Step01:导入数据

In [498]:

```
import pandas as pd
import numpy as np
import nltk
import re
from gensim.models import word2vec
import gensim
import nltk.tokenize as tk
from nltk.corpus import stopwords
import nltk.stem.snowball as sb
import nltk.stem.porter as pt
import nltk.stem.lancaster as lc
from sklearn.linear_model import LogisticRegression
import copy
import math
```

In [409]:

```
train_u = pd.read_table("train.tsv")
test_n = pd.read_table("test_news.tsv")
test_u = pd.read_table("test.tsv")
train_n = pd.read_table("train_news.tsv")
```

In [473]:

```
test_n
```

Out[473]:

	Nid	Category	SubCategory	Title	Abstract
0	N55528	lifestyle	lifestyleroyals	The Brands Queen Elizabeth, Prince Charles, an...	Shop the notebooks, jackets, and more that the...
1	N18955	health	medical	Dispose of unwanted prescription drugs during ...	NaN
2	N61837	news	newsworld	The Cost of Trump's Aid Freeze in the Trenches...	Lt. Ivan Molchanets peeked over a parapet of s...
3	N53526	health	voices	I Was An NBA Wife. Here's How It Affected My M...	I felt like I was a fraud, and being an NBA wi...
4	N38324	health	medical	How to Get Rid of Skin Tags, According to a De...	They seem harmless, but there's a very good re...
...	...	...	...	...	...
42411	N63550	lifestyle	lifestyleroyals	Why Kate & Meghan Were on Different Balconies ...	There's no scandal here. It's all about the or...
42412	N30345	entertainment	entertainment-celebrity	See the stars at the 2019 Baby2Baby gala	Stars like Chrissy Teigen and Kate Hudson supp...
42413	N30135	news	newsgoodnews	Tennessee judge holds lawyer's baby as he swea...	Tennessee Court of Appeals Judge Richard Dinki...
42414	N44276	autos	autossports	Best Sports Car Deals for October	NaN
42415	N39563	sports	more_sports	Shall we dance: Sports stars shake their leg	NaN

42416 rows × 5 columns

得到训练集top10，和测试集top10

In [411]:

```
def getTopN(petro,N):
    res = []
    for i in range(0,len(petro)):
        res.extend(ImpressT(petro["Impression"][i]))
    counter = {}
    for i in res:
        counter[i] = counter.get(i,0)+1
    topN = sorted(counter,key = counter.get,reverse = True)[:N]
    return topN
trainTop = getTopN(train_u,10)
#训练集热门新闻
```

In [412]:

```
def gethistory(i):
    res = (test_u["History"][i]).split(" ")
    return res
def getTopNtest(petro,N):
    res = []
    for i in range(0,len(petro)):
        res.extend(gethistory(i))
    counter = {}
    for i in res:
        counter[i] = counter.get(i,0)+1
    topN = sorted(counter,key = counter.get,reverse = True)[:N]
    return topN
testTop = getTopNtest(test_u,10)
#测试集热门新闻
```

In [413]:

```
train_n.loc[train_n["Nid"]=="N61837",["Title","Abstract"]]
```

Out[413]:

	Title	Abstract
2	The Cost of Trump's Aid Freeze in the Trenches...	Lt. Ivan Molchanets peeked over a parapet of s...

获得正样本

In [414]:

```
def ImpressT(rec):
    res = []
    rec = rec.split(' ')
    for i in rec:
        if i[-1]=='1':
            res.append(i[:-2])
    return res
def ImpressF(rec):
    res = []
    rec = rec.split(' ')
    for i in rec:
        if i[-1]=='0':
            res.append(i[:-2])
    return res
Impress(train_u["Impression"][0])
```

Out[414]:

```
['N357', 'N46029', 'N56598']
```

In [415]:

```
def newtrainT(petro):
    res = {}
    for i in range(0,len(petro)):
        res[petro["Uid"][i]]=ImpressT(petro["Impression"][i])
    return res
def newtrainF(petro):
    res = {}
    for i in range(0,len(petro)):
        res[petro["Uid"][i]]=ImpressF(petro["Impression"][i])
    return res
trainT = newtrainT(train_u) #每个user的正样本
trainF = newtrainF(train_u) #每个user的负样本
```

结构化特征向量定义

In [416]:

```
CateCate  = np.unique(np.hstack((train_n["Category"].unique(),test_n["Category"].unique())))
CateScore = {}
temp = 10
for i in CateCate:
    CateScore[i] = temp
    temp+=10
CateScore #每个Cate的值对应   字典
```

Out[416]:

```
{'autos': 10,
 'entertainment': 20,
 'finance': 30,
 'foodanddrink': 40,
 'games': 50,
 'health': 60,
 'kids': 70,
 'lifestyle': 80,
 'middleeast': 90,
 'movies': 100,
 'music': 110,
 'news': 120,
 'northamerica': 130,
 'sports': 140,
 'travel': 150,
 'tv': 160,
 'video': 170,
 'weather': 180}
```

In [417]:

```
SubCate  = np.unique(np.hstack((train_n["SubCategory"].unique(),test_n["SubCategory"].unique())))
SubScore = {}
temp = 10
for i in SubCate:
    SubScore[i] = temp
    temp+=10
SubScore  #每个Subcate的值对应  字典
```

Out[417]:

```
{'ads-latingrammys': 10,
 'ads-lung-health': 20,
 'advice': 30,
 'animals': 40,
 'autosbuying': 50,
 'autoscartech': 60,
 'autosclassics': 70,
 'autoscompact': 80,
 'autosenthusiasts': 90,
 'autoshybrids': 100,
 'autoslosangeles': 110,
 'autosluxury': 120,
 'autosmidsize': 130,
 'autosmotorcycles': 140,
 'autosnews': 150,
 'autosownership': 160,
 'autospassenger': 170,
 'autosresearch': 180,
```

In [445]:

```
#文本预处理
pattern = re.compile("[^a-zA-Z0-9\n ]") #用于去除标点
def sentence2word(text):
    text = re.sub(pattern,"",text).lower() #转换为小写，并去除标点
    tokens = tk.word_tokenize(text) #进行单词切分
    return tokens
def getwordlist(txt_train):
    for i in range(0,txt_train.shape[0]):
        txt_train['Title'][i] = sentence2word(str(txt_train['Title'][i]))
        txt_train['Abstract'][i] = sentence2word(str(txt_train['Abstract'][i]))
    word_list = list(txt_train['Abstract'])+list(txt_train['Title'])
    return word_list
wlist = getwordlist(train_n)
```

In [446]:

```
#得到词向量模型
w2v_model = gensim.models.Word2Vec(wlist, vector_size=20, window=5, min_count=1, workers=4)
```

In [447]:

```
vocab = w2v_model.wv.index_to_key #得到单词表
```

In [501]:

```
#得到句向量
def sen2vec(text):
    i=0
    lis = sentence2word(text)
    #print(lis)
    sum = np.zeros(20,dtype=np.float32)
    for word in lis:
        if word in vocab:
            sum += w2v_model.wv.vectors[w2v_model.wv.key_to_index[word]]
            i+=1
    try:
        sum = sum/i
    except ZeroDivisionError:
        sum = sum
    if math.isnan(sum[0]):
        return np.zeros(20,dtype=np.float32)
    #print(sum,i)
    return sum
```

In [502]:

```
col = ["CateS","SubS"]
for i in range(20):
    col.append("TitVec"+str(i))
for i in range(20):
    col.append("AbsVec"+str(i))
```

得到新闻结构特征向量

In [503]:

```
def charicVec(news):
    col = ["CateS","SubS"]
    for i in range(20):
        col.append("TitVec"+str(i))
    for i in range(20):
        col.append("AbsVec"+str(i))
    temp = []
    res = []
    length = news.shape[0]
    for i in range(0,length):
        temp = []
        temp.append(CateScore[news["Category"][i]])
        temp.append(SubScore[news["SubCategory"][i]])
        if (news["Title"][i]) != np.nan:
            temp.extend(sen2vec(str(news["Title"][i])))
        else:
            temp.extend(sum = np.zeros(20,dtype=np.float32))
        if (news["Abstract"][i]) != np.nan:
            temp.extend(sen2vec(str(news["Abstract"][i])))
        else:
            temp.extend(sum = np.zeros(20,dtype=np.float32))
        res.append(copy.deepcopy(temp))
    res = pd.DataFrame(res,columns=col)
    return res
trainChar = charicVec(train_n)
testChar = charicVec(test_n)
```

In [504]:

```
train_nC = pd.concat([train_n,trainChar],axis=1)
train_nC.drop(["Category","SubCategory","Title","Abstract"],axis=1,inplace=True)
#训练集新闻特征向量
```

In [505]:

```
test_nC = pd.concat([test_n,testChar],axis=1)
test_nC.drop(["Category","SubCategory","Title","Abstract"],axis=1,inplace=True)
#测试集新闻特征向量
```

In [508]:

```
#某新闻特征向量
test_nC.loc[test_nC["Nid"]=="N3347",col].values
```

Out[508]:

```
array([[ 1.40000000e+02,  1.00000000e+03,  1.12987089e+00,
        -7.24367082e-01,  1.60059357e+00,  1.20050967e+00,
         2.49791220e-02, -1.23277020e+00, -5.00418723e-01,
         7.83931732e-01, -2.00756812e+00, -2.02456787e-01,
         7.97964573e-01,  9.00134742e-01,  6.51192009e-01,
         4.64457005e-01, -5.49746454e-01,  1.16794896e+00,
        -9.48290765e-01,  1.21693611e+00,  1.22371280e+00,
        -1.51993608e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00]])
```

用于得到正负样本的函数

In [509]:

```
#得到每个用户his
def gethistory(petro):
    res = {}
    for i in range(0,len(petro)):
        res[petro["Uid"][i]] = petro["History"][i].split(" ")
    return res
#得到每个用户未读的热门新闻
def getNeg_train(petro,top):
    res = {}
    for i in range(0,len(petro)):
        temp = train_history[petro["Uid"][i]] ###修改his可以获得训练集与测试集不同负样本
        tempNe = []
        for e in top:
            if e not in temp:
                tempNe.append(e)
        res[petro["Uid"][i]] = copy.deepcopy(tempNe)
    return res
#得到每个用户未读的热门新闻
def getNeg_test(petro,top):
    res = {}
    for i in range(0,len(petro)):
        temp = test_history[petro["Uid"][i]] ###修改his可以获得训练集与测试集不同负样本
        tempNe = []
        for e in top:
            if e not in temp:
                tempNe.append(e)
        res[petro["Uid"][i]] = copy.deepcopy(tempNe)
    return res
```

得到正负样本

In [510]:

```
#得到训练集每个用户history （以下数据结构均为字典形式 Uid:[Nid]）
train_history = gethistory(train_u)
```

In [511]:

```
#得到测试集每个用户history
test_history = gethistory(test_u)
```

In [512]:

```
#得到训练集未读的热门新闻样本
train_Neg = getNeg_train(train_u,trainTop)
```

In [513]:

```
#得到测试集未读的热门新闻样本
test_Neg = getNeg_test(test_u,testTop)
```

得到特征向量与标签的函数

In [514]:

```
#用于得到训练集每个Uid特征向量与标签
def gettrain_xy(Uid):
    x = []
    y = []
    for i in train_history[Uid]:
        x.append((train_nC.loc[train_nC["Nid"]==i,col]).values.tolist()[0])
        y.append(1)
    for i in train_Neg[Uid]:
        x.append((train_nC.loc[train_nC["Nid"]==i,col]).values.tolist()[0])
        y.append(0)
    return x,y
```

In [515]:

```
#用于得到测试集每个Uid特征向量与标签
def gettest_xy(Uid):
    x = []
    y = []
    for i in test_history[Uid]:
        x.append((test_nC.loc[test_nC["Nid"]==i,col]).values.tolist()[0])
        y.append(1)
    for i in test_Neg[Uid]:
        x.append((test_nC.loc[test_nC["Nid"]==i,col]).values.tolist()[0])
        y.append(0)
    return x,y
```

在训练集上测试准确度

In [460]:

```
reg = LogisticRegression(max_iter=1000) #最大迭代次数设置为100
```

In [461]:

```
#用于得到准确率
TP = 0
FP = 0
TN = 0
FN = 0
threshold = 0.9
```

In [462]:

```
#用于得到训练集中要预测的新闻
def getTrain_news(i):
    train_news = train_u.loc[i,"Impression"]
    train_news = train_news.split(" ")
    res = []
    for i in train_news:
        res.append(i[:-2])
    return res
```

In [463]:

```
#用于得到当前Uid的逻辑回归模型
def getReg(i):
    Uid = train_u.loc[i,"Uid"]
    xy = gettrain_xy(Uid)
    global reg
    reg.fit(xy[0],xy[1])
    return 0
```

In [520]:

```
#得到测试集的预测结果
def getTrainRes():
    global TP,FP,TN,FN,reg
    for i in range(0,len(train_u)):
        Uid = train_u.loc[i,"Uid"]
        tempNews = getTrain_news(i)
        getReg(i)
        for e in tempNews:
            if ( (reg.predict_proba([(train_nC.loc[train_nC["Nid"]==e,col]).values.tolist()[0]]))[0
                TP += 1
            if ( (reg.predict_proba([(train_nC.loc[train_nC["Nid"]==e,col]).values.tolist()[0]]))[0
                FP += 1
            if ( (reg.predict_proba([(train_nC.loc[train_nC["Nid"]==e,col]).values.tolist()[0]]))[0
                TN += 1
            if ( (reg.predict_proba([(train_nC.loc[train_nC["Nid"]==e,col]).values.tolist()[0]]))[0
                FN += 1
    return 0
```

In [521]:

```
getTrainRes()
```

Out[521]:

```
0
```

In [522]:

```
(TP+TN)/(TP+TN+FP+FN)
```

Out[522]:

```
0.5418797091054917
```

得到在测试集上的结果

In [516]:

```
#用于得到每个test里Uid的待评测新闻
def getTestnews(i):
    testnews = test_u.loc[i,"Impression"]
    testnews = testnews.split(" ")
    return testnews
```

In [517]:

```
#用于得到当前Uid的逻辑回归模型_test
def getReg_test(i):
    Uid = test_u.loc[i,"Uid"]
    xy = gettest_xy(Uid)
    global reg
    reg.fit(xy[0],xy[1])
    return 0
```

In [518]:

```
def getPred():
    reslis = []
    for i in range(0,len(test_u)):
        Uid = train_u.loc[i,"Uid"]
        sample = getTestnews(i)
        getReg_test(i)
        strTemp = ""
        for e in sample:
            strTemp = strTemp +  str((reg.predict_proba([(test_nC.loc[test_nC["Nid"]==e,col]).value
        reslis.append([strTemp])
    coll = ["Predict"]
    res =  pd.DataFrame(reslis,columns=coll)
    return res
```

In [523]:

```
Predict = getPred()
```

In [524]:

```
result = pd.concat([test_u,Predict],axis=1)
```

In [526]:

```
result.to_csv("test_res.tsv",index=False, sep='\t', encoding='utf-8')
```