**Predicting the marketing strategy of e-commerce products**

**based on text big data**

By Ding Lichao and Hu Zijiao

2021/10/16

**1 Introduction**

Nowadays with more and more merchants joining the e-commerce camp, competition in Internet marketing has become increasingly intense. Meanwhile, as more and more customers participate in online reviews and interactions, comprehensive analysis of reviews and ratings plays an increasingly important role in understanding customers’ pain points and specifying future strategies. Specifically speaking, in Amazon’s design, customers are allowed to choose a number from 1 to 5 to express their satisfaction level with the product, write any text-based messages as reviews and freely vote for other reviews they consider helpful. These are the main data sources for related companies to gain insights into the markets in which they participate, the timing of that participation, and the potential success of product design feature choices.

**2 Assumption**

*●****Assumption 1.*** *The online marketplace operates stably. And there were no situations such as an outbreak of an epidemic which would seriously affect the production chain of online shopping.*

●***Assumption 2.*** *The ratings and reviews depict customers’ real experience and feeling about their purchased products. The sentiment in the review text reflects one’s feelings on the products.*

●***Assumption 3.*** *The vast majority of individual differences of customers e.g., economic status and educational level, are ignored.*

●***Assumption 4.*** *It takes some time for shipping the product. Some customers would prefer making reviews sometime after receiving the purchased products.*

●***Assumption 5.*** *Consumers pay more attention to the negative comments e.g., low-star rating or negative reviews when purchasing the products.*

●***Assumption 6.****The influence of word order in semantic analysis is ignored, and the influence of word order on overall emotion is not considered when analyzing and commenting feelings. That is, we assume that "I love this hair Dryer "and" The hair Dryer Love Me "have the same emotional color.*

**3 Data acquisition**

We download three products’ data sets which are hair dryer,microwave and pacifier.

Each data set contains the following:

●marketplace（string）

●customer\_id（string）

●review\_id（string）

●product\_id（string）

●product\_parent（string）

●product\_title（string）

●product\_category（string）

●star\_rating（int）

●helpful\_votes（int）

●total\_votes（int）

●vine（string）

●verified\_purchase（string）

●review\_headline（string）

●review\_body（string）

●review\_date（bigint）

**4 Data Clean**

We need to:

●Delete reviewers didn’t buy the products actually.

●Delete the column of marketplace and product\_category.

●Rearrange index.

Then we will get three new data set after cleaning.

Essential Code：



### **4 Naive Bayes Model**

### In order to extract information about product from user reviews, we need to use Natural Language Processing (NLP) to extract emotionally relevant words in the reviews, and judge the user's emotional tendencies from these words, and further quantify the content of the reviews. We will use this quantified value as a measure of user satisfaction.

First of all, we want to get the goal that we can predict the positive or negative probability of the emotion of a review through a word in the review, so as to determine whether the review is positive or negative. In the field of Natural Language Processing (NLP), Naive Bayes is usually used to Text-Categorization, and it is direct and efficient when dealing with problems. Therefore, we use Naive Bayes to classify user reviews.

When the Naive Bayes method classifies words into emotions, the learned model is used to calculate the posterior probability distribution of the input word *x*, and the class with the greatest posterior probability is used as the class output of the input word *x*.  
 Suppose *x ∈ X*, *X* is a collection of review words, *ck ∈ Y*, *Y* = *{Pos,Neg}* represents both positive and negative comments.*Y* = 1 represents this review as a positive comment, while *Y* = 0 represents this review as a negative review.Then we need to get its posterior probability *P*(*Y* = *ck|X* = *x*),

According to Bayes formula,we have:

*P*(*Y* = *ck|X* = *x*) = *P*(*X* = *x|Y* = *ck*)*P*(*Y* = *ck*) /(∑*k P*(*X* = *x|Y* = *ck*)*P*(*Y* = *ck*) )

Because

∑*k P*(*X* = *x|Y* = *ck*)*P*(*Y* = *ck*) =

*P*(*X* = *x|Y* = *Pos*)*P*(*Y* = *Pos*)+*P*(*X* = *x|Y* = *Neg*)*P*(*Y* = *Neg*)

is a fixed value,the problem we solved becomes the following optimization problem:

*y* = *argmaxckP*(*Y* = *ck*)*P*(*Y* = *ck*)*P*(*X* = *x|Y* = *ck*)

Among them, *y ∈ Y* is the category with the greatest posterior probability of *x*,We output this category as the category of the word. We find a labeled corpus of Amazon reviews on the kaggle website as a training set to train our Bayesian classifier, and then use it to analyze the sentiment tendency of reviews.

**5 Other Reference codes**

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

#VADER is freely available athttps://github.com/cjhutto/vaderSentiment

import pandas as pd

data = pd.read\_csv(hair\_dryer.csv,sep='\t')[['star\_rating','review\_headline','review\_body']]

#VADER Model

def transform(df):

    d=df.values.tolist()

    for element in d:

        vs1 = analyzer.polarity\_scores(str(element[1]))

        l1=list(vs1.values())

        vs2 = analyzer.polarity\_scores(str(element[2]))

        l2=list(vs2.values())

        element.append(l1,l2)

    return pd.DataFrame(d)

data=transform(data)

data.columns=['star','head','body','h\_sn','h\_wn','h\_m','h\_wp','h\_sp','b\_sn','b\_wn','b\_m''b\_wp','b\_sp']

data.to\_csv(outputurl)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

# synthesis score regression

def regression\_s(microwave):

    data = pd.read\_csv('total\_score/' + microwave + ".csv")

    data\_x = np.array(data['review\_date'])

    data\_y = np.array(data['total\_score'])

    x\_train, test\_x, y\_train, test\_y = train\_test\_split(data\_x, data\_y, test\_size=0.33, random\_state=1)

    x = x\_train.reshape(-1, 1)

    model = LinearRegression()

    model.fit(x, y\_train)

    test\_x = test\_x.reshape(-1, 1)

    weights = np.polyfit(x\_train, y\_train, 4)

    ans = np.poly1d(weights)

    print("y=", ans)

    model = np.poly1d(weights)

    xp = np.linspace(test\_x.min(), test\_x.max(), 70)

    pred\_plot = model(xp)

    pred = model(test\_x)

    plt.scatter(test\_x, test\_y, facecolor='blue', edgecolor='blue')

    plt.plot(xp, pred\_plot, color='red')

    plt.title("synthesis score regression")

    plt.ylabel(product\_name + ' synthesis score')

    plt.savefig('total\_score/' + product\_name + '.png', dpi=200, bbox\_inches='tight')

    plt.show()

    MAE = mean\_squared\_error(test\_y, pred.flatten())

    MSE = mean\_absolute\_error(test\_y, pred.flatten())

    print("MSE", MSE)

    print("MAE", MAE)

if \_\_name\_\_ == '\_\_main\_\_':

    regression\_s("hair\_dryer")

'''

pip install PyDrive

pip install gensim

pip install pyldavis

python - m spacy download en'''

import gzip

import json

import os

import re

from itertools import chain

from string import punctuation

import gensim

import jieba

import matplotlib .pyplot as plt

import nltk

import numpy as np

import pandas as pd

from gensim import corpora

from google.colab import auth , drive

from nltk import FreqDist , ngrams

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from sklearn import svm

from sklearn.dummy import DummyClassifier

from sklearn. feature\_extraction .text import CountVectorizer , TfidfVectorizer

from sklearn. linear\_model import LogisticRegression

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

import pyLDAvis

import pyLDAvis.gensim

import seaborn as sns

import spacy

from oauth2client .client import GoogleCredentials

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive

from wordcloud import WordCloud

drive.mount('/content/drive')

nltk.download('stopwords')

pd. set\_option ("display. max\_colwidth ", 200)

%matplotlib inline

hair\_dryer = pd.read\_csv('/content/drive/My Drive/data/ new\_hair\_dryer .csv')

microwave = pd.read\_csv('/content/drive/My Drive/data/ new\_microwave .csv')

pacifier = pd. read\_csv('/content/drive/My Drive/data/ new\_pacifier .csv')

hair\_dryer1 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / hair\_dryer /rating1.csv')

hair\_dryer2 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / hair\_dryer /rating2.csv')

hair\_dryer3 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / hair\_dryer /rating3.csv')

hair\_dryer4 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / hair\_dryer /rating4.csv')

hair\_dryer5 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / hair\_dryer /rating5.csv')

microwave1 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / microwave /rating1.csv')

microwave2 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / microwave /rating2.csv')

microwave3 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / microwave /rating3.csv')

microwave4 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / microwave /rating4.csv')

microwave5 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating / microwave /rating5.csv')

pacifier1 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating /pacifier/rating1.csv')

pacifier2 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating /pacifier/rating2.csv')

pacifier3 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating /pacifier/rating3.csv')

pacifier4 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating /pacifier/rating4.csv')

pacifier5 = pd.read\_csv('/content/drive/My Drive/ classify\_byrating /pacifier/rating5.csv')

add\_punc =' {}() %^ >.^ -=&#@'

add\_punc = add\_punc+ punctuation

h\_head1 = hair\_dryer1 . review\_headline .tolist ()

m\_head1 = microwave1 . review\_headline .tolist ()

p\_head1 = pacifier1 . review\_headline .astype(str).tolist ()

h\_head2 = hair\_dryer2 . review\_headline .tolist ()

m\_head2 = microwave2 . review\_headline .tolist ()

p\_head2 = pacifier2 . review\_headline .astype(str).tolist ()

h\_head3 = hair\_dryer3 . review\_headline .tolist ()

m\_head3 = microwave3 . review\_headline .tolist ()

p\_head3 = pacifier3 . review\_headline .astype(str).tolist ()

h\_head4 = hair\_dryer4 . review\_headline .tolist ()

m\_head4 = microwave4 . review\_headline .tolist ()

p\_head4 = pacifier4 . review\_headline .astype(str).tolist ()

h\_head5 = hair\_dryer5 . review\_headline .tolist ()

m\_head5 = microwave5 . review\_headline .tolist ()

p\_head5 = pacifier5 . review\_headline .astype(str).tolist ()

h\_body1 = hair\_dryer1 . review\_body .tolist ()

m\_body1 = microwave1 . review\_body .tolist ()

p\_body1 = pacifier1 . review\_body .astype(str).tolist ()

 h\_body2 = hair\_dryer2 . review\_body .tolist ()

m\_body2 = microwave2 . review\_body .tolist ()

p\_body2 = pacifier2 . review\_body .astype(str).tolist ()

h\_body3 = hair\_dryer3 . review\_body .tolist ()

m\_body3 = microwave3 . review\_body .tolist ()

p\_body3 = pacifier3 . review\_body .astype(str).tolist ()

h\_body4 = hair\_dryer4 . review\_body .tolist ()

m\_body4 = microwave4 . review\_body .tolist ()

p\_body4 = pacifier4 . review\_body .astype(str).tolist ()

h\_body5 = hair\_dryer5 . review\_body .tolist ()

m\_body5 = microwave5 . review\_body .tolist ()

p\_body5 = pacifier5 . review\_body .astype(str).tolist ()

def freq\_words (x, filepath , terms =30):

all\_words ='.join ([ text for text in x])

all\_words = all\_words .split ()

fdist = FreqDist( all\_words)

words\_df = pd. DataFrame(

{'word': list(fdist.keys ()),'count': list(fdist.values ())})

# selecting top 20 most frequent words

d = words\_df.nlargest(columns="count", n=terms)

plt.figure(figsize =(20 , 5))

ax = sns.barplot(data=d, x="word", y="count")

ax.set(ylabel='Count')

 plt.show ()

plt.savefig(filepath)

hair\_dryer ['review\_body'] = hair\_dryer ['review\_body'].str.replace("n\'t", " not")

 # remove unwanted characters , numbers and symbols

hair\_dryer ['review\_body'] = hair\_dryer ['review\_body'].str.replace([^a-zA -Z#]", " ")

stop\_words = stopwords .words('english')

# function to remove stopwords

def remove\_stopwords (rev):

rev\_new = " ".join ([i for i in rev if i not in stop\_words ])

return rev\_new

# remove short words (length < 3)

hair\_dryer ['review\_body'] = hair\_dryer ['review\_body'].apply(

lambda x:'.join ([w for w in x.split () if len(w) > 2]))

# remove stopwords from the text

reviews = [ remove\_stopwords (r.split ()) for r in hair\_dryer ['review\_body']]

# make entire text lowercase

reviews = [r.lower () for r in reviews]

nlp = spacy.load('en', disable =['parser','ner'])

def lemmatization (texts , tags =['NOUN','ADJ']):

    output = []

    for sent in texts:

    doc = nlp(" ".join(sent))

    output.append ([ token.lemma\_ for token in doc if token.pos\_ in tags ])

    return output

tokenized\_reviews = pd.Series(reviews).apply(lambda x: x.split ())

reviews\_2 = lemmatization ( tokenized\_reviews )

reviews\_3 = []

for i in range(len( reviews\_2 )):

    reviews\_3 .append(''.join( reviews\_2 [i]))

freq\_words (reviews\_3 ,'/content/drive/My Drive/ classify\_byrating / main\_word\_h', 35)

dictionary = corpora. Dictionary ( reviews\_2)

doc\_term\_matrix = [ dictionary .doc2bow(rev) for rev in reviews\_2]

LDA = gensim.models.ldamodel.LdaModel

lda\_model = LDA(corpus=doc\_term\_matrix ,

id2word=dictionary ,

num\_topics =7,

random\_state =100 ,

chunksize =1000 ,

passes =50)

lda\_model . print\_topics ()

 # Visualize the topics

pyLDAvis. enable\_notebook ()

vis = pyLDAvis.gensim.prepare(lda\_model , doc\_term\_matrix , dictionary )

vis