WTF Daily Blog

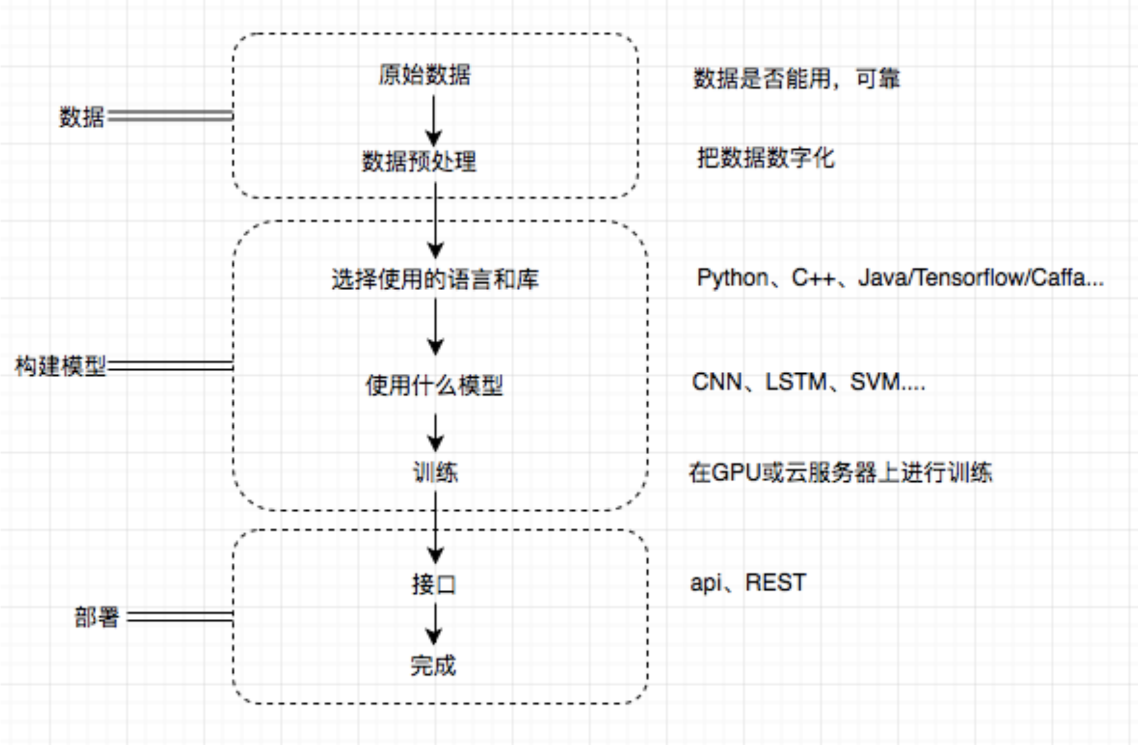
斗大的熊猫

TensorFlow练习2: 对评论进行分类

本帖是前一贴的补充：

1. 使用大数据，了解怎么处理数据不能一次全部加载到内存的情况。如果你内存充足，当我没说
2. 训练好的模型的保存和使用
3. 使用的模型没变，还是简单的feedforward神经网络
4. 如果你要运行本帖代码，推荐使用GPU版本或强大的VPS，我使用小笔记本差点等吐血

在正文开始之前，我画了一个机器学习模型的基本开发流程图：



使用的数据集

使用的数据集：http://help.sentiment140.com/for-students/ (情绪分析)

数据集包含1百60万条推特，包含消极、中性和积极tweet。不知道有没有现成的微博数据集。

数据格式：移除表情符号的CSV文件，字段如下：

- 0 – the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
- 1 – the id of the tweet (2087)
- 2 – the date of the tweet (Sat May 16 23:58:44 UTC 2009)
- 3 – the query (lyx). If there is no query, then this value is NO_QUERY.
- 4 – the user that tweeted (robotickilldozr)
- 5 – the text of the tweet (Lyx is cool)

training.1600000.processed.noemoticon.csv（238M）
testdata.manual.2009.06.14.csv（74K）

数据预处理

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

import pickle
import numpy as np
import pandas as pd
from collections import OrderedDict


org_train_file = 'training.1600000.processed.noemoticon.csv'
org_test_file = 'testdata.manual.2009.06.14.csv'

# 提取文件中有用的字段
def usefull_filed(org_file, output_file):
    output = open(output_file, 'w')
    with open(org_file, buffering=10000, encoding='latin-1') as f:
        try:
            for line in f:                # "4","2193601966","Tue Jun
                line = line.replace('"', '')
                clf = line.split(',')[0]   # 4
                if clf == '0':
                    clf = [0, 0, 1]  # 消极评论
                elif clf == '2':
                    clf = [0, 1, 0]  # 中性评论
                elif clf == '4':
                    clf = [1, 0, 0]  # 积极评论

                tweet = line.split(',')[-1]
                outputline = str(clf) + ':%:%:%:' + tweet
                output.write(outputline)  # [0, 0, 1]:%:%:%: that's a
        except Exception as e:
            print(e)
    output.close()  # 处理完成，处理后文件大小127.5M

usefull_filed(org_train_file, 'training.csv')
usefull_filed(org_test_file, 'tesing.csv')

# 创建词汇表
def create_lexicon(train_file):
    lex = []
    lemmatizer = WordNetLemmatizer()
    with open(train_file, buffering=10000, encoding='latin-1') as f:
        try:
            count_word = {}  # 统计单词出现次数
            for line in f:
                tweet = line.split(':%:%:%:')[1]
                words = word_tokenize(line.lower())
                for word in words:
                    word = lemmatizer.lemmatize(word)
                    if word not in count_word:
                        count_word[word] = 1
                    else:
                        count_word[word] += 1

            count_word = OrderedDict(sorted(count_word.items(), key=l
            for word in count_word:
                if count_word[word] < 100000 and count_word[word] > 1
                    lex.append(word)
        except Exception as e:
            print(e)
    return lex

lex = create_lexicon('training.csv')

with open('lexcion.pickle', 'wb') as f:
    pickle.dump(lex, f)


"""
# 把字符串转为向量
def string_to_vector(input_file, output_file, lex):
    output_f = open(output_file, 'w')
    lemmatizer = WordNetLemmatizer()
    with open(input_file, buffering=10000, encoding='latin-1') as f:
        for line in f:
            label = line.split(':%:%:%:')[0]
            tweet = line.split(':%:%:%:')[1]
            words = word_tokenize(tweet.lower())
            words = [lemmatizer.lemmatize(word) for word in words]

            features = np.zeros(len(lex))
            for word in words:
                if word in lex:
                    features[lex.index(word)] = 1  # 一个句子中某个词可能

            features = list(features)
            output_f.write(str(label) + ":" + str(features) + '\n')
    output_f.close()


f = open('lexcion.pickle', 'rb')
lex = pickle.load(f)
f.close()

# lexcion词汇表大小112k,training.vec大约112k*1600000  170G  太大，只能边转
# string_to_vector('training.csv', 'training.vec', lex)
# string_to_vector('tesing.csv', 'tesing.vec', lex)
"""
```

上面代码把原始数据转为training.csv、和tesing.csv，里面只包含label和tweet。lexcion.pickle文件保存了词汇表。

> *如果数据文件太大，不能一次加载到内存，可以把数据导入数据库*
>
> *Dask可处理大csv文件*

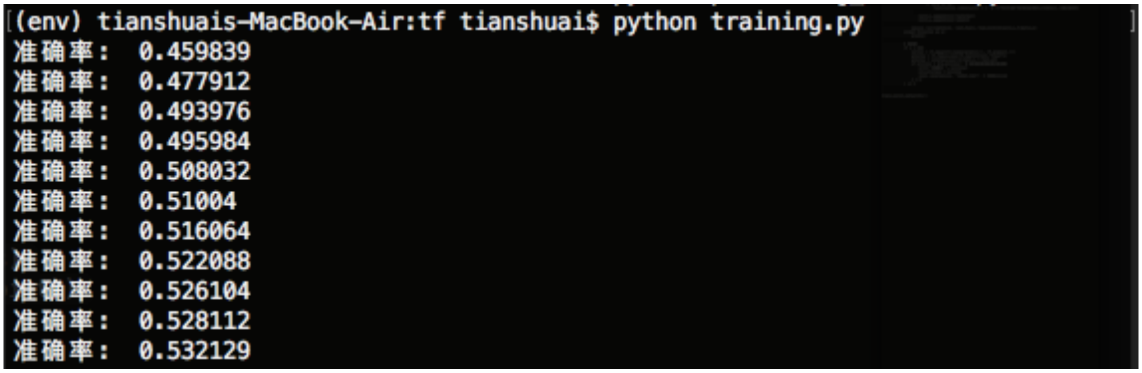
开始漫长的训练

```
import os
import random
import tensorflow as tf
import pickle
import numpy as np
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

f = open('lexcion.pickle', 'rb')
lex = pickle.load(f)
f.close()

```

```

def get_random_line(file, point):
    file.seek(point)
    file.readline()
    return file.readline()
# 从文件中随机选择n条记录
def get_n_random_line(file_name, n=150):
    lines = []
    file = open(file_name, encoding='latin-1')
    total_bytes = os.stat(file_name).st_size
    for i in range(n):
        random_point = random.randint(0, total_bytes)
        lines.append(get_random_line(file, random_point))
    file.close()
    return lines


def get_test_dataset(test_file):
    with open(test_file, encoding='latin-1') as f:
        test_x = []
        test_y = []
        lemmatizer = WordNetLemmatizer()
        for line in f:
            label = line.split(':%:%:%:')[0]
            tweet = line.split(':%:%:%:')[1]
            words = word_tokenize(tweet.lower())
            words = [lemmatizer.lemmatize(word) for word in words]
            features = np.zeros(len(lex))
            for word in words:
                if word in lex:
                    features[lex.index(word)] = 1

            test_x.append(list(features))
            test_y.append(eval(label))
    return test_x, test_y

test_x, test_y = get_test_dataset('tesing.csv')


####################################################################

n_input_layer = len(lex)  # 输入层

n_layer_1 = 2000     # hide layer
n_layer_2 = 2000    # hide layer(隐藏层)听着很神秘，其实就是除输入输出层外

n_output_layer = 3       # 输出层


def neural_network(data):
    # 定义第一层"神经元"的权重和biases
    layer_1_w_b = {'w_':tf.Variable(tf.random_normal([n_input_layer,
    # 定义第二层"神经元"的权重和biases
    layer_2_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_1, n_l
    # 定义输出层"神经元"的权重和biases
    layer_output_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_2

    # w·x+b
    layer_1 = tf.add(tf.matmul(data, layer_1_w_b['w_']), layer_1_w_b
    layer_1 = tf.nn.relu(layer_1)  # 激活函数
    layer_2 = tf.add(tf.matmul(layer_1, layer_2_w_b['w_']), layer_2_
    layer_2 = tf.nn.relu(layer_2 ) # 激活函数
    layer_output = tf.add(tf.matmul(layer_2, layer_output_w_b['w_'])

    return layer_output


X = tf.placeholder('float')
Y = tf.placeholder('float')
batch_size = 90

def train_neural_network(X, Y):
    predict = neural_network(X)
    cost_func = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logi
    optimizer = tf.train.AdamOptimizer().minimize(cost_func)

    with tf.Session() as session:
        session.run(tf.initialize_all_variables())

        lemmatizer = WordNetLemmatizer()
        saver = tf.train.Saver()
        i = 0
        pre_accuracy = 0
        while True:   # 一直训练
            batch_x = []
            batch_y = []

            #if model.ckpt文件已存在:
            #   saver.restore(session, 'model.ckpt')  恢复保存的sessio

            try:
                lines = get_n_random_line('training.csv', batch_size
                for line in lines:
                    label = line.split(':%:%:%:')[0]
                    tweet = line.split(':%:%:%:')[1]
                    words = word_tokenize(tweet.lower())
                    words = [lemmatizer.lemmatize(word) for word in

                    features = np.zeros(len(lex))
                    for word in words:
                        if word in lex:
                            features[lex.index(word)] = 1  # 一个句子

                    batch_x.append(list(features))
                    batch_y.append(eval(label))

                session.run([optimizer, cost_func], feed_dict={X:bat
            except Exception as e:
                print(e)

            # 准确率
            if i > 100:
                correct = tf.equal(tf.argmax(predict,1), tf.argmax(Y
                accuracy = tf.reduce_mean(tf.cast(correct,'float'))
                accuracy = accuracy.eval({X:test_x, Y:test_y})
                if accuracy > pre_accuracy:  # 保存准确率最高的训练模型
                    print('准确率: ', accuracy)
                    pre_accuracy = accuracy
                    saver.save(session, 'model.ckpt')  # 保存session
                i = 0
            i += 1


train_neural_network(X,Y)
```

上面程序占用内存600M，峰值1G。

运行：



```
(env) tianshuais-MacBook-Air:tf tianshuai$ python training.py
准确率:  0.459839
准确率:  0.477912
准确率:  0.493976
准确率:  0.495984
准确率:  0.508032
准确率:  0.51004
准确率:  0.516064
准确率:  0.522088
准确率:  0.526104
准确率:  0.528112
准确率:  0.532129
```

训练模型保存为model.ckpt。

使用训练好的模型

```
import tensorflow as tf
import pickle
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import numpy as np

```

```
f = open('lexcion.pickle', 'rb')
lex = pickle.load(f)
f.close()


n_input_layer = len(lex)  # 输入层

n_layer_1 = 2000     # hide layer
n_layer_2 = 2000    # hide layer(隐藏层)听着很神秘，其实就是除输入输出层外的

n_output_layer = 3       # 输出层
def neural_network(data):
    # 定义第一层"神经元"的权重和biases
    layer_1_w_b = {'w_':tf.Variable(tf.random_normal([n_input_layer,
    # 定义第二层"神经元"的权重和biases
    layer_2_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_1, n_la
    # 定义输出层"神经元"的权重和biases
    layer_output_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_2,

    # w·x+b
    layer_1 = tf.add(tf.matmul(data, layer_1_w_b['w_']), layer_1_w_b[
    layer_1 = tf.nn.relu(layer_1)  # 激活函数
    layer_2 = tf.add(tf.matmul(layer_1, layer_2_w_b['w_']), layer_2_w
    layer_2 = tf.nn.relu(layer_2 ) # 激活函数
    layer_output = tf.add(tf.matmul(layer_2, layer_output_w_b['w_']),

    return layer_output

X = tf.placeholder('float')
def prediction(tweet_text):
    predict = neural_network(X)

    with tf.Session() as session:
        session.run(tf.initialize_all_variables())
        saver = tf.train.Saver()
        saver.restore(session, 'model.ckpt')

        lemmatizer = WordNetLemmatizer()
        words = word_tokenize(tweet_text.lower())
        words = [lemmatizer.lemmatize(word) for word in words]

        features = np.zeros(len(lex))
        for word in words:
            if word in lex:
                features[lex.index(word)] = 1

        #print(predict.eval(feed_dict={X:[features]})) [[val1,val2,va
        res = session.run(tf.argmax(predict.eval(feed_dict={X:[featur
        return res


prediction("I am very happe")
```

Facebook Google+ Twitter Weibo Email

相关文章

Ubuntu 16.04 安装 Tensorflow(GPU支持)

TensorFlow练习1: 对评论进行分类

TensorFlow练习3: RNN, Recurrent Neural Networks

TensorFlow练习4: CNN, Convolutional Neural Networks...

TensorFlow练习5: 训练一个简单的游戏AI（Deep Q Network）...

用微信 OR 支付宝 扫描二维码

为本文作者 打个赏

微信扫一扫转账

用支付宝扫二维码，加我好友

2016年11月16日 wtf ML、coding TensorFlow、教程