第一部分 背景与问题描述

Environment

- jupyter notebook (测试、试验环境)
- Pycharm （开发环境）
- python3.6
- networkx
- jieba
- numpy, pandas, matplotlib
- gensim
- (optional) bottle

Background

在自然语言处理中，有一个常见的问题就是对客户的评价进行分析。 这些用户评论中，包含了大量的有用信息，例如情感分析，或者相关事实描述。 例如，

> “味道不错的面馆，性价比也相当之高，分量很足～女生吃小份，胃口小的，可能吃不完呢。环境在面馆来说算是好的，至少看上去堂子很亮，也比较干净，一般苍蝇馆子还是比不上这个卫生状况的。中午饭点的时候，人很多，人行道上也是要坐满的，隔壁的冒菜馆子，据说是一家，有时候也会开放出来坐吃面的人。“

首先情感是正向的，除此之外我们还能够进行知道这个的几个事实描述：1. 性价比比较高； 2. 装修比较好； 3. 分量足。

这些信息是非常重要宝贵的，不论是对于公司进行商业分析或者要建立一个搜索引擎排序，这些信息都是重要的参考因素。 那么在这个时候，我们就需要进行文本的情感分类了。

注: 此次问题的来源数据集来源于大众点评， 这个数据集也被用作 AI 全球挑战赛的数据集。 而且细粒度情感分类这个问题其实在目前而言是一个 *未解之谜*， 人类现在对这个问题并没有什么特别好的方法， 因为人的情感表达真的是很有变化的，例如“我不认为这个地方不好是不对的”。所以这个问题也需要大家在之后做出来基础模型之后，大家多想想办法，*八仙过海，各显神通*

所以，我们这一次是要使用深度学习的方法，建立一个模型，这个模型能够将一句话进行分类判断，判断出来这句话到底表达了什么重要信息。 说实话，这个看似简单的问题，曾经是困扰了科学家数十年的问题，就算是现在，深度学习，人工智能有了很大的进步，其效果也达不到人们预期的那么好,但是比起前些年已经好多了:)

这个问题我们希望的是，输入一句话，输出是这句话对于以下6大类，20小类进行打标，对于每个小类而言，都会有**< 正面情感, 中性情感, 负面情感, 情感倾向未提及 >** 这4个类别。

总得来说，我们现在这6大类，20小类的类别如下：

- 位置: location
  - 交通是否便利(traffic convenience)
  - 距离商圈远近(distance from business district)
  - 是否容易寻找(easy to find)
- 服务(service)
  - 排队等候时间(wait time)
  - 服务人员态度(waiter’s attitude)
  - 是否容易停车(parking convenience)
  - 点菜/上菜速度(serving speed)
- 价格(price)
  - 价格水平(price level)
  - 性价比(cost-effective)
  - 折扣力度(discount)
- 环境(environment)
  - 装修情况(decoration)
  - 嘈杂情况(noise)
  - 就餐空间(space)
  - 卫生情况(cleaness)
- 菜品(dish)
  - 分量(portion)
  - 口感(taste)
  - 外观(look)
  - 推荐程度(recommendation)
- 其他(others)
  - 本次消费感受(overall experience)
  - 再次消费的意愿(willing to consume again)

而为了方便训练数据的标标注，训练数据中，**< 正面情感, 中性情感, 负面情感, 情感倾向未提及 >** 分别对应与 (1, 0, -1, -2).

例如说，

> “味道不错的面馆，性价比也相当之高，分量很足～女生吃小份，胃口小的，可能吃不完呢。环境在面馆来说算是好的，至少看上去堂子很亮，也比较干净，一般苍蝇馆子还是比不上这个卫生状况的。中午饭点的时候，人很多，人行道上也是要坐满的，隔壁的冒菜馆子，据说是一家，有时候也会开放出来坐吃面的人。“

这句话在训练数据中的标签就是：

- 交通是否便利(traffic convenience) -2
- 距离商圈远近(distance from business district) -2
- 是否容易寻找(easy to find) -2
- 排队等候时间(wait time) -2
- 服务人员态度(waiter’s attitude) -2
- 是否容易停车(parking convenience) -2
- 点菜/上菜速度(serving speed) -2
- 价格水平(price level) -2
- 性价比(cost-effective) 1
- 折扣力度(discount) -2
- 装修情况(decoration) 1
- 嘈杂情况(noise) -2
- 就餐空间(space) -2
- 卫生情况(cleaness) 1
- 分量(portion) 1
- 口感(taste) 1
- 外观(look) -2
- 推荐程度(recommendation) -2
- 次消费感受(overall experience) 1
- 再次消费的意愿(willing to consume again) -2

数据集下载

数据集的下载在 https://challenger.ai/competition/fsauor2018， (https://challenger.ai/competition/fsauor2018，) 大家下载数据集， 以及测试集，注意，训练集我们需要分成 training data, validation data 然后 test data里边的数据绝对不能在训练的时候用。 否则的话就是去了意义。

注, 这个数据集之所以被用到了 AI 挑战赛中，是因为其难度很大。 绝大数人在公司中遇到的问题难度不会超过这个问题。

In [1]:
```
import os
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee649
1hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fww
w.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.
com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

This Pre-trained WordVec From: (https://www.kaggle.com/guiyihan/fasttext-chinese-word-embedding#cc.zh.300.vec.gz)

Reference from kaggle competition (https://www.kaggle.com/yekenot/fasttext-crawl-300d-2m/downloads/crawl-300d-2M.vec/1)

In [4]:
```
import numpy as np
np.random.seed(42)
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

from keras.models import Model
from keras.layers import Input, Embedding, Dense, Conv2D, MaxPool2D
from keras.layers import Reshape, Flatten, Concatenate, Dropout, SpatialDropout1D
from keras.preprocessing import text, sequence
from keras.callbacks import Callback

import warnings
warnings.filterwarnings('ignore')

import os
os.environ['OMP_NUM_THREADS'] = '4'
```

```
Using TensorFlow backend.
```

In [0]:
```
EMBEDDING_FILE = '/content/drive/My Drive/Colab Notebooks/2comment_classification/cc.zh.300.vec' # consider change this into Chinese Embedding File
```

Clean Chinese Content

Reference Blog (https://www.dlology.com/blog/tutorial-chinese-sentiment-analysis-with-hotel-review-data/)

In [0]:
```
#繁体转简体
from langconv import *
import jieba

def is_CN_char(ch):
    return ch >= u'\u4e00' and ch <= u'\u9fa5'

def cut(string):
    return list(jieba.cut(string))

def get_stopwords(filename = "chinese_stopwords.txt"):
    stopwords_dic = open(filename, encoding= 'utf-8')
    stopwords = stopwords_dic.readlines()
    stopwords = [w.split() for w in stopwords]
    stopwords_dic.close()
    return stopwords

def convert2simple(word):
    return Converter('zh-hans').convert(word)
```

In [0]:
```
def clean_sentence(sentence):
    stopwords = get_stopwords()
    sentence = ''.join(filter(is_CN_char,sentence))
    sentence = convert2simple(sentence)
    words = [w for w in cut(sentence) if len(w)>1 and w not in stopwords]
    words = ' '.join(words)
    return words
```

In [0]:
```
# train["content"] = train["content"].apply(clean_sentence)
# test["content"] = test["content"].apply(clean_sentence)
# submission["content"] = submission["content"].apply(clean_sentence)
```

In [0]:
```
# train.to_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/train_cut.csv", index = False)
# test.to_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/test_cut.csv", index = False)
# submission.to_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/submission_cut.csv", index = Fals
e)
```

In [0]:
```
train = pd.read_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/train_cut.csv")
test = pd.read_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/test_cut.csv")
submission = pd.read_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/submission_cut.csv")
```

In [46]:
```
submission.head(2)
```

Out[46]:

	id	content	location_traffic_convenience	location_distance_from_business_district	location_easy_to_find	service_wait_time	serv
0	0	想当年 佘山 时候 没有 三品 香算 镇上 最大 看起来 像样 饭店 菜品 有点 感觉 有杂...	-2	-2	-2	0	-2
1	1	趁着 国庆节 一家人 白天 山里 玩耍 之后 晚上 决定 李记 搅团 东门外 这家 店门口 ...	-2	-2	-1	-2	-2

数据预处理及评论分析

In [7]:
```
train.columns
```

Out[7]:
```
Index(['id', 'content', 'location_traffic_convenience',
       'location_distance_from_business_district', 'location_easy_to_find',
       'service_wait_time', 'service_waiters_attitude',
       'service_parking_convenience', 'service_serving_speed', 'price_level',
       'price_cost_effective', 'price_discount', 'environment_decoration',
       'environment_noise', 'environment_space', 'environment_cleaness',
       'dish_portion', 'dish_taste', 'dish_look', 'dish_recommendation',
       'others_overall_experience', 'others_willing_to_consume_again'],
      dtype='object')
```

In [0]:
```
index_row = list(train.columns)
index_row.remove("id")
index_row.remove("content")
```

In [0]:
```
from collections import defaultdict
row_type_count = defaultdict(int)
```

In [0]:
```
for i in index_row:
    Distribution_types = {i: 0 for i in [1, -1, 0, -2]} #计数初始化
    for j in [1, -1, 0, -2]:
        Distribution_types[j] = list(train[i]).count(j)
    row_type_count[i] = Distribution_types
```

In [0]:
```
row_type_count = pd.DataFrame(row_type_count)
```

In [12]:
```
row_type_count
```

Out[12]:

	location_traffic_convenience	location_distance_from_business_district	location_easy_to_find	service_wait_time	service_waiters
-2	81382	83680	80605	92763	42410
-1	1318	586	3976	3034	8684
0	1046	533	2472	4382	12534
1	21254	20201	17947	4821	41372

In [0]:
```
row_type_percent = row_type_count/len(train)
```

In [14]:
```
row_type_percent
```

Out[14]:

	location_traffic_convenience	location_distance_from_business_district	location_easy_to_find	service_wait_time	service_waiters
-2	0.775067	0.796952	0.767667	0.883457	0.403905
-1	0.012552	0.005581	0.037867	0.028895	0.082705
0	0.009962	0.005076	0.023543	0.041733	0.119371
1	0.202419	0.192390	0.170924	0.045914	0.394019

评论分析

- **差评率分析：** 可见类别为-1（差评）的评论数占比最少，但是，该类评论是最重要的以及我们最关心的类别。之后我们对店家提出改进措施也是主要针对该类评论
- **好评率分析：** 类别为1（好评）的评论数占比总体不少，在显示生活中也是如此，大部分消费者给出的评论往往是好评。分析店家的好评率也很重要，这对于横向比较以及纵向挖掘店家具体优势点很有帮助。
- **提及率分析：** 其次才考虑-2（未提及），提及率体现了消费者对某个特征的关心程度，店家应该针对高提及率的特征进行布局。
- **合并变量：** 最后才考虑0（提及但为表达明确感情），这个变量扰动最大，有些消费者提到某个特点，但是1.未表达感情或者2.即说喜欢有说不喜欢的词；比如在dish_taste和price_level该特征中0的比例比较高，因为“吃起来”、“味道”|“价格”等高频词都将评论归为dish_taste，但是消费者的评论出现1和2的情况较常见。考虑到该类别（0）的重要性较小，且占比总体不高，**我们将其合并到-1该类中，统称为not_positive**。
- **合并影响：** **合并的弊端：**这样分析的结果往往比真实结果的差评率高一些，尤其是在dish_taste和price_level上。**合并的好处：**1. 分析次数减半，提高了极大的便利；2.一定程度弥补了“评论的客户多为好评客户”以及“模糊评论往往不大满意”的数据缺陷

变量转换

In [0]:
```
# previous columns
pre_col = list(train.columns)
for x in ["id", "content"]:
    pre_col.remove(x)
```

In [0]:
```
suffix_type = ["mentioned", "positive"]

# new columns
new_col = []
for i in pre_col:
    for j in suffix_type:
        new_col.append(i + "_" + j)
```

In [17]:
```
len(new_col)
```

Out[17]: 40

In [0]:
```
for i in pre_col:
    for j in suffix_type:
        new_col_i_j = i + "_" + j
        if j == "mentioned":
            train[new_col_i_j] = np.where(train[i] == -2, 0, 1)
        elif j == "positive":
            train[new_col_i_j] = np.where(train[i] == 1, 1, 0)
        else:
            print("error")

#train.to_csv("/content/drive/My Drive/Colab Notebooks/2comment_classification/train_cut_add_new_col.csv", index = Fal
se)
```

In [19]:
```
train.head(2)
```

Out[19]:

	id	content	location_traffic_convenience	location_distance_from_business_district	location_easy_to_find	service_wait_time	serv
0	0	吼吼 吼萌 棒棒糖 大众点评 霸王餐 可爱 一直 好奇 这个 棒棒糖 怎么 东西 大众 点...	-2	-2	-2	-2	1
1	1	第三次 参加 大众 点评 霸王餐 活动 这家 整体 感觉 一般 首先 环境 只能 中等 其次...	-2	-2	-2	-2	-2

In [0]:
```
for i in pre_col:
    for j in suffix_type:
        new_col_i_j = i + "_" + j
        if j == "mentioned":
            test[new_col_i_j] = np.where(test[i] == -2, 0, 1)
        elif j == "positive":
            test[new_col_i_j] = np.where(test[i] == 1, 1, 0)
        else:
            print("error")

#test.to_csv("test_cut_add_new_col.csv", index = False)
```

In [0]:
```
for i in pre_col:
    for j in suffix_type:
        new_col_i_j = i + "_" + j
        if j == "mentioned":
            submission[new_col_i_j] = np.where(submission[i] == -2, 0, 1)
        elif j == "positive":
            submission[new_col_i_j] = np.where(submission[i] == 1, 1, 0)
        else:
            print("error")

#submission.to_csv("submission_cut_add_new_col.csv", index = False)
```

In [48]:
```
submission.head(2)
```

Out[48]:

	id	content	location_traffic_convenience	location_distance_from_business_district	location_easy_to_find	service_wait_time	serv
0	0	想当年 佘山 时候 没有 三品 香算 镇上 最大 看起来 像样 饭店 菜品 有点 感觉 有杂...	-2	-2	-2	0	-2
1	1	趁着 国庆节 一家人 白天 山里 玩耍 之后 晚上 决定 李记 搅团 东门外 这家 店门口 ...	-2	-2	-1	-2	-2

变量分组训练与测试-7组*6

In [22]:
```
new_col[:2]
```

Out[22]:
```
['location_traffic_convenience_mentioned',
 'location_traffic_convenience_positive']
```

In [0]:
```
group = []
for i in [0, 6, 14, 20, 26, 32, 36]:
    if i < 36:
        group.append(new_col[i:i+6])
    else:
        group.append(new_col[i:] + new_col[12:14]) # 人为安排，尽可能让同一大类在一组
```

In [24]:
```
group
```

```
Out[24]: [['location_traffic_convenience_mentioned',
           'location_traffic_convenience_positive',
           'location_distance_from_business_district_mentioned',
           'location_distance_from_business_district_positive',
           'location_easy_to_find_mentioned',
           'location_easy_to_find_positive'],
          ['service_wait_time_mentioned',
           'service_wait_time_positive',
           'service_waiters_attitude_mentioned',
           'service_waiters_attitude_positive',
           'service_parking_convenience_mentioned',
           'service_parking_convenience_positive'],
          ['price_level_mentioned',
           'price_level_positive',
           'price_cost_effective_mentioned',
           'price_cost_effective_positive',
           'price_discount_mentioned',
           'price_discount_positive'],
          ['environment_decoration_mentioned',
           'environment_decoration_positive',
           'environment_noise_mentioned',
           'environment_noise_positive',
           'environment_space_mentioned',
           'environment_space_positive'],
          ['environment_cleaness_mentioned',
           'environment_cleaness_positive',
           'dish_portion_mentioned',
           'dish_portion_positive',
           'dish_taste_mentioned',
           'dish_taste_positive'],
          ['dish_look_mentioned',
           'dish_look_positive',
           'dish_recommendation_mentioned',
           'dish_recommendation_positive',
           'others_overall_experience_mentioned',
           'others_overall_experience_positive'],
          ['others_overall_experience_mentioned',
           'others_overall_experience_positive',
           'others_willing_to_consume_again_mentioned',
           'others_willing_to_consume_again_positive',
```

```
'service_serving_speed_mentioned',
'service_serving_speed_positive']]
```

循环7次训练，打印结果并预测submission数据

In [0]:
```
class RocAucEvaluation(Callback):
    def __init__(self, validation_data=(), interval=1):
        super(Callback, self).__init__()

        self.interval = interval
        self.X_val, self.y_val = validation_data

    def on_epoch_end(self, epoch, logs={}):
        if epoch % self.interval == 0:
            y_pred = self.model.predict(self.X_val, verbose=0)
            score = roc_auc_score(self.y_val, y_pred)
            print("\n ROC-AUC - epoch: %d - score: %.6f \n" % (epoch+1, score))
```

In [0]:
```
def get_model():
    inp = Input(shape=(maxlen, ))
    x = Embedding(max_features, embed_size, weights=[embedding_matrix])(inp)
    x = SpatialDropout1D(0.4)(x)
    x = Reshape((maxlen, embed_size, 1))(x)

    conv_0 = Conv2D(num_filters, kernel_size=(filter_sizes[0], embed_size), kernel_initializer='normal',
                                                                            activation='elu')(x)
    conv_1 = Conv2D(num_filters, kernel_size=(filter_sizes[1], embed_size), kernel_initializer='normal',
                                                                            activation='elu')(x)
    conv_2 = Conv2D(num_filters, kernel_size=(filter_sizes[2], embed_size), kernel_initializer='normal',
                                                                            activation='elu')(x)
    conv_3 = Conv2D(num_filters, kernel_size=(filter_sizes[3], embed_size), kernel_initializer='normal',
                                                                            activation='elu')(x)

    maxpool_0 = MaxPool2D(pool_size=(maxlen - filter_sizes[0] + 1, 1))(conv_0)
    maxpool_1 = MaxPool2D(pool_size=(maxlen - filter_sizes[1] + 1, 1))(conv_1)
    maxpool_2 = MaxPool2D(pool_size=(maxlen - filter_sizes[2] + 1, 1))(conv_2)
    maxpool_3 = MaxPool2D(pool_size=(maxlen - filter_sizes[3] + 1, 1))(conv_3)

    z = Concatenate(axis=1)([maxpool_0, maxpool_1, maxpool_2, maxpool_3])
    z = Flatten()(z)
    z = Dropout(0.1)(z)

    outp = Dense(6, activation="sigmoid")(z) # 输出层维度为6，与预测变量数保持一致

    model = Model(inputs=inp, outputs=outp)
    model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

    return model
```

In [ ]:
```
from keras.models import model_from_json
def save_model(model, group_i):
  # serialize model to JSON
  model_json = model.to_json()
  with open("/content/drive/My Drive/Colab Notebooks/2comment_classification/saved_model/model" + str(group_i) + ".jso
n", "w") as json_file:
    json_file.write(model_json)

  # serialize weights to HDF5
  model.save_weights("/content/drive/My Drive/Colab Notebooks/2comment_classification/saved_model/model" + str(group_i
) + ".h5")
  print("Saved model to disk")
```

In [30]:
```
def get_coefs(word, *arr): return word, np.asarray(arr, dtype='float32')
embeddings_index = dict(get_coefs(*o.rstrip().rsplit(' ')) for o in open(EMBEDDING_FILE))

test2 = test
submission2 = submission

for group_i in range(7):
    X_train = train["content"].fillna("fillna").values
    X_test = test["content"].fillna("fillna").values
    X_submission = submission["content"].fillna("fillna").values

    y_train = train[group[group_i]] # 此处用到i，作为7组变量依次放入模型
    y_test = test[group[group_i]]
    y_submission = submission[group[group_i]]

    y_train_values = y_train.values
    y_test_values = y_test.values
    y_submission_values = y_submission.values

    # 对X中的分词进行text_to_sequences和pad_sequences处理，转换成数值
    max_features = 100000 # 汉语词汇量是英语词汇量5倍左右，考虑用500,000
    maxlen = 200
    embed_size = 300

    filter_sizes = [1,2,3,5] # 和输入和输出的维度有关
    num_filters = 32

    batch_size = 256
    epochs = 3 # 迭代3批次，每批次输入256堆数据

    tokenizer = text.Tokenizer(num_words=max_features)
    tokenizer.fit_on_texts(list(X_train) + list(X_test) + list(X_submission))
    X_train = tokenizer.texts_to_sequences(X_train)
    X_test = tokenizer.texts_to_sequences(X_test)
    X_submission = tokenizer.texts_to_sequences(X_submission)

    x_train = sequence.pad_sequences(X_train, maxlen=maxlen)
    x_test = sequence.pad_sequences(X_test, maxlen=maxlen)
    x_submission = sequence.pad_sequences(X_submission, maxlen=maxlen)

    # 结合预训练词向量embeddings_index优化embedding_vector
```

```
    word_index = tokenizer.word_index
    nb_words = min(max_features, len(word_index))
    embedding_matrix = np.zeros((nb_words, embed_size))
    for word, i in word_index.items():
        if i >= max_features: continue
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None: embedding_matrix[i] = embedding_vector

    model = get_model() # 自定义函数作为对象赋值给model

    X_tra, X_val, y_tra, y_val = train_test_split(x_train, y_train_values, train_size=0.95, random_state=233)
    RocAuc = RocAucEvaluation(validation_data=(X_val, y_val), interval=1)

    hist = model.fit(X_tra, y_tra, batch_size=batch_size, epochs=epochs, validation_data=(X_val, y_val),
                     callbacks=[RocAuc], verbose=2)
    print("############ finish第 " + str(group_i+1) + " 组输入 ############")

    # 存储模型
    save_model(model, group_i)

    y_test_pred = model.predict(x_test, batch_size=1024)
    test2[group[group_i]] = y_test_pred #存储进新的DF，之后与原DF做比较，计算accuracy
    if group_i == 6:
        test2.to_csv('/content/drive/My Drive/Colab Notebooks/2comment_classification/test_cut_add_prediction'+str(gro
up_i+1)+'.csv', index=False)

    y_submission_pred = model.predict(x_submission, batch_size=1024)
    submission2[group[group_i]] = y_submission_pred #存储进新的DF，之后与原DF做比较，计算accuracy
    if group_i == 6:
        submission2.to_csv('/content/drive/My Drive/Colab Notebooks/2comment_classification/submission_cut_add_predict
ion'+str(group_i+1)+'.csv', index=False)
```

```
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 23s - loss: 0.3166 - acc: 0.8770 - val_loss: 0.2384 - val_acc: 0.9119

 ROC-AUC - epoch: 1 - score: 0.928242

Epoch 2/3
 - 22s - loss: 0.2324 - acc: 0.9121 - val_loss: 0.2246 - val_acc: 0.9162

 ROC-AUC - epoch: 2 - score: 0.940003

Epoch 3/3
 - 23s - loss: 0.2094 - acc: 0.9187 - val_loss: 0.2237 - val_acc: 0.9165

 ROC-AUC - epoch: 3 - score: 0.940799

############ finish第 1 组输入 ############
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 24s - loss: 0.2413 - acc: 0.9043 - val_loss: 0.1713 - val_acc: 0.9383

 ROC-AUC - epoch: 1 - score: 0.928728

Epoch 2/3
 - 23s - loss: 0.1665 - acc: 0.9388 - val_loss: 0.1591 - val_acc: 0.9427

 ROC-AUC - epoch: 2 - score: 0.937617

Epoch 3/3
 - 23s - loss: 0.1488 - acc: 0.9455 - val_loss: 0.1549 - val_acc: 0.9438

 ROC-AUC - epoch: 3 - score: 0.939374

############ finish第 2 组输入 ############
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 24s - loss: 0.3749 - acc: 0.8368 - val_loss: 0.2894 - val_acc: 0.8814

 ROC-AUC - epoch: 1 - score: 0.918742

Epoch 2/3
```

```
 - 24s - loss: 0.2883 - acc: 0.8790 - val_loss: 0.2764 - val_acc: 0.8872

 ROC-AUC - epoch: 2 - score: 0.926919

Epoch 3/3
 - 23s - loss: 0.2660 - acc: 0.8892 - val_loss: 0.2748 - val_acc: 0.8879

 ROC-AUC - epoch: 3 - score: 0.928833

############ finish第 3 组输入 ############
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 24s - loss: 0.4203 - acc: 0.8049 - val_loss: 0.3389 - val_acc: 0.8559

 ROC-AUC - epoch: 1 - score: 0.913760

Epoch 2/3
 - 24s - loss: 0.3282 - acc: 0.8580 - val_loss: 0.3231 - val_acc: 0.8636

 ROC-AUC - epoch: 2 - score: 0.922992

Epoch 3/3
 - 23s - loss: 0.2976 - acc: 0.8731 - val_loss: 0.3207 - val_acc: 0.8640

 ROC-AUC - epoch: 3 - score: 0.924588

############ finish第 4 组输入 ############
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 24s - loss: 0.4532 - acc: 0.7764 - val_loss: 0.3731 - val_acc: 0.8316

 ROC-AUC - epoch: 1 - score: 0.874294

Epoch 2/3
 - 24s - loss: 0.3661 - acc: 0.8351 - val_loss: 0.3503 - val_acc: 0.8434

 ROC-AUC - epoch: 2 - score: 0.890272

Epoch 3/3
 - 23s - loss: 0.3333 - acc: 0.8537 - val_loss: 0.3457 - val_acc: 0.8457

 ROC-AUC - epoch: 3 - score: 0.892329
```

```
############ finish第 5 组输入 ############
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 24s - loss: 0.3768 - acc: 0.8399 - val_loss: 0.3201 - val_acc: 0.8719

 ROC-AUC - epoch: 1 - score: 0.807823

Epoch 2/3
 - 24s - loss: 0.3094 - acc: 0.8733 - val_loss: 0.3068 - val_acc: 0.8787

 ROC-AUC - epoch: 2 - score: 0.814380

Epoch 3/3
 - 23s - loss: 0.2843 - acc: 0.8854 - val_loss: 0.3098 - val_acc: 0.8769

 ROC-AUC - epoch: 3 - score: 0.816205

############ finish第 6 组输入 ############
Train on 99750 samples, validate on 5250 samples
Epoch 1/3
 - 24s - loss: 0.3644 - acc: 0.8385 - val_loss: 0.2936 - val_acc: 0.8776

 ROC-AUC - epoch: 1 - score: 0.827505

Epoch 2/3
 - 24s - loss: 0.2834 - acc: 0.8833 - val_loss: 0.2786 - val_acc: 0.8888

 ROC-AUC - epoch: 2 - score: 0.841356

Epoch 3/3
 - 23s - loss: 0.2560 - acc: 0.8972 - val_loss: 0.2714 - val_acc: 0.8910

 ROC-AUC - epoch: 3 - score: 0.843494

############ finish第 7 组输入 ############
```

计算总体预测正确率

In [0]:
```
sum_test = [0]*len(new_col)
for index, i in enumerate(new_col):
    for j in range(len(test)):
        if test2[i].iloc[j] == test[i].iloc[j]:
            sum_test[index] += 1
```

In [33]:
```
print([float(i / len(test)) for i in sum_test])
print(float(sum(sum_test) / (len(test) * len(new_col))))
```

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
1.0
```

In [0]:
```
sum_submission = [0]*len(new_col)
for index, i in enumerate(new_col):
    for j in range(len(submission)):
        if abs(submission2[i].iloc[j] - submission[i].iloc[j]) < 0.5:
            sum_submission[index] += 1
```

In [50]:
```
print([i / len(submission) for i in sum_submission])
print(sum(sum_submission) / (len(submission) * len(new_col)))
```

```
[0.9423333333333334, 0.9474666666666667, 0.8803333333333333, 0.8862, 0.907, 0.9318666666666666, 0.9238, 0.96, 0.91726
66666666667, 0.8975333333333333, 0.9882, 0.9788666666666667, 0.9265333333333333, 0.9569333333333333, 0.84006666666666
66, 0.8782666666666666, 0.9018666666666667, 0.9054, 0.898, 0.8999333333333334, 0.8890666666666667, 0.870933333333333
3, 0.8619333333333333, 0.8718666666666667, 0.8328, 0.8647333333333334, 0.8711333333333333, 0.8783333333333333, 0.784
2, 0.8195333333333333, 0.961, 0.7751333333333333, 0.8088666666666666, 0.8476, 0.8934, 0.8950666666666667, 0.981, 0.82
28666666666666, 0.828, 0.8491333333333333]
0.8893616666666667
```

评价

参照 https://challenger.ai/competition/fsauor2018 (https://challenger.ai/competition/fsauor2018), 这个问题的评价其实就是多个分类的f1 score 的平均值。

查看模型对新评论的预测

In [1]:
```
import pandas as pd
validation = pd.read_csv("ai_challenger_sentiment_analysis_validationset_20180816/sentiment_analysis_validationset.csv")
content = validation["content"]

validation_predict = pd.read_csv("submission_cut_add_prediction7.csv")
```

In [2]:
```
# 抽取含mentioned的变量名，用来筛选所有被提及的指标
predict_col = list(validation_predict.columns)
all_mentioned = [i for i in predict_col if "mentioned" in i]
```

In [6]:
```
sample = validation_predict.iloc[2,:] # 抽取第三条查看测试结果

# 筛选出mentioned为1的变量名
mentioned_1 = []
for i in all_mentioned:
    if sample[i] >= 0.5:
        mentioned_1.append(i)

positive = [i[:-10] + "_positive" for i in mentioned_1]
```

In [15]:
```
content.iloc[2]
```

Out[15]:
'"这家店是我目前吃到的最干净的串串店，目前看到最有意思的串串店。\n菜单是张试题卷，很有新意，和80后餐厅有点像，比较怀旧，装饮料的杯子也是80年代的搪瓷杯！\n\n整体味道还不错，至少锅底看起来挺干净的。不过锅底味道一般，点的鸳鸯锅，基本没吃过白锅，太咸，也不鲜。下次直接红锅就好了。\n\n2个人干掉将近60串串串，感觉还是挺能吃的。\n老板服务什么都还不错，小工就差很多，看来还需要多培训培训。\n当场点评还能送一杯冰粉，冰粉好大一杯，感觉不需要再点饮料了。\n\n唯一让我很想吐槽的就是油碟—花生酱之类的酱料竟然不能续加！！！不能续加！！！上来给我的一碟本来就不多，吃完了还不让加！！！还要再付钱！！！下次去吃，油碟必须加满！"'

In [13]:
```
sample[positive]
```

Out[13]:
```
service_waiters_attitude_positive          0.848832
environment_cleaness_positive              0.662375
dish_taste_positive                        0.176822
others_overall_experience_positive         0.168516
others_willing_to_consume_again_positive   0.558983
Name: 2, dtype: object
```

- 可以看到，“目前吃到的最干净的串串店”体现了环境干净度为正向评价（相符）
- “老板服务什么都还不错，小工就差很多，看来还需要多培训培训”体现服务态度还不错（估计是根据关键词“服务”判断的，“小工”表现此处未体现）
- “不过锅底味道一般，点的鸳鸯锅，基本没吃过白锅，太咸，也不鲜”体现了口味为负向评价（相符）
- 整体消费感受具有负向评价（相符）
- 回头率难说，既包含“下次去吃”，又具有一些负面评价。

In [16]:
```
sample = validation_predict.iloc[-100,:] # 抽取第三条查看测试结果

# 筛选出mentioned为1的变量名
mentioned_1 = []
for i in all_mentioned:
    if sample[i] >= 0.5:
        mentioned_1.append(i)

positive = [i[:-10] + "_positive" for i in mentioned_1]
```

In [17]:
```
content.iloc[-100]
```

Out[17]:
```
'"路过德基，不想吃午饭，就来吃他家的下午茶，团购很赞！团了双人下午茶套餐！一壶有机乌龙茶，一杯卡布奇洛咖啡、还有一个三层下午茶甜点！乌龙茶很香，服务员会主动加水，给你倒茶，服务很赞，卡布奇洛咖啡好喝不腻。他家马卡龙甜而不腻，是用蔬菜做的，喜欢可丽路，有焦香的味道，很Q，小汉堡是咸的，口感很清新，瑞士卷颜色很亮丽，奶油不腻，有坚果的小饼很脆很香，最喜欢。总而言之，他家的三层下午茶价格不贵，还好吃，性价比超高。之前吃的素食很棒，摆盘、食物口感都很好，连我这个肉食动物都爱上。"'
```

In [18]:
```
sample[positive]
```

Out[18]:
```
service_waiters_attitude_positive     0.910944
price_level_positive                  0.766989
price_cost_effective_positive         0.992662
price_discount_positive               0.236092
dish_taste_positive                   0.911531
dish_look_positive                    0.426262
others_overall_experience_positive    0.922351
Name: 14900, dtype: object
```

- 可以看到，“服务员会主动加水，给你倒茶，服务很赞”，服务员态度正向（相符）
- 口味赞了很多，dish_taste_positive相符
- “三层下午茶价格不贵，还好吃，性价比超高”，price_cost_effective_positive（相符）
- price_level_positive正向评价（相符）
- price_discount_positive此处未提及或者正向评价（团购很赞！团了双人下午茶套餐）（不符）
- dish_look_positive此处为正向评价（瑞士卷颜色很亮丽）（不符）
- 总体消费体验为正向（相符）

In [21]:
```
sample = validation_predict.iloc[193,:] # 抽取第三条查看测试结果

# 筛选出mentioned为1的变量名
mentioned_1 = []
for i in all_mentioned:
    if sample[i] >= 0.5:
        mentioned_1.append(i)

positive = [i[:-10] + "_positive" for i in mentioned_1]
```

In [20]:
```
content.iloc[193]
```

Out[20]:
```
'"第二次来。这次来吃酸菜鱼火锅，服务员好努力的销58一位的自助火锅，好卖力，我团好了券就不想改计划了婉拒之后服务员又hard sale各种饮料和飞饼。想说经理开会了吧执行不错可惜全程销的很生硬感觉不buy她的客人就做错事了一样，服务员带着不开心的脸走开了。酸菜鱼这次好酸呀额滴神，比冬阴公汤还酸，吃完鱼噜菜和面劲觉得好酸好酸。上饭那个男员工指甲有1cm那么长吧，瞬间觉得好恶心。最后隔壁桌有个素质极低的女的（见图绿裙biao），一声不吭跑来我的台上倒她的洗碗水！真是一声不吭！旁若无人当我这桌的人是空气，尼玛你桌上没缸你特马不会叫服务员拿给你吗？别人桌上的缸她理所当然老倒邋遢洗碗水！这特马什么素质！没下次。"'
```

In [22]:
```
sample[positive]
```

Out[22]:
```
service_waiters_attitude_positive          0.0204602
dish_taste_positive                        0.0428773
others_overall_experience_positive         0.0456303
others_willing_to_consume_again_positive    0.102588
Name: 193, dtype: object
```

- 这条评论里面一顿批评，其中涉及到服务员态度负向（相符）
- 味道负向（相符）
- 整体体验负向（相符）
- 回头率负向（相符）
- 环境负向未被预测出来（不相符） 可以考虑调低阈值
- 比如将0.5调低一点，获得更多mentioned，此处environment_cleaness_mentioned被漏检

In [24]:
```
sample[all_mentioned]
```

Out[24]:
```
location_traffic_convenience_mentioned               0.0363344
location_distance_from_business_district_mentioned   0.0235446
location_easy_to_find_mentioned                      0.0218006
service_wait_time_mentioned                          0.0589759
service_waiters_attitude_mentioned                    0.989928
service_parking_convenience_mentioned                0.0068225
service_serving_speed_mentioned                      0.0375848
price_level_mentioned                                 0.311168
price_cost_effective_mentioned                       0.0288143
price_discount_mentioned                              0.452502
environment_decoration_mentioned                      0.072329
environment_noise_mentioned                          0.0672673
environment_space_mentioned                          0.0850905
environment_cleaness_mentioned                        0.459837
dish_portion_mentioned                                0.160806
dish_taste_mentioned                                  0.577765
dish_look_mentioned                                    0.24382
dish_recommendation_mentioned                        0.0415919
others_overall_experience_mentioned                   0.987002
others_willing_to_consume_again_mentioned             0.712736
Name: 193, dtype: object
```

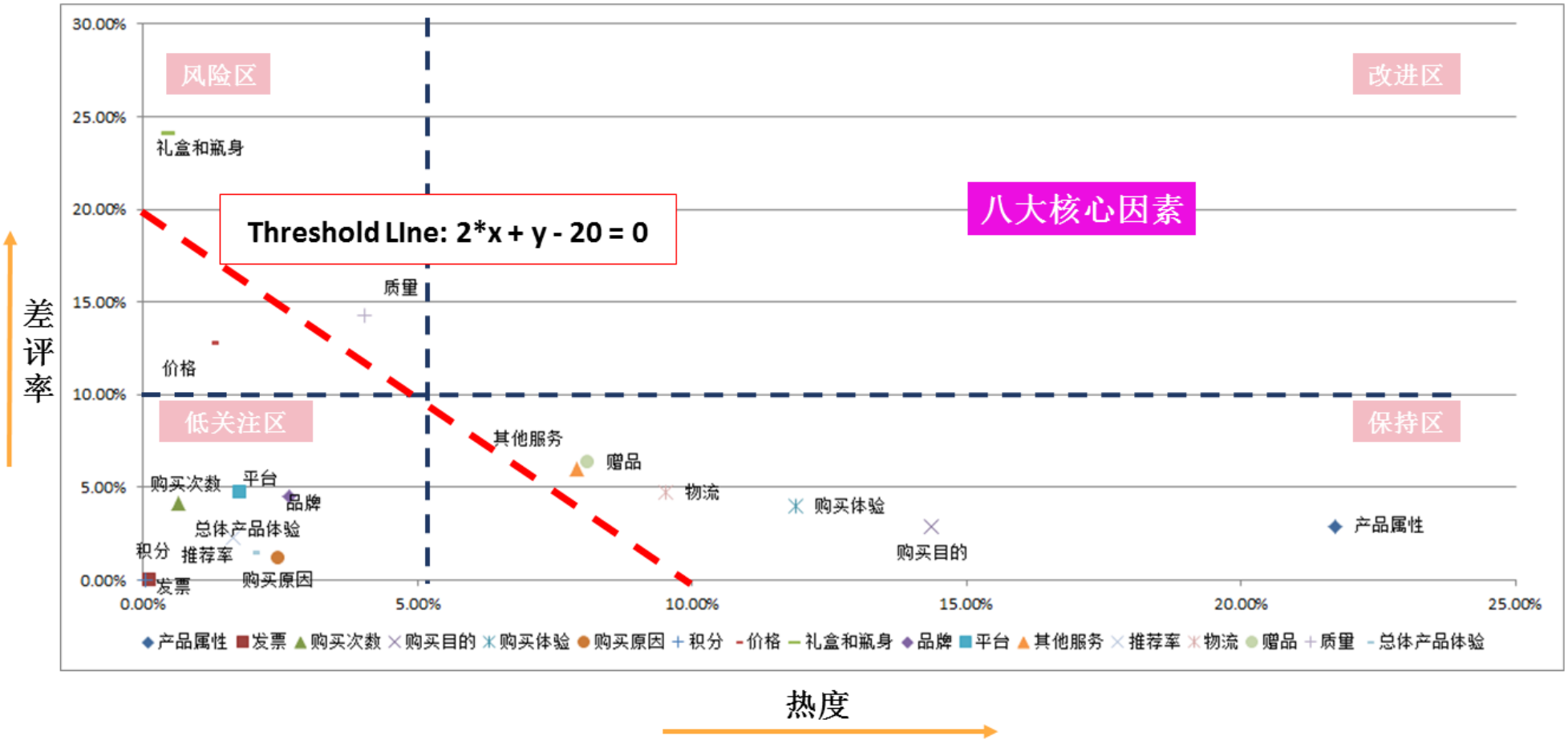
In [25]:
```
sample["environment_cleaness_positive"]
```

Out[25]: 0.038001805543899536

运用与实践

> 提及率及差评率分析

消费者最关注的核心因素



为客户针对性地提供建议

4.6 评论分析总结

二级指标	关注度	差评率	深入探究	E-marketing建议
产品属性	高	低	线上浓淡及香味不易分清	线下结合线上，“气味图书馆”活动
购买目的	中	低	送自己需求高	“爱自己”活动宣传
礼盒和瓶身	低	高	礼盒粗糙、瓶盖、喷头	礼盒优化，反馈瓶身问题给生产方
其他服务	中	中	仓库发货，漏发问题	精准发现该类问题并迅速补发解决
物流	中	低	包装简陋	改进包装，从细节赢得客户
赠品	中	中	小样粗糙、无喷头、无包装	小样赠品也需要注重细节
质量	低	高	尚未使用，担心假货	正品宣传，提供样品或线下体验

注：此处高低是七个指标间的比较

第二部分 基础理论部分

Q1: 机器学习中的**Loss**函数的作用为何？

回答：评判预测值和真实值之间的误差，Loss函数因模型而变，线性模型可用均方差表示，Logistics模型可用y*log(h) - (1-y)*log(1-h)表示

In [0]:
```
# remove the # before hint, what you find?
#hint('8fd9.4e2a.7b54.6848.53ef.4ee5.5199.5f88.591a.ff0c.4f46.662f.4e3b.8981.662f.8981.6d89.53ca.5230.ff1a.4c.6f.73.73.20.51fd.6570.7528.6765.8861.91cf.673a.5668.5b66.4e60.8fc7.7a0b.4e2d.6a21.578b.8868.73b0.7684.ff0c.6211.4eec.901a.8fc7.20.4c.6f.73.73.20.51fd.6570.6765.8fdb.884c.4f18.5316.6a21.578b.6216.8005.9009.62e9.6a21.578b')
```

Q2: 为什么 **SVM** 适合核函数的方法？（考虑基于拉格朗日距离的 **SVM** 的 **Loss** 函数）

回答：

In [0]:
```
#hint('8be6.60c5.8bf7.53c2.8003.6211.4eec.7684.8bfe.7a0b.89c6.9891.ff0c.4e00.4e2a.4e3b.8981.7684.70b9.662f.ff0c.6700.540e.8bc1.660e.20.53.56.4d.20.6a21.578b.7684.6027.80fd.53ea.4e0e.20.78.5f.69.78.5f.6a.20.7684.4e58.673a.76f8.5173.ff0c.6240.4ee5.6211.4eec.53ef.4ee5.65b9.4fbf.7684.628a.78.5f.69.20.78.5f.6a.20.6620.5c04.5230.67d0.4e2a.65b0.51fd.6570.4e0a.ff0c.4e0d.6539.53d8.5176.20.4c.6f.73.73.20.7684.5355.8c03.6027.5373.53ef')
```

Q3: 决策树的 **Loss** 函数是什么？随机森林是什么？

回答:

In [0]:
```
#hint("""4e00.3001.71b5.7684.548c.6700.5c0f.ff0c.8981.9009.62e9.4e00.4e2a.6761.4ef6.8ba9.8fd9.6b21.5206.7c7b.7684.4e24.8fb9.7ed3.679c.7adf.53ef.80fd.7684.2018.7eaf.2019.2c.20.4f8b.5982.ff0c.6211.4eec.6709.5b.32.2c.20.31.2c.20.30.2c.20.30.2c.20.30.2c.20.30.2c.20.31.2c.20.31.2c.20.31.2c.20.31.2c.20.31.5d.a.5982.679c.6211.4eec.9009.62e9.4e00.4e2a.6761.4ef6.662f.27.662f.4e0d.662f.31.27.ff0c.6211.4eec.53ef.4ee5.5206.6210.5b.30.2c.20.30.2c.20.30.2c.20.30.2c.20.32.5d.2c.20.5b.31.2c.20.31.2c.20.31.2c.20.31.2c.20.31.5d.ff0c.20.4e5f.53ef.4ee5.6761.4ef6.662f.27.662f.4e0d.662f.32.27.2c.20.a.6211.4eec.5c31.53ef.4ee5.5206.6210.5b.32.5d.2c.20.5b.31.2c.20.30.2c.20.30.2c.20.30.2c.20.30.2c.20.31.2c.20.31.2c.20.31.2c.20.31.2c.20.31.5d.ff0c.20.663e.7136.524d.8005.66f4.7eaf.3002.20.5982.679c.540c.5b66.4eec.5fd8.4e86.71b5.7684.6982.5ff5.ff0c.5927.5bb6.8d76.7d27.518d.67e5.4e00.4e0b.ff0c.7136.540e.8ba1.7b97.4e00.4e0b.8fd9.4e24.4e2a.5206.7c7b.7ed3.679c.7684.71b5.662f.591a.5927.ff1b.a.4e8c.3001.968f.673a.68ee.6797.662f.7528.5f88.591a.5c0f.7684.51b3.7b56.6811.7528.6765.6295.7968.7684.96c6.6210.6a21.578b.ff08.45.6e.73.65.6d.62.6c.65.ff09.ff0c.6bcf.4e2a.5c0f.51b3.7b56.6811.4f7f.7528.4e00.90e8.5206.7684.20.66.65.61.74.75.72.65.20.8fdb.884c.8bad.7ec3""")
```

插入：我们在这里快速的过一下如何在 Jupyter 中写数学公式。 这个其实很简单，如果我们要输入一个公式，例如A1, 那么，我们在Jupyter中输入`$A_i$`, 然后Enter， 是不是就变成了A_i? 其实两个`$$`之间的东西就是 Latex 的符号，`$..$`这个我们叫做inline模式，意思就是说你写出来的公式是和你的文字在一行里，如果你``` ``，这个公式就会单独是一行。

我们现在再试一个, 输入`$$\frac{P_i}{\sum_{j \in \mathbf{V}}^NP_j}$$`, 输完之后 Enter， 你看到了什么？

$$\frac{P_i}{\sum_{j \in \mathbf{V}}^N P_j}$$

这个时候会有同学说，可是这些符号，我怎么记得住呢？ 我给大家提供了一个参考手册，大家有空就看看 https://github.com/Artificial-Intelligence-for-NLP/comment-setimental-classification/blob/master/Latex-Symbols.pdf，熟能生巧。 (https://github.com/Artificial-Intelligence-for-NLP/comment-setimental-classification/blob/master/Latex-Symbols.pdf，熟能生巧。)

Q4: 使用**Latex** 写出来决策树希望找到一个 **feature**，这个 **feature** 使得熵的和最少的公式。

回答：

Q5: *贝叶斯公式的原理是什么？ 我们现在用的贝叶斯分类器为什么是“朴素贝叶斯”， 它为什么朴素？*

回答：

Q6: 神经网络的**Loss**函数的作用为何？

回答：

In [0]:
```
#hint("""795e.7ecf.7f51.7edc.91cc.9762.7684.4c.6f.73.73.51fd.6570.662f.7528.6765.8861.91cf.6a21.578b.7684.597d.574f.ff0c.4c.6f.73.73.51fd.6570.8d8a.5927.ff0c.9884.6d4b.4e0e.5b9e.9645.7684.8bef.5dee.8d8a.5927.ff0c.9884.6d4b.8d8a.4e0d.51c6.786e.3002.4e3a.4e86.8ba9.9884.6d4b.7ed3.679c.66f4.52a0.7cbe.51c6.ff0c.6211.4eec.8981.51cf.5c11.4c.6f.73.73.51fd.6570.ff0c.901a.8fc7.68af.5ea6.4e0b.964d.6cd5.ff0c.5229.7528.53cd.5411.4f20.64ad.4e0d.505c.8fed.4ee3.8c03.6574.795e.7ecf.7f51.7edc.4e2d.7684.53c2.6570.ff0c.627e.5230.4f7f.4c.6f.73.73.51fd.6570.6700.5c0f.7684.53c2.6570.ff0c.786e.5b9a.6a21.578b.3002""")
```

Q7: 神经网络的激活函数**(activation function)**起什么作用？ 如果没有激活函数会怎么样？

回答：

In [0]:
```
#hint('6fc0.6d3b.51fd.6570.7528.6765.8fdb.884c.975e.7ebf.6027.53d8.5316.ff0c.4e0d.65ad.5f97.975e.7ebf.6027.53d8.5316.4f7f.5f97.6211.4eec.28.7406.8bba.4e0a.29.53ef.4ee5.62df.5408.4efb.610f.51fd.6570.ff0c.8fd9.4e5f.662f.4e3a.4ec0.4e48.795e.7ecf.7f51.7edc.80fd.60.5b66.4e60.60.7684.539f.56e0.3002.795e.7ecf.7f51.7edc.91cc.8fb9.7684.60.5b66.4e60.60.5176.5b9e.5c31.662f.51fd.6570.62df.5408.7684.610f.601d')
```

Q8: 神经网络的**softmax**如何理解， 其作用是什么？ 在答案中写出**softmax**的**python**表达；

回答：

Q9: 简述 **normalized_1** 和**softmax**函数的相同点和不同点， 说明**softmax**相比**normalized_1**该函数的优势所在

```
output = np.array([y1, y2, y3])

normalized_1 = output / np.sum(output)
```

回答：

Q10: 写出**crossentropy**的函数表达式，说明该函数的作用和意义

回答：

..

..

---------- 休息一下，接下来是关于 Word2Vec的 ----------

Q11: 说明**word2vec**要解决的问题背景， 以及**word2vec**的基本思路， 说明**word2vec**比起之前方法的优势；

回答：

Q12: 说明**word2vec**的预测目标， **predication target,** 在答案中写出**skip-gram**和**cbow**的预测概率；

回答：

Q13: 请说明**word2vec**的两种常见优化方法，分别阐述其原理；

回答：

Q14: 请说明**word2vec**中哈夫曼树的作用；

回答：

Q15: 哈夫曼树如何构建？

In [0]:
```
#hint('a.31.2e.20.68.74.74.70.73.3a.2f.2f.67.69.74.68.75.62.2e.63.6f.6d.2f.68.65.69.6e.65.6d.61.6e.2f.70.79.74.68.6f.6e.2d.64.61.74.61.2d.73.74.72.75.63.74.75.72.65.73.2f.62.6c.6f.62.2f.6d.61.73.74.65.72.2f.35.2e.25.32.30.48.65.61.70.2d.62.61.73.65.64.25.32.30.53.74.72.75.63.74.75.72.65.73.2f.68.75.66.66.6d.61.6e.2e.70.79.a.32.2e.20.68.74.74.70.73.3a.2f.2f.67.69.74.68.75.62.2e.63.6f.6d.2f.52.61.52.65.2d.54.65.63.68.6e.6f.6c.6f.67.69.65.73.2f.67.65.6e.73.69.6d.2f.62.6c.6f.62.2f.33.64.35.61.32.31.63.31.63.38.31.32.38.63.62.38.64.64.34.66.36.65.35.31.65.39.65.66.33.64.63.35.61.66.30.30.30.38.37.31.2f.67.65.6e.73.69.6d.2f.6d.6f.64.65.6c.73.2f.64.65.70.72.65.63.61.74.65.64.2f.77.6f.72.64.32.76.65.63.2e.70.79.23.4c.36.37.30.22.a.33.2e.20.68.74.74.70.73.3a.2f.2f.77.77.77.2e.77.69.6b.69.77.61.6e.64.2e.63.6f.6d.2f.65.6e.2f.48.75.66.66.6d.61.6e.5f.63.6f.64.69.6e.67.a')
```

Q16: 在**gensim**中如何实现词向量？ 请将**gensim**中实现词向量的代码置于答案中

回答：

Q17: 请说出除了 **skip-gram**和**cbow**的其他**4**中词向量方法的名字， 并且选取其中两个叙述其基本原理。

回答：

In [0]:
```
#hint('a.4f.6e.65.68.6f.74.2c.20.47.6c.6f.76.65.2c.43.6f.76.65.2c.45.4d.4c.6f.a')
```

---------- 休息一下，接下来是关于 Keras 和 Tensorflow 使用的 ----------

大家先熟悉一下什么是MNIST数据集：

> http://yann.lecun.com/exdb/mnist/ (http://yann.lecun.com/exdb/mnist/)
>
> https://en.wikipedia.org/wiki/MNIST_database (https://en.wikipedia.org/wiki/MNIST_database)

Q18: 参考**keras**参考手册，构建一个机器学习模型，该模型能够完成使用**DNN(deep neural networks)** 实现**MNIST**数据集的分类；

关键代码:

Q19:参考**tensorflow**的参考手册，构建一个机器学习模型，该模型能够完成使用**DNN(deep neural networks)**实现**MNIST**数据集的分类；

关键代码：

In [0]:
```
#hint('68.69.6e.74.73.3a.74.65.6e.73.6f.72.66.6c.6f.77.5b9e.73b0.4d.4e.49.53.54.20.68.74.74.70.73.3a.2f.2f.67.69.74.68.75.62.2e.63.6f.6d.2f.74.65.6e.73.6f.72.66.6c.6f.77.2f.74.65.6e.73.6f.72.66.6c.6f.77.2f.62.6c.6f.62.2f.6d.61.73.74.65.72.2f.74.65.6e.73.6f.72.66.6c.6f.77.2f.65.78.61.6d.70.6c.65.73.2f.75.64.61.63.69.74.79.2f.32.5f.66.75.6c.6c.79.63.6f.6e.6e.65.63.74.65.64.2e.69.70.79.6e.62')
```

Q20: 参考**keras**和**tensorflow**对同一问题的实现，说明**keras**和**tensorflow**的异同；

回答：

Q21: tensorflow 使用 **Graph** 计算机制的优缺点是什么？

回答：

Q22: Q18， Q19 的**tensorflow** 或 **keras** 模型的训练时准确率和测试集准确率分别是多少？

回答：

In [0]:
```
#### Q23: 训练时准确率大于测试集准确率的现象叫什么名字，在神经网络中如何解决该问题？(至少提出5个解决方法)
```

回答：

In [0]:
```
#### Q24: 请使用自己的语言简述通过正则化 (regularization)减小过拟合的原理；
```

回答：

In [0]:
```
#### Q25: 在tensorflow官方实例中给出的fully connected 神经网络的分类模型中，数据进行了哪些预处理，这些预处理的原因是什么？
```

回答：

---------- 休息一下，接下来是关于 RNN 和 CNN 的 ----------

Q26: 简述**CNN**的原理

回答：

Q27: CNN的 **Spatial Invariant**是什么意思？ 是如何做到的？

回答：

Q28: CNN增加了很多层数，这些层数使用 **filter** 进行计算。 按说需要拟合的参数变得很多，请问 **CNN** 是如何解决这个问题的，如何加快速度的？

回答：

In [0]:
```
#hint('a.6d.61.69.6e.20.70.6f.69.6e.74.73.3a.20.50.6f.6f.6c.69.6e.67.2c.20.50.61.72.61.6d.65.74.65.72.20.53.68.61.72.69.6e.67.a')
```

Q29: CNN中的 Batch Normalization有什么意义？

回答:

Q30: CNN中的 Pooling 起到什么作用？

回答：

Q31: CNN中的 Fully Connect起到什么作业？

回答：

Q32: 深度网络中的权值初始化有什么讲究？

回答：

Reading: 参照 **Keras** 和 **Tensorflow** 的示例，手敲使用 **keras, tensorflow + CNN** 实现**MNIST**分类的问题：

- https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py (https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py)
- https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/udacity/4_convolutions.ipynb (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/udacity/4_convolutions.ipynb)

把代码手敲一遍

Q33: 简述**RNN**解决的问题所具有的特点；

回答：

Q34: 写出**RNN**实现时间或者序列相关的数学实现**(**见课程**slides)**

回答：

Q35: 简述**RNN**的两种重要变体的提出原因和基本原理？

回答:

Q36: Attentional RNN 以及 **Stacked RNN** 和 **Bi-RNN** 分别是什么，其做了什么改动？

回答：

Reading 和 **CNN** 类似，请在 **Keras,Tensorflow**中查找如何实现 **RNN** 模型

- https://github.com/Artificial-Intelligence-for-NLP/References/blob/master/AI%20%26%20Machine%20Learning/Hands.On.TensorFlow.pdf (https://github.com/Artificial-Intelligence-for-NLP/References/blob/master/AI%20%26%20Machine%20Learning/Hands.On.TensorFlow.pdf)

第二部分： 项目解决过程

代码主要在 Pycharm 里边写，jupyter 里边写一个关键步骤就行

Q37: 要实现文本分类或情感分类，文本信息需要进行哪些初始化操作？自己手工实现，**keras**提供的**API**，**tenorflow**提供的**API**，分别是哪些？请提供关键代码置于下边回答中

回答：

In [0]:
```
#hint('id_to_word, word_to_id, padding, batched')
```

Q38 在没有预训练的词向量时候， **keras** 如何实现**embedding**操作，即如何依据一个单词的序列获得其向量表示？

回答：

Q 39: 在有预先训练的词向量时候，**keras**和**tensorflow**又如何实现**embeding**操作

回答：

Q40：基于上文进行的数据预处理，使用**keras**和**tensorflow**如何构建神经网络模型？请提供关键代码

好的 现在开始切入正题 --

其实，我们解决实际问题的时候，很少自己从头到尾写一个神经网络模型，我们往往是找一个效果比较好的类似问题的模型，然后在这个问题上改造。 或者我们在去一个公司的时候，接手的工作也往往是改动以前的模型，所以我们解决这个语义分类问题我们也首先是找一个类似的问题，然后参考一个模型进行修改，变成能够解决我们这个问题的模型。

我们以上所以的理论知识，都是用来支持我们做修改，能够看懂别人为何要这样写，然后自己要改哪里。

kaggle上的“恶意评价识别”这个项目和我们的这个项目是类似的, 大家请首先在这个的 Kernel 里边找到一个公开代码的示例，然后选择一个自己能够看懂且效果较好的模型进行改造。

- https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge)
- https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/kernels (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/kernels)

Kaggle这个问题和我们的问题类似，但是并不是完全一样， 其中最不一样的其实是我们的期望的结果这里, Kaggle 这里的输出是5个类别，然后类别的是0~1直接的数字来预测是否是这个类别，然后我们的客户评价问题中，打标对20个分类的-2, -1, 0, 1四个标记. 一种最简单的方法，是把这个20分类问题变成80分类问题，然后每个分类的输出是0或者1.

Q41. 依据**Kernel** 中选择方法，对数据和代码进行改造，使其符合选择该问题。

回答：

Q42. 你现在的模型的准确率是多少？ 如何知道你的模型是不是真的学习了 而不是随机的进行猜测？

回答：

Q43. 你的模型现在准确度不高的原因，你猜测主要是什么？

回答：

Q43. 如前文所述，这个问题很难，其实现在也没有什么万灵药方法。 所以需要同学们多想想如何有效， 可以给大家参考的优化方式有， 修改**vocabulary size, embedding size,**去掉停用词，重新组合词组等。 并且结合使用**LSTM， GRU， Bi-RNN， Stacked， Attentional, regularization,** 等各种方法组合进行模型的优化， 至少进行**10**次优化，每次优化请按照以下步骤填写：

回答：

回答：

---这是一个实例----

第1次优化：

1. 存在的问题： loss下降太慢；
2. 准备进行的优化：减小模型的神经单元数量；
3. 期待的结果：loss下降加快；
4. 实际结果：loss下降的确加快(或者并没有加快)
5. 原因分析：模型神经元数量减小，收敛需要的次数减少，loss下降加快

---你的实验优化结构记录在此---

第1次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第2次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第3次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第**4**次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第**5**次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第**6**次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第**7**次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

第**9**次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：

5. 原因分析：

第10次优化：

1. 存在的问题：
2. 准备进行的优化：
3. 期待的结果：
4. 实际结果：
5. 原因分析：

最后一步： 使用Flask、Bottle、Bootstrap变成一个网络应用并且部署在服务器上，这样别人就可以直接通过网址访问你的应用啦。

最后一步，我们使用Bottle，Bootstrap,Flask等工具进行可视化现实，做出网页能够访问的形式，就像我们的第一个项目一样 😁.

本次项目的总结

请写项目的总结报告，描述此次项目的主要过程，其中遇到的问题，以及如何解决这些问题的，以及有什么经验和收获。

恭喜你，你完成了一个十分复杂的问题， 能完成这个问题，求是求是，你的能力其实已经达到了国内绝大多数公司的要求，你缺的只是熟练程度。 多多在Kaggle， 阿里天池里边找一些自己感兴趣的问题，多练习练习。 熟能生巧。