

WTF Daily Blog

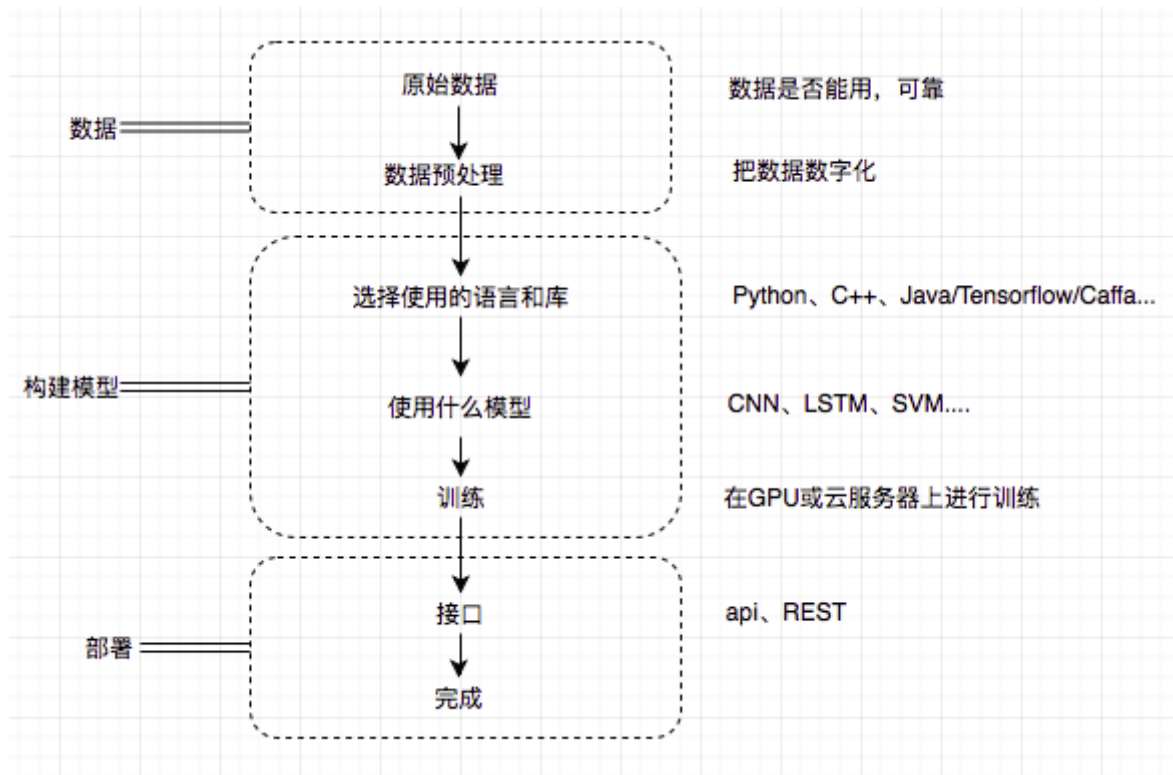
斗大的熊猫

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（update：添加CNN模型）
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
5. 后续有关于中文的练习《[TensorFlow练习13: 制作一个简单的聊天机器人](#)》《[TensorFlow练习7: 基于RNN生成古诗词](#)》《[TensorFlow练习18: 根据姓名判断性别](#)》

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

```
1 import nltk
2 from nltk.tokenize import word_tokenize
3 from nltk.stem import WordNetLemmatizer
4
5 import pickle
6 import numpy as np
7 import pandas as pd
8 from collections import OrderedDict
9
10 org_train_file = 'training.1600000.processed.noemoticon.csv'
11 org_test_file = 'testdata.manual.2009.06.14.csv'
12
13 # 提取文件中有用的字段
14 def usefull_filed(org_file, output_file):
15     output = open(output_file, 'w')
16     with open(org_file, buffering=10000, encoding='latin-1') as f:
17         try:
18             for line in f:
19                 line = line.replace('"', '')
20                 clf = line.split(',')[0] # 4
21                 if clf == '0':
22                     clf = [0, 0, 1] # 消极评论
23                 elif clf == '2':
24                     clf = [0, 1, 0] # 中性评论
25                 elif clf == '4':
26                     clf = [1, 0, 0] # 积极评论
27
28                 tweet = line.split(',')[1]
29                 outputline = str(clf) + ':%:~::~:' + tweet
```

```

30         output.write(outputline) # [0, 0, 1]:%:%:%: that's a bummer. You shoulda got David Carr of
31     except Exception as e:
32         print(e)
33     output.close() # 处理完成, 处理后文件大小127.5M
34
35 usefull_filed(org_train_file, 'training.csv')
36 usefull_filed(org_test_file, 'tesing.csv')
37
38 # 创建词汇表
39 def create_lexicon(train_file):
40     lex = []
41     lemmatizer = WordNetLemmatizer()
42     with open(train_file, buffering=10000, encoding='latin-1') as f:
43         try:
44             count_word = {} # 统计单词出现次数
45             for line in f:
46                 tweet = line.split('%:%:%:')[1]
47                 words = word_tokenize(line.lower())
48                 for word in words:
49                     word = lemmatizer.lemmatize(word)
50                     if word not in count_word:
51                         count_word[word] = 1
52                     else:
53                         count_word[word] += 1
54
55             count_word = OrderedDict(sorted(count_word.items(), key=lambda t: t[1]))
56             for word in count_word:
57                 if count_word[word] < 100000 and count_word[word] > 100: # 过滤掉一些词
58                     lex.append(word)
59         except Exception as e:
60             print(e)
61     return lex
62
63 lex = create_lexicon('training.csv')
64
65 with open('lexcion.pickle', 'wb') as f:
66     pickle.dump(lex, f)
67
68
69 """
70 # 把字符串转为向量
71 def string_to_vector(input_file, output_file, lex):

```

```
72 output_f = open(output_file, 'w')
73 lemmatizer = WordNetLemmatizer()
74 with open(input_file, buffering=10000, encoding='latin-1') as f:
75     for line in f:
76         label = line.split(':%:~:%:')[0]
77         tweet = line.split(':%:~:%:')[1]
78         words = word_tokenize(tweet.lower())
79         words = [lemmatizer.lemmatize(word) for word in words]
80
81         features = np.zeros(len(lex))
82         for word in words:
83             if word in lex:
84                 features[lex.index(word)] = 1 # 一个句子中某个词可能出现两次, 可以用+=1, 其实区别不大
85
86         features = list(features)
87         output_f.write(str(label) + ":" + str(features) + '\n')
88 output_f.close()
89
90
91 f = open('lexcion.pickle', 'rb')
92 lex = pickle.load(f)
93 f.close()
94
95 # lexcion词汇表大小112k, training.vec大约112k*1600000 170G 太大, 只能边转边训练了
96 # string_to_vector('training.csv', 'training.vec', lex)
97 # string_to_vector('tesing.csv', 'tesing.vec', lex)
98 """
```

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

```
1 import os
2 import random
```

```
3 import tensorflow as tf
4 import pickle
5 import numpy as np
6 from nltk.tokenize import word_tokenize
7 from nltk.stem import WordNetLemmatizer
8
9 f = open('lexcion.pickle', 'rb')
10 lex = pickle.load(f)
11 f.close()
12
13
14 def get_random_line(file, point):
15     file.seek(point)
16     file.readline()
17     return file.readline()
18 # 从文件中随机选择n条记录
19 def get_n_random_line(file_name, n=150):
20     lines = []
21     file = open(file_name, encoding='latin-1')
22     total_bytes = os.stat(file_name).st_size
23     for i in range(n):
24         random_point = random.randint(0, total_bytes)
25         lines.append(get_random_line(file, random_point))
26     file.close()
27     return lines
28
29
30 def get_test_dataset(test_file):
31     with open(test_file, encoding='latin-1') as f:
32         test_x = []
33         test_y = []
34         lemmatizer = WordNetLemmatizer()
35         for line in f:
36             label = line.split(':%:~::~')[0]
37             tweet = line.split(':%:~::~')[1]
38             words = word_tokenize(tweet.lower())
39             words = [lemmatizer.lemmatize(word) for word in words]
40             features = np.zeros(len(lex))
41             for word in words:
42                 if word in lex:
43                     features[lex.index(word)] = 1
44
```

```
45         test_x.append(list(features))
46         test_y.append(eval(label))
47     return test_x, test_y
48
49 test_x, test_y = get_test_dataset('tesing.csv')
50
51
52 #####
53
54 n_input_layer = len.lex) # 输入层
55
56 n_layer_1 = 2000 # hide layer
57 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中间层
58
59 n_output_layer = 3 # 输出层
60
61
62 def neural_network(data):
63     # 定义第一层"神经元"的权重和biases
64     layer_1_w_b = {'w_':tf.Variable(tf.random_normal([n_input_layer, n_layer_1])), 'b_':tf.Variable(tf.random_normal([n_layer_1]))}
65     # 定义第二层"神经元"的权重和biases
66     layer_2_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_1, n_layer_2])), 'b_':tf.Variable(tf.random_normal([n_layer_2]))}
67     # 定义输出层"神经元"的权重和biases
68     layer_output_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_2, n_output_layer])), 'b_':tf.Variable(tf.random_normal([n_output_layer]))}
69
70     # w·x+b
71     layer_1 = tf.add(tf.matmul(data, layer_1_w_b['w_']), layer_1_w_b['b_'])
72     layer_1 = tf.nn.relu(layer_1) # 激活函数
73     layer_2 = tf.add(tf.matmul(layer_1, layer_2_w_b['w_']), layer_2_w_b['b_'])
74     layer_2 = tf.nn.relu(layer_2) # 激活函数
75     layer_output = tf.add(tf.matmul(layer_2, layer_output_w_b['w_']), layer_output_w_b['b_'])
76
77     return layer_output
78
79
80 X = tf.placeholder('float')
81 Y = tf.placeholder('float')
82 batch_size = 90
83
84 def train_neural_network(X, Y):
85     predict = neural_network(X)
86     cost_func = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(predict, Y))
```

```
87 optimizer = tf.train.AdamOptimizer().minimize(cost_func)
88
89 with tf.Session() as session:
90     session.run(tf.initialize_all_variables())
91
92     lemmatizer = WordNetLemmatizer()
93     saver = tf.train.Saver()
94     i = 0
95     pre_accuracy = 0
96     while True: # 一直训练
97         batch_x = []
98         batch_y = []
99
100         #if model.ckpt文件已存在:
101         # saver.restore(session, 'model.ckpt') 恢复保存的session
102
103         try:
104             lines = get_n_random_line('training.csv', batch_size)
105             for line in lines:
106                 label = line.split(':%:%%:')[0]
107                 tweet = line.split(':%:%%:')[1]
108                 words = word_tokenize(tweet.lower())
109                 words = [lemmatizer.lemmatize(word) for word in words]
110
111                 features = np.zeros(len(lex))
112                 for word in words:
113                     if word in lex:
114                         features[lex.index(word)] = 1 # 一个句子中某个词可能出现两次,可以用+=1,其实区别不大
115
116                 batch_x.append(list(features))
117                 batch_y.append(eval(label))
118
119             session.run([optimizer, cost_func], feed_dict={X:batch_x,Y:batch_y})
120         except Exception as e:
121             print(e)
122
123     # 准确率
124     if i > 100:
125         correct = tf.equal(tf.argmax(predict,1), tf.argmax(Y,1))
126         accuracy = tf.reduce_mean(tf.cast(correct,'float'))
127         accuracy = accuracy.eval({X:test_x, Y:test_y})
128         if accuracy > pre_accuracy: # 保存准确率最高的训练模型
```



```
129         print('准确率: ', accuracy)
130         pre_accuracy = accuracy
131         saver.save(session, 'model.ckpt') # 保存session
132         i = 0
133         i += 1
134
135
136 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

```
1 import tensorflow as tf
2 import pickle
3 from nltk.tokenize import word_tokenize
4 from nltk.stem import WordNetLemmatizer
5 import numpy as np
6
7 f = open('lexcion.pickle', 'rb')
8 lex = pickle.load(f)
```

```

9  f.close()
10
11 n_input_layer = len(lex) # 输入层
12
13 n_layer_1 = 2000 # hide layer
14 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中间层
15
16 n_output_layer = 3 # 输出层
17 def neural_network(data):
18     # 定义第一层"神经元"的权重和biases
19     layer_1_w_b = {'w_':tf.Variable(tf.random_normal([n_input_layer, n_layer_1])), 'b_':tf.Variable(tf.random_normal([n_layer_1]))}
20     # 定义第二层"神经元"的权重和biases
21     layer_2_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_1, n_layer_2])), 'b_':tf.Variable(tf.random_normal([n_layer_2]))}
22     # 定义输出层"神经元"的权重和biases
23     layer_output_w_b = {'w_':tf.Variable(tf.random_normal([n_layer_2, n_output_layer])), 'b_':tf.Variable(tf.random_normal([n_output_layer]))}
24
25     # w·x+b
26     layer_1 = tf.add(tf.matmul(data, layer_1_w_b['w_']), layer_1_w_b['b_'])
27     layer_1 = tf.nn.relu(layer_1) # 激活函数
28     layer_2 = tf.add(tf.matmul(layer_1, layer_2_w_b['w_']), layer_2_w_b['b_'])
29     layer_2 = tf.nn.relu(layer_2) # 激活函数
30     layer_output = tf.add(tf.matmul(layer_2, layer_output_w_b['w_']), layer_output_w_b['b_'])
31
32     return layer_output
33
34 X = tf.placeholder('float')
35 def prediction(tweet_text):
36     predict = neural_network(X)
37
38     with tf.Session() as session:
39         session.run(tf.initialize_all_variables())
40         saver = tf.train.Saver()
41         saver.restore(session, 'model.ckpt')
42
43         lemmatizer = WordNetLemmatizer()
44         words = word_tokenize(tweet_text.lower())
45         words = [lemmatizer.lemmatize(word) for word in words]
46
47         features = np.zeros(len(lex))
48         for word in words:
49             if word in lex:
50                 features[lex.index(word)] = 1

```

```
51
52     #print(predict.eval(feed_dict={X:[features]})) [[val1,val2,val3]]
53     res = session.run(tf.argmax(predict.eval(feed_dict={X:[features]}),1 ))
54     return res
55
56
57 prediction("I am very happy")
```

上面使用简单的feedforward模型，下面使用CNN模型

```
1  # https://github.com/Lab41/sunny-side-up
2  import os
3  import random
4  import tensorflow as tf
5  import pickle
6  import numpy as np
7  from nltk.tokenize import word_tokenize
8  from nltk.stem import WordNetLemmatizer
9
10 f = open('lexcion.pickle', 'rb')
11 lex = pickle.load(f)
12 f.close()
13
14 def get_random_line(file, point):
15     file.seek(point)
16     file.readline()
17     return file.readline()
18 # 从文件中随机选择n条记录
19 def get_n_random_line(file_name, n=150):
20     lines = []
21     file = open(file_name, encoding='latin-1')
22     total_bytes = os.stat(file_name).st_size
23     for i in range(n):
24         random_point = random.randint(0, total_bytes)
25         lines.append(get_random_line(file, random_point))
26     file.close()
27     return lines
28
29 def get_test_dataset(test_file):
```

```

30 with open(test_file, encoding='latin-1') as f:
31     test_x = []
32     test_y = []
33     lemmatizer = WordNetLemmatizer()
34     for line in f:
35         label = line.split(':%:%%:')[0]
36         tweet = line.split(':%:%%:')[1]
37         words = word_tokenize(tweet.lower())
38         words = [lemmatizer.lemmatize(word) for word in words]
39         features = np.zeros(len(lex))
40         for word in words:
41             if word in lex:
42                 features[lex.index(word)] = 1
43
44         test_x.append(list(features))
45         test_y.append(eval(label))
46     return test_x, test_y
47
48 test_x, test_y = get_test_dataset('tesing.csv')
49 #####
50 input_size = len(lex)
51 num_classes = 3
52
53 X = tf.placeholder(tf.int32, [None, input_size])
54 Y = tf.placeholder(tf.float32, [None, num_classes])
55
56 dropout_keep_prob = tf.placeholder(tf.float32)
57
58 batch_size = 90
59
60 def neural_network():
61     # embedding layer
62     with tf.device('/cpu:0'), tf.name_scope("embedding"):
63         embedding_size = 128
64         W = tf.Variable(tf.random_uniform([input_size, embedding_size], -1.0, 1.0))
65         embedded_chars = tf.nn.embedding_lookup(W, X)
66         embedded_chars_expanded = tf.expand_dims(embedded_chars, -1)
67     # convolution + maxpool layer
68     num_filters = 128
69     filter_sizes = [3,4,5]
70     pooled_outputs = []
71     for i, filter_size in enumerate(filter_sizes):

```

```

72     with tf.name_scope("conv-maxpool-%s" % filter_size):
73         filter_shape = [filter_size, embedding_size, 1, num_filters]
74         W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1))
75         b = tf.Variable(tf.constant(0.1, shape=[num_filters]))
76         conv = tf.nn.conv2d(embedded_chars_expanded, W, strides=[1, 1, 1, 1], padding="VALID")
77         h = tf.nn.relu(tf.nn.bias_add(conv, b))
78         pooled = tf.nn.max_pool(h, ksize=[1, input_size - filter_size + 1, 1, 1], strides=[1, 1, 1, 1],
79                                 pooled_outputs.append(pooled)
80
81     num_filters_total = num_filters * len(filter_sizes)
82     h_pool = tf.concat(3, pooled_outputs)
83     h_pool_flat = tf.reshape(h_pool, [-1, num_filters_total])
84     # dropout
85     with tf.name_scope("dropout"):
86         h_drop = tf.nn.dropout(h_pool_flat, dropout_keep_prob)
87     # output
88     with tf.name_scope("output"):
89         W = tf.get_variable("W", shape=[num_filters_total, num_classes], initializer=tf.contrib.layers.xavier_initializer())
90         b = tf.Variable(tf.constant(0.1, shape=[num_classes]))
91         output = tf.nn.xw_plus_b(h_drop, W, b)
92
93     return output
94
95 def train_neural_network():
96     output = neural_network()
97
98     optimizer = tf.train.AdamOptimizer(1e-3)
99     loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(output, Y))
100    grads_and_vars = optimizer.compute_gradients(loss)
101    train_op = optimizer.apply_gradients(grads_and_vars)
102
103    saver = tf.train.Saver(tf.global_variables())
104    with tf.Session() as sess:
105        sess.run(tf.global_variables_initializer())
106
107        lemmatizer = WordNetLemmatizer()
108        i = 0
109        while True:
110            batch_x = []
111            batch_y = []
112
113            #if model.ckpt文件已存在:

```

```
114 # saver.restore(session, 'model.ckpt') 恢复保存的session
115 try:
116     lines = get_n_random_line('training.csv', batch_size)
117     for line in lines:
118         label = line.split(':%:~::~')[0]
119         tweet = line.split(':%:~::~')[1]
120         words = word_tokenize(tweet.lower())
121         words = [lemmatizer.lemmatize(word) for word in words]
122
123         features = np.zeros(len(lex))
124         for word in words:
125             if word in lex:
126                 features[lex.index(word)] = 1 # 一个句子中某个词可能出现两次,可以用+=1,其实区别不大
127
128         batch_x.append(list(features))
129         batch_y.append(eval(label))
130
131     _, loss_ = sess.run([train_op, loss], feed_dict={X:batch_x, Y:batch_y, dropout_keep_prob:0.5})
132     print(loss_)
133 except Exception as e:
134     print(e)
135
136 if i % 10 == 0:
137     predictions = tf.argmax(output, 1)
138     correct_predictions = tf.equal(predictions, tf.argmax(Y, 1))
139     accuracy = tf.reduce_mean(tf.cast(correct_predictions, "float"))
140     accur = sess.run(accuracy, feed_dict={X:test_x[0:50], Y:test_y[0:50], dropout_keep_prob:1.0})
141     print('准确率:', accur)
142
143     i += 1
144
145 train_neural_network()
```

使用了CNN模型之后, 准确率有了显著提升。

[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5: 训练一个简单的游戏AI (Deep Q Network) ...

TensorFlow练习8: 生成音乐

TensorFlow练习15: 中文语音识别

📅 2016年11月16日 👤 wtf 📁 ML、coding 🔖 TensorFlow、教程

《TensorFlow练习2: 对评论进行分类》有5个想法

 liang

2017年3月2日 下午5:20

你好，我在运行fnn时有一些错误，你写的这行 `label = line.split(':', 1)[0]` 返回的是字符串吧，是不是不能直接当做标签呢

 骆炜

2017年2月28日 下午2:44

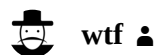
你好，我在训练的时候遇到两个错误，不知道是不是TF版本的问题（1.0）。能帮我看一下么

```
cost_func = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(predict, Y))
```

ValueError: Only call softmax_cross_entropy_with_logits with named arguments (labels=..., logits=..., ...)

```
optimizer = tf.train.AdamOptimizer().minimize(cost_func)
```

ValueError: No gradients provided for any variable, check your graph for ops that do not support gradients, between variables



2017年2月28日 下午4:04

API变了 看一下文档



2016年12月28日 下午4:35

我按照你给的地址下载了数据，然后发现数据里面有很多乱码，你是怎么解决的啊？



2017年2月4日 下午4:44

with open(org_file, buffering=10000, errors='ignore') as f:

