model.fit(trainX, trainY, validation_set=(testX, testY), show_metric=True,batch_size=32)
```

到这里整个代码就结束了,这里有两个地方需要说明一下，一个是embedding，一个是fully_connected，分别表示词嵌套和全连接。这里对这两部分简要说下，不会进行详尽的公式推导和阐释。

首先说说这里的embedding.

在注释部分已经注释的比较明确了，其目的就是将要表示的东西进行向量化表示。原来每个字用一个ID表示，这样能表示的信息太少了，不能够表达词所在语料内更多的意思,比如和那个词更相近。通过词嵌套将一个单一的ID表示为一个128纬度(此处是128)的向量，能够表达更多的意思。词嵌套是向量化的一个重要手段，这个技巧一定要掌握的。

再聊聊这里的fully_connected

全连接层（fully connected layers，FC），在整个神经网络中起到类似于“分类器”的作用。如果说卷积层、池化层和激活函数层等操作是将原始数据映射到隐层特征空间的话，那么全连接层则起到将学到的“分布式特征表示”映射到样本标记空间的作用。在代码中tflearn.fully_connected(net, 2,activation='softmax')，这里的第二个参数2，表示的就是输出神经元的个数，即训练集中对应的打标的向量，也就是说将学习到的分布式特征转化为一个只包含两个元素的向量。

全连接层的每一个结点都与上一层的所有结点相连，用来把前边提取到的特征综合起来。由于其全相连的特性，一般全连接层的参数也是最多的。不负责任的讲，全连接层一般由两部分组成，即线性部分和非线性部分。线性部分主要做线性转换，输入用X表示，输出用Z表示。

线性部分的运算方法基本上就是线性加权求和的感觉，如果对于一个输入向量

```
x=[x_0,x_1,...x_n]^T,
```

线性部分的输出向量是

```
z=[z_0,z_1,z_2,...z_m]^T,
```

那么线性部分的参数就可以想象一个m*n的矩阵W，再加上一个偏置项

```
b=[b_0,...b_m]^T
```

于是有：


```
W*x+b=z
```

对于非线性部分，那当然是做非线性变换了，输入用线性部分的输出Z表示，输出用Y表示，假设用sigmoid作为非线性激活，那么有

```
Y = sigmoid(Z)
```

那么为什么要有非线性部分呢？个人理解，其一是作数据的归一化。不管前面的线性部分做了怎样的工作，到了非线性这里，所有的数值将被限制在一个范围内，这样后面的网络层如果要基于前面层的数据继续计算，这个数值就相对可控了。其二就是打破之前的线性映射关系。如果全连接层没有非线性部分，只有线性部分，我们在模型中叠加多层神经网络是没有意义的，我们假设有一个2层全连接神经网络，其中没有非线性层，那么对于第一层有：

```
W^0*x^0+b^0=z^1
```

对于第二层有：

```
W^1*z^1+b^1=z^2
```

两式合并，有：

```
W^1*(W^0*x^0+b^0)+b^1=z^2
W^1*W^0*x^0+(W^1*b^0+b^1)=z^2
```

所以我们只要令：

```
W^{0'}=W^1*W^0 ,
b^{0'}=W^1*b^0+b^1,
```

就可以用一层神经网络表示之前的两层神经网络了。所以非线性层的加入，使得多层神经网络的存在有了意义。

关于非线性激活常用的函数，以及什么样的激活函数才是好的激活函数后面会专门介绍，这里不再赘述。

我们在看一下tflearn中fully_connected的函数的参数解释，当然tensorflow中也有相关函数，这里暂不进行注解。

tflearn.layers.core.fully_connected

```
incoming: 输入Tensor，维度不小于2

n_units: 整型，表示输出神经元个数

activation: 激活函数，输入激活函数的名称或者函数定义(自定义非线性激活函数)，默认值:'linear',参见 t

bias: 布尔值，表示是否使用偏置项

weights_init: 初始化权重W的参数，String 或者一个Tensor,默认是truncated_normal，参见tflearn.init

bias_init: 初始化偏置项，String或Tensor,默认'zeros'。参见tflearn.initializations

regularizer: 规范化函数，String或者一个Tensor,对权重W进行规范化操作，默认不进行，参见tflearn.regu

weight_decay: 浮点数，规范化方法的衰减参数，默认值 0.001.

trainable: 可选参数，布尔值，如果为True，那么变量将会加到图模型中。同tf.Variable的trainable

restore: bool. If True, this layer weights will be restored when loading a model.

reuse: 可选参数，布尔类型，如果指定了Scope且当前参数置为True,那么该layer的变量可复用

scope: 可选参数，String类型，定义layer的Scope(指定Scope可以在不同层间共享变量，但需要注意Scope可被

name: 可选值，表示该层的名称，默认值: 'FullyConnected'.
```


总结

基于tflearn进行深度学习相关功能的实现，要比基于原生的tensorflow要简单的多，封装的比较干净。本篇只介绍了使用LSTM进行NLP的相关任务处理，读者完全可以自行下载代码进行改造，这里希望读者能够了解embedding和fully_connected的意思，知道其作用，如果希望深入了解，可以参考相关论文和他人注解的blog，本篇就写到这里，欢迎拍砖。

从代码的角度看AI，原来也没那么复杂~

Jerry_wl (/u/2c7269783d3f) ♂
写了 12558 字，被 19 人关注，获得了 23 个喜欢
(/u/2c7269783d3f)

+关注

2015年毕业于哈尔滨工业大学 本科在企业智能实验室，进行情景感知相关学习 硕士在自然计算实验室，进...

登录 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-comment-form) 后发表评论

(/apps/redirect?utm_source=side-banner-click)

评论

智慧如你，不想发表一点想法 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-nocomments-text)咩~

被以下专题收入，发现更多相似内容

从代码走进深度学习 (/c/9b785e9ff8df?utm_source=desktop&utm_medium=notes-included-collection)

(/p/ea922866e3be?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

AlexNet论文翻译——中英文对照 (/p/ea922866e3be?utm_campaign=mal…

声明：作者翻译论文仅为学习，如有侵权请联系作者删除博文，谢谢！ 翻译论文汇总：https://github.com/SnailTyan/deep-learning-papers-translation ImageNet Classification with Deep Co...

SnailTyan (/u/7731e83f3a4e?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/15411de409f1?

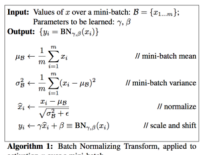
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

使用 TensorFlow 做文本情感分析 (/p/15411de409f1?utm_campaign=mal…

使用 TensorFlow 做文本情感分析 本文将通过使用TensorFlow中的LSTM神经网络方法探索高效的深度学习方法。 作者： Adit DeshpandeJuly 13, 2017翻译来源：https://www.oreilly.com/learning/perf...

Datartisan数据工匠 (/u/ad75474d9e73?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/2bc6dab16cfe?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

Batch Normalization论文翻译——中英文对照 (/p/2bc6dab16cfe?utm_ca…

文章作者：Tyan博客：noahsnail.com | CSDN | 简书 声明：作者翻译论文仅为学习，如有侵权请联系作者删除博文，谢谢！ 翻译论文汇总：https://github.com/SnailTyan/deep-learning-papers-translatio...

SnailTyan (/u/7731e83f3a4e?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/f054f8daec68?
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

多图 | 从神经元到CNN、RNN、GAN…神经网络看本…

作者 | FJODOR VAN VEEN 编译 | AI100（ID：rgznai100） 在深度学习十分火热的今天，不时会涌现出各种新型的人工神经网络，想要实时了解这些新型神…

AI科技大本营 (/u/da69420ec62d?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/apps/redirect?
utm_source=side-banner-click)

(/p/d15a023b8ab6?
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

你已在我深深地脑海里 (/p/d15a023b8ab6?utm_campaign=maleskine&ut…

不知从何时起，期待周五，不再是因为马上又可以周末happy了，而是因为等待一周，又可以看到你笑呵呵的脸，听到你磁性的开场白： 历史不是镜子，历史是精子，牺牲亿万，才有一个活到今天人生不是故事，…

XWAY (/u/8ed8de89ed21?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

你若不动，风又奈何 (/p/62bd3f7191b0?utm_campaign=maleskine&utm…

岁月不停变换，行色匆匆，斑斑驳驳……或温暖或悲凉，丝丝缕缕无不渗入我们的生活，染上七彩的喜怒哀乐。 每天每天，生活就像变着花样儿的风，极进玩弄的能事，舞动着你的发丝，扰乱着你的心魄。 你可否…

释牧萨拉 (/u/9c80c6c2372f?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

素食者联盟 (/p/c2e3a4dadbb1?utm_campaign=maleskine&utm_conten…

不知从何时开始，肉类就成为了人类餐桌上不可缺少的菜。要知道，肉从古代到近代，一直都是副食啊。只有过年过节的时候，才会吃肉庆祝，平时不怎么吃肉的。那肉类究竟是怎么一步步侵略我们的餐桌呢？ 要…

谢小芬 (/u/015ab23e97ad?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/de736faf131a?
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

信仰深处的凝视，让他为“造佛“倾注了全部的温柔 (/p/de736faf131a?utm_…

用一生的努力去建设的佛教艺术馆，凝聚的不仅仅是精湛的技艺，还有坚定的信仰。 宁式佛像|宁式佛像风格起源于西元时晋代的余姚陆埠，在唐宋、明清时期备受推广，多见于敦煌石窟、龙门石窟等处。宁式佛像…

谷雨CHN (/u/84ffc7c63f11?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

你，才是自己真正的贵人 (/p/b27610774d20?utm_campaign=maleskine&…

(文章首发写手圈)1.昨天中午，参加了杨红丽的婚宴，让我感触颇深。 在我们这个城市。二婚的话，很少有典礼，一般都在晚上宴请一下亲朋好友即可。而杨红丽的婚宴不仅排场，还很讲究。足以可见新郎的用心…

三月的木棉花 (/u/4fcd23514460?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)