Project 2步骤

说明： movie\_comment\*.json 中是爬取了246个电影中，每个电影短评的前50页，约有2000条短评或者更少。这样的考虑的原因有如下几点：

绝大部分电影的短评数量非常庞大，基本都在2w以上，完全爬取并不现实；

豆瓣电影短评的排名是基于豆瓣官方的某种算法排序的——“ 短评的排序是将豆瓣成员的投票加权平均计算后的结果，通过算法的调校，更好地反映短评内容的价值。”；

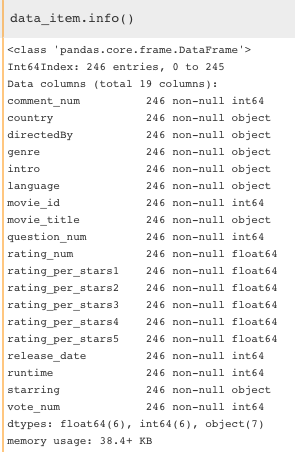
查看短评的排序，可以发现短评有用数基本是指数递减的；

items中comment\_id是每条短评的唯一ID编码，people是每个短评人的用户名，可作为点评人的唯一ID编码。

实际爬取的过程中，由于代码要求先存下来每个电影的短评URL，然后同时批量每页爬取，很可能是截止到短评的代码存在bug，而导致电影的短评数量并不是完全一样的，每个电影的短评数从1060到819不等分布：

1. 数据清理DataWash.py 清理和去重

收集到的json数据非常的raw，所以首先进行数据清理。得到 DataFrame



2.添加label 信息，

drop掉用于爬虫时候检查爬取质量的URL信息，并且添加了label信息，标示出给出3星及其以上的为“喜欢”，其他为"不喜欢"：



##### 3. 用朴素贝叶斯完成中文文本分类器

##### 3.1 打印出来正负样本的个数时，发现

# 1 993

# 0 47

# Name: label, dtype: int64

##### 正负样本极为不平衡

##### 所以 预处理过程加上复制负样本20遍使得正负样本平衡，并且drop停用词，最后生成乱序的训练集

##### while n < 20:

##### preprocess\_text(data\_com\_X\_0.content.dropna().values.tolist(), sentences, 'nlike')

##### n += 1

##### 3. 2 接下来就是通过交叉验证，在朴素贝叶斯分类器下构建模型，给出准确率：

##### 得到准确率

##### 0.910392190906

##### 看起来还不错，但是但其实由于负样本太不丰富，且数据总量也小，所以测试短评时并不一定能给出理想的结果，如下面自定义的中文文本分类器例子 SelfIdentyBayes.py ：

##### import re

##### from sklearn.feature\_extraction.text import CountVectorizer

##### from sklearn.model\_selection import train\_test\_split

##### from sklearn.naive\_bayes import MultinomialNB

##### class TextClassifier():

##### def \_\_init\_\_(self, classifier=MultinomialNB()):

##### self.classifier = classifier

##### self.vectorizer = CountVectorizer(analyzer='word'

##### ,ngram\_range=(1,4)

##### ,max\_features=20000)

##### def features(self, X):

##### return self.vectorizer.transform(X)

##### 

##### def fit(self, X, y):

##### self.vectorizer.fit(X)

##### self.classifier.fit(self.features(X), y)

##### 

##### def predict(self, x):

##### return self.classifier.predict(self.features([x]))

##### 

##### def score(self, X, y):

##### return self.classifier.score(self.features(X), y)

##### 得到准确率 0.913223140496

##### 4 用SVC完成中文文本分类器 见 svc.py

与上面类似的，我们用SVC构建模型，看下效果会如何：

import re

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

class TextClassifier():

def \_\_init\_\_(self, classifier=SVC(kernel='linear')):

self.classifier = classifier

self.vectorizer = TfidfVectorizer(analyzer='word'

,ngram\_range=(1,4)

,max\_features=20000)

def features(self, X):

return self.vectorizer.transform(X)

def fit(self, X, y):

self.vectorizer.fit(X)

self.classifier.fit(self.features(X), y)

def predict(self, x):

return self.classifier.predict(self.features([x]))

def score(self, X, y):

return self.classifier.score(self.features(X), y)

text\_classifier=TextClassifier()

text\_classifier.fit(x\_train,y\_train)

print(text\_classifier.predict('一点 不觉得震撼'))

print(text\_classifier.predict('好看'))

print(text\_classifier.score(x\_test,y\_test))

得到准确率

**0.971074380165**

**目前看来，支持向量机svc 的分类效果最好**

##### 5. 用RNN做中文文本分类 见RNN.py

##### 最终得到准确率Accuracy: 0.925620 不如 svc

遇到的问题和技巧：

①在自定义类中文文本分类器时，需要引入DataWash.py文件中调用另一个SeltIdentyBayes.py文件的类TextClassifier(), 通过查询了解到

如果是在同一个 module中(也就是同一个py 文件里),直接用就可以

如果在不同的module里,例如

a.py里有 class A:

b.py 里有 class B:

如果你要在class B里用class A 需要在 b.py的开头写上 from a import A

②当我尝试用pd.read\_json('A.json') 读取文件时，给我报错了，报错部分情况如下：File Expected object or value, 一顿百度后发现原来是 json 格式错误问题，。需要将文件里面的字典作为元素保存在列表当中才行，即逐条json 文件需要放在[] 里面读取。

③ 开始受到 IMDB数据集的影响，一直想要直接利用Project 1 爬取的文本，预处理成词向量的形式，奈何需要下载词库和掌握word2vec的用法，于是换了一种思路，直接爬取和解析json文件。

虽然解析的过程不如词向量简介，但是在没有语料库对照的情况下，不失为一种好方法

##### 总结放在

##### 1. 仅通过短评文本来分类点评人对电影的喜好确实浅薄了些，因为太多的短评内容与打分之间的关联事很弱的，有着很大的随意性：“我反正给不了5星”->4星->like、“还是经典，有机会以后要看高清的”->0星->nlike。。。。更何况，我们的训练数据label有着较为严重的样本不平衡的问题。所以，如果真的想较为全面且准确的预测点评人短评的喜好程度的话，就不仅需要更多且平衡label的训练数据，还需要考虑更多维度的信息，如横向对比电影简介、点评人简介以及点评人曾短评过的电影等信息，来判断全方面的判断点评人的喜好。

##### 2. 未能完整爬取每个点评人与电影短评相关的特征信息，豆瓣上的用户群体庞大，每个豆瓣用户的电影短评（不仅仅是Top250电影）也应该是非常有用的特征信息，但是因时间精力有限为能爬取。

##### 3.使用的集中预测方法中除了svc ,准确率都并不高。原因可能很多，诸如：数据量太少、文本特征不够明显、需要加入更多的训练特征等地。

##### 4. 文本分类的方式有很多，比如这里还没有涉及到的Facebook FastText有监督学习和无监督学习，CNN中文文本分类，GRU来完成中文文本分类。这些方式都很经典，都值得我们去探究和比较。

##### 5. 如果有更多的训练方式，可以横向比较他们的差异，总结出每种模型的适用范围和优势，劣势，再将数据可视化，更加直观看出模型和数据的特点。

##### 后续工作，可以把每个点评人的个人简介文本作为训练数据，对地域信息（分类）或者被关注数（回归）作为label，构建模型。如此一来所得到的分类器，可以用以预测什么样的个人简介可能来自哪个省份或者预测其可以得到多少的被关注数，以及其它更多的预测。

**参考文献：**

《Keras入门与实践》 以及 遇到问题时查阅的各种文档和论坛。