

INTRODUCTION

1.1 Overview

Mobile phones have revolutionized the way we purchase products online, making all the information available at our fingertips. Reviews and ratings submitted by consumers became an integral part of the customer's buying decision process. The review and rating platform provided by eCommerce players creates a transparent system for consumers to take decisions and feel confident about it.

However, it is difficult to read all the feedback for a particular item especially for the popular items with many comments. In this project, we will attempt to understand the factors that contribute to classifying reviews as positive or negative

1.2 Purpose

The main objective of this project is to understand the factors that contribute to classifying reviews as positive or negative. Natural language processing is used to analyze the sentiment (positive or a negative) of the given review. A sample web application is integrated to the model built

key objectives are:

- Know fundamental concepts and techniques of natural language processing (NLP).
- Gain a broad understanding of text data.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- know how to build a web application using Flask framework.

2. LITERATURE SURVEY

Existing Solution:

A Comparison of Sentiment Analysis Methods on Amazon Reviews of Mobile Phones

Sara Ashour Aljuhani¹, Norah Saleh Alghamdi² School of Computing, Dublin City University (DCU)¹Dublin, Ireland Department of Computer Science^{1,2} Princess Nourah bint Abdulrahman University (PNU) Riyadh, KSA

Existing approaches to solve the problem:

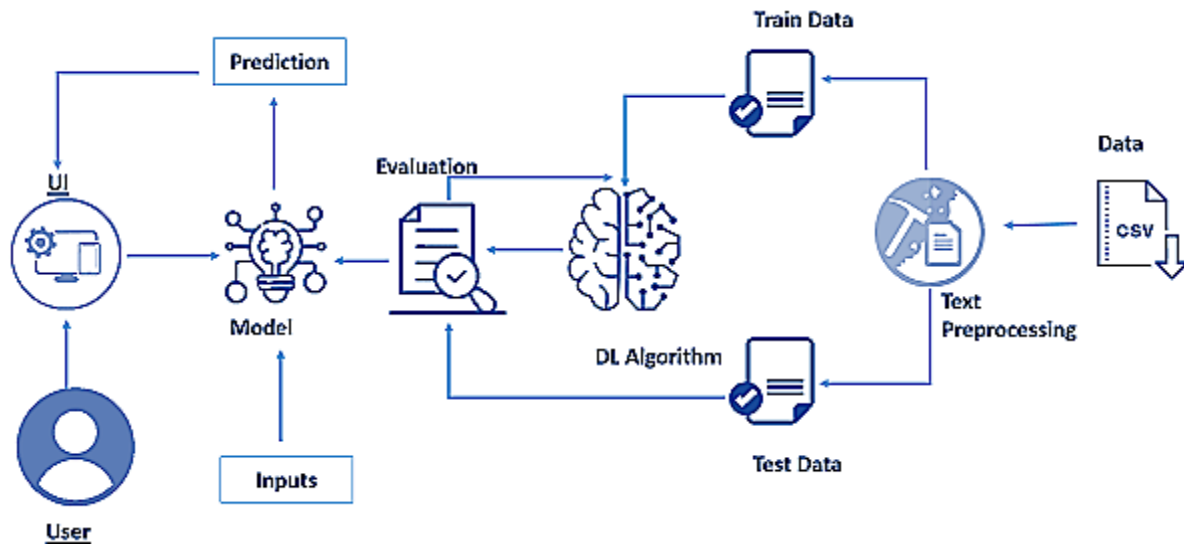
In this research they have studied sentiment analysis of mobile phone reviews using different types of machine learning classifiers, such as Logistic Regression (LR), Naive Bayes (NB), Stochastic Gradient Decent (SGD) and deep learning algorithms such as Convolutional Neural Networks (CNN). These algorithms are applied using different feature extraction approaches. For example, Bag-of-words with (Bigram, Trigram), TF-IDF with (Unigram, Bigram, Trigram), word2vec, word2vec with Bigram, and glove. We evaluated them with different classification methods such as bag-of-words revealing that when the size of 'n' in n-gram increases, the accuracy will also increase, and Log loss value will decrease. On the other hand, our Bigram approach provided best results with TF-IDF in unbalanced data, and Trigram in balanced data.

3.Propose Solution:

The proposed solution is to understand the factors that contribute to classifying reviews as positive or negative. Natural language processing is used to analyze the sentiment (positive or a negative) of the given review.

4.THEORETICAL ANALYSIS

Block diagram



Hardware Requirements

- minimum of 8GB RAM
- Intel Core i7 processor) is recommended
- Windows 10 or above
- NVIDIA GPU is preferable

Software Requirements

- Python
- Python Web Frame Works NLP

5.FLOWCHART



6. RESULT

Text Preprocessing

The screenshot shows the Spyder Python IDE interface. The main window displays a list of text reviews, with the following columns: Index, Type, Size, and Value. The reviews are sorted by index, and the first 16 reviews are visible. The right sidebar shows a summary of the corpus statistics, including the total number of elements (67986) and the number of unique words (67985).

Index	Type	Size	Value
0	str	1	def best worst samsung awhile absolute doo doo read review detect rage ...
1	str	1	text messaging work due software issue nokia sprint phone text messagi ...
2	str	1	love phone great reliable phone also purchased phone samsung died menu ...
3	str	1	love phone love phone really need one expect price bill received one a ...
4	str	1	great phone service option lousy case phone great every purpose offer ...
5	str	1	worked great hello phone used decided buy flip phone problem battery n ...
6	str	1	wanna cool nokia cool cheap color word describe nokia perfectly mean w ...
7	str	1	problem universal headset overall nice phone except nokia made univers ...
8	str	1	cool phone never owned nokia phone first really like phone alot recept ...
9	str	1	pissed little bit ok well in school need text messaging ive phone mont ...
10	str	1	work great dropt phone year really like never partial flip phone appre ...
11	str	1	slow annoying fragile heavy bulky slow want check missed call list pho ...
12	str	1	worth paying something else bought phone year ago using ever since spr ...
13	str	1	great free phone sprint customer excellent choice sprint use enrolling ...
14	str	1	stupid phone buy service
15	str	1	exellent service nextel nearly year started time last year motorola up ...

The right sidebar shows the following statistics:

- Size: 67986
- Value: [3, 1, ...]
- Count: 1
- Column: [1, 0, ...]
- Column: (67986, 2)
- Column: (67986, 4)
- Column: (67986, 1)
- 67985
- Porter

Variable explorer

Name	Type	Size	Value
a	list	67986	[3, 1, 5, 3, 4, 4, 5, 4, 5, 3, ...]
corpus	list	67986	['def best worst samsung awhile absolute doo doo read review detect ra ...]
cv	feature_extraction.text.CountVectorizer	1	CountVectorizer object of sklearn.feature_extraction.text module
d	list	67986	[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, ...]
data1	DataFrame	(67986, 2)	Column names: emotions, Review
dataset	DataFrame	(67986, 4)	Column names: rating, verified, title, body
dt	DataFrame	(67986, 1)	column names: emotions
i	int	1	67985
ps	stem.porter.PorterStemmer	1	PorterStemmer object of nltk.stem.porter module
stopwords	corpus.reader.wordlist.WordListCorpusReader	1	WordListCorpusReader object of nltk.corpus.reader.wordlist module
temp	str	1	outstanding phone price love size style phone great size phone many gr ...
wordnet	stem.wordnet.WordNetLemmatizer	1	WordNetLemmatizer object of nltk.stem.wordnet module
x	Array of int64	(67986, 2000)	[[0 0 1 ... 0 0 0] [0 0 0 ... 0 0 0]
x_test	Array of int64	(13598, 2000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
x_train	Array of int64	(54388, 2000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
y	Array of int64	(67986,)	[1 0 1 ... 1 1 1]
y_test	Array of int64	(13598,)	[1 1 1 ... 1 1 1]
y_train	Array of int64	(54388,)	[1 0 1 ... 1 1 1]

Model building

IPython console

```

In [48]: runcell(['37'], 'C:/Users/jayas/PycharmProjects/AmazonCellPhoneReviewsProject/AmazonCellPhoneReviewSpyder.py')
Train on 54388 samples
Epoch 1/50
54388/54388 [=====] - 44s 817us/sample - loss: 0.2388 - accuracy: 0.9014 15s - loss: 0.2525 - accuracy: 0.8954
Epoch 2/50
54388/54388 [=====] - 41s 749us/sample - loss: 0.1428 - accuracy: 0.9424
Epoch 3/50
54388/54388 [=====] - 39s 720us/sample - loss: 0.0664 - accuracy: 0.9752
Epoch 4/50
54388/54388 [=====] - 39s 714us/sample - loss: 0.0309 - accuracy: 0.9892
Epoch 5/50
54388/54388 [=====] - 39s 721us/sample - loss: 0.0206 - accuracy: 0.9929
Epoch 6/50
54388/54388 [=====] - 40s 744us/sample - loss: 0.0134 - accuracy: 0.9956
Epoch 7/50
54388/54388 [=====] - 43s 788us/sample - loss: 0.0122 - accuracy: 0.9958
Epoch 8/50
54388/54388 [=====] - 39s 719us/sample - loss: 0.0113 - accuracy: 0.9962
Epoch 9/50
54388/54388 [=====] - 42s 772us/sample - loss: 0.0122 - accuracy: 0.9962
Epoch 10/50
54388/54388 [=====] - 39s 723us/sample - loss: 0.0100 - accuracy: 0.9966
Epoch 11/50
54388/54388 [=====] - 39s 718us/sample - loss: 0.0090 - accuracy: 0.9972
Epoch 12/50
54388/54388 [=====] - 39s 718us/sample - loss: 0.0090 - accuracy: 0.9973
Epoch 13/50
54388/54388 [=====] - 39s 721us/sample - loss: 0.0076 - accuracy: 0.9975
Epoch 14/50
54388/54388 [=====] - 39s 718us/sample - loss: 0.0070 - accuracy: 0.9977
Epoch 15/50
54388/54388 [=====] - 39s 718us/sample - loss: 0.0078 - accuracy: 0.9977
Epoch 16/50
54388/54388 [=====] - 38s 705us/sample - loss: 0.0078 - accuracy: 0.9977
Epoch 17/50
54388/54388 [=====] - 39s 722us/sample - loss: 0.0055 - accuracy: 0.9983
Epoch 18/50

```

```
IPython console
Console 1/A x
Epoch 18/50
54388/54388 [=====] - 48s 885us/sample - loss: 0.0057 - accuracy: 0.9981
Epoch 19/50
54388/54388 [=====] - 45s 834us/sample - loss: 0.0046 - accuracy: 0.9985
Epoch 20/50
54388/54388 [=====] - 49s 897us/sample - loss: 0.0050 - accuracy: 0.9983
Epoch 21/50
54388/54388 [=====] - 42s 768us/sample - loss: 0.0077 - accuracy: 0.9978
Epoch 22/50
54388/54388 [=====] - 40s 741us/sample - loss: 0.0069 - accuracy: 0.9980
Epoch 23/50
54388/54388 [=====] - 40s 741us/sample - loss: 0.0097 - accuracy: 0.9972TA: 34s - loss: 0.0121 - accuracy: 0.9973
Epoch 24/50
54388/54388 [=====] - 42s 766us/sample - loss: 0.0083 - accuracy: 0.9979
Epoch 25/50
54388/54388 [=====] - 41s 753us/sample - loss: 0.0085 - accuracy: 0.9979
Epoch 26/50
54388/54388 [=====] - 40s 729us/sample - loss: 0.0059 - accuracy: 0.9981
Epoch 27/50
54388/54388 [=====] - 39s 717us/sample - loss: 0.0067 - accuracy: 0.9980
Epoch 28/50
54388/54388 [=====] - 39s 717us/sample - loss: 0.0056 - accuracy: 0.9984
Epoch 29/50
54388/54388 [=====] - 39s 717us/sample - loss: 0.0053 - accuracy: 0.9985
Epoch 30/50
54388/54388 [=====] - 41s 752us/sample - loss: 0.0049 - accuracy: 0.9985
Epoch 31/50
54388/54388 [=====] - 45s 833us/sample - loss: 0.0038 - accuracy: 0.9986TA: 28s - loss: 0.0042 - accuracy: 0.9983
Epoch 32/50
54388/54388 [=====] - 51s 932us/sample - loss: 0.0054 - accuracy: 0.9983
Epoch 33/50
54388/54388 [=====] - 40s 731us/sample - loss: 0.0050 - accuracy: 0.9985
Epoch 34/50
54388/54388 [=====] - 42s 774us/sample - loss: 0.0045 - accuracy: 0.9985
Epoch 35/50
54388/54388 [=====] - 50s 915us/sample - loss: 0.0038 - accuracy: 0.9988
Epoch 36/50
54388/54388 [=====] - 44s 800us/sample - loss: 0.0033 - accuracy: 0.9988
Epoch 37/50
54388/54388 [=====] - 40s 736us/sample - loss: 0.0029 - accuracy: 0.9989
Epoch 38/50
54388/54388 [=====] - 40s 733us/sample - loss: 0.0031 - accuracy: 0.9989
Epoch 39/50
54388/54388 [=====] - 40s 740us/sample - loss: 0.0039 - accuracy: 0.9987
Epoch 40/50
54388/54388 [=====] - 38s 699us/sample - loss: 0.0107 - accuracy: 0.9972
Epoch 41/50
54388/54388 [=====] - 40s 735us/sample - loss: 0.0106 - accuracy: 0.9974
Epoch 42/50
54388/54388 [=====] - 42s 763us/sample - loss: 0.0068 - accuracy: 0.9981
Epoch 43/50
54388/54388 [=====] - 40s 729us/sample - loss: 0.0062 - accuracy: 0.9981
Epoch 44/50
54388/54388 [=====] - 38s 699us/sample - loss: 0.0044 - accuracy: 0.9986
Epoch 45/50
54388/54388 [=====] - 38s 701us/sample - loss: 0.0042 - accuracy: 0.9988
Epoch 46/50
54388/54388 [=====] - 38s 705us/sample - loss: 0.0034 - accuracy: 0.9988
Epoch 47/50
54388/54388 [=====] - 40s 732us/sample - loss: 0.0037 - accuracy: 0.9986
Epoch 48/50
54388/54388 [=====] - 38s 705us/sample - loss: 0.0045 - accuracy: 0.9986
Epoch 49/50
54388/54388 [=====] - 39s 713us/sample - loss: 0.0038 - accuracy: 0.9988
Epoch 50/50
54388/54388 [=====] - 38s 707us/sample - loss: 0.0081 - accuracy: 0.9980
In [49]:
```

Testing

Variable explorer

Name	Type	Size	Value
corpus	list	67986	['def best worst samsung awhile absolute doo doo read review detect ra ...
cv	feature_extraction.text.CountVectorizer	1	CountVectorizer object of sklearn.feature_extraction.text module
d	list	67986	[1, 0, 1, 1, 1, 1, 1, 1, 1, ...]
data1	DataFrame	(67986, 2)	column names: emotions, Review
dataset	DataFrame	(67986, 4)	column names: rating, verified, title, body
dt	DataFrame	(67986, 1)	Column names: emotions
i	int	1	67985
ps	stem.porter.PorterStemmer	1	PorterStemmer object of nltk.stem.porter module
stopwords	corpus.reader.wordlist.WordListCorpusReader	1	WordListCorpusReader object of nltk.corpus.reader.wordlist module
temp	str	1	outstanding phone price love size style phone great size phone many gr ...
text	str	1	great design smooth user experience
wordnet	stem.wordnet.WordNetLemmatizer	1	WordNetLemmatizer object of nltk.stem.wordnet module
x	Array of int64	(67986, 2000)	[[0 0 1 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
x_test	Array of int64	(13598, 2000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
x_train	Array of int64	(54388, 2000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
y	Array of int64	(67986,)	[1 0 1 ... 1 1 1]
y_p	Array of float32	(1, 1)	[[1.]]
y_pred	Array of float32	(13598, 1)	[[9.8354441e-01] [1.0000000e+00]
y_test	Array of int64	(13598,)	[1 1 1 ... 1 1 1]
y_train	Array of int64	(54388,)	[1 0 1 ... 1 1 1]

Type here to search

12:38 03/11/2021

App

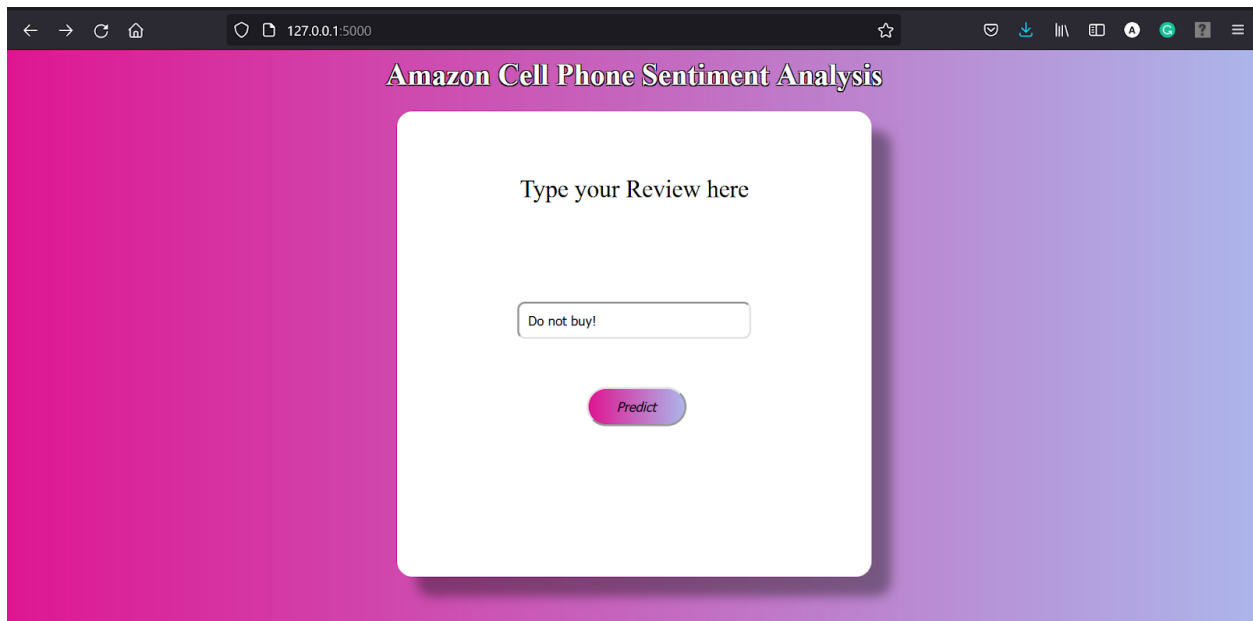
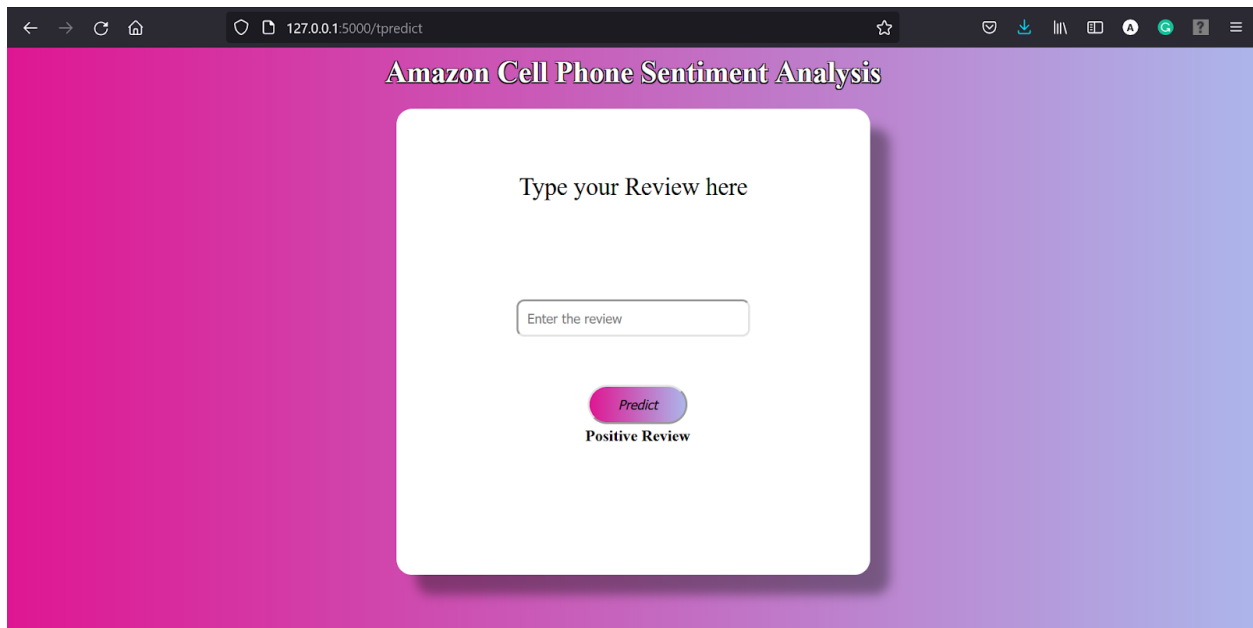
← → ↺ 🏠 127.0.0.1:5000 ☆ 🛡️ ⬇️ 📄 🗑️ 🗂️ ? ☰

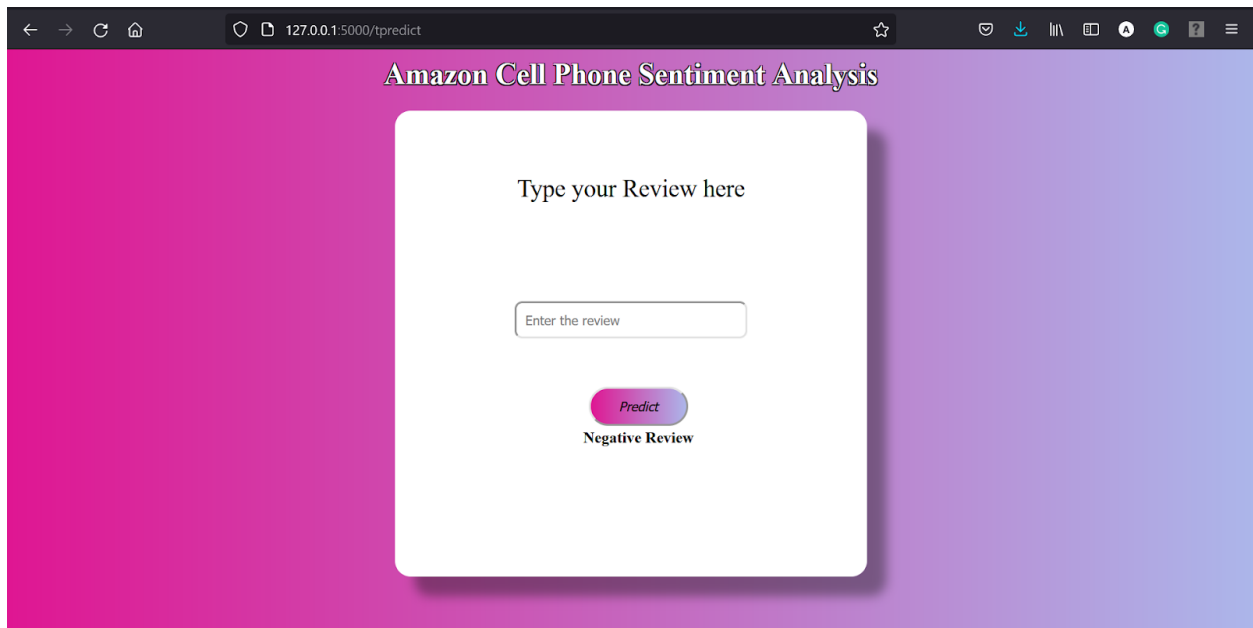
Amazon Cell Phone Sentiment Analysis

Type your Review here

Great user design and experience!Wk

Predict





IBM Watson Studio	All	Search	Buy	Anushka R's Account	AR
Projects					🔍 ⌚ 💬 👤 ?
🔍 Which project are you looking for?			All my projects	New project +	
Name	Role	Storage	Collaborators	Creator	Date created
Analysis of Amazon Cell Phone Reviews using ibm	Admin	COS	AR	Anushka R	Nov 26, 2021
Show more					

Devspace1Amazon

OverviewAssetsDeploymentsJobsManage

General

- Access control
- Environments

Name
Devspace1Amazon

Description
No description provided.

Space GUID
951f2357-ae42-4855-ad40-a6a7f652...

Date created
Dec 15, 2021 2:15 PM
by Anushka R

Last updated
Dec 15, 2021 2:16 PM

Deployment space tags
No tags are set to this space.

Storage used
68.57 MB used

Name
CloudObjectStorage

Bucket
489c9a2e-1d40-4660-be01-b270be7b04a7

Machine learning service

Machine Learning-jl

OverviewAssetsEnvironmentsJobsAccess controlSettings

What assets are you looking for?

Data assets

New Data asset

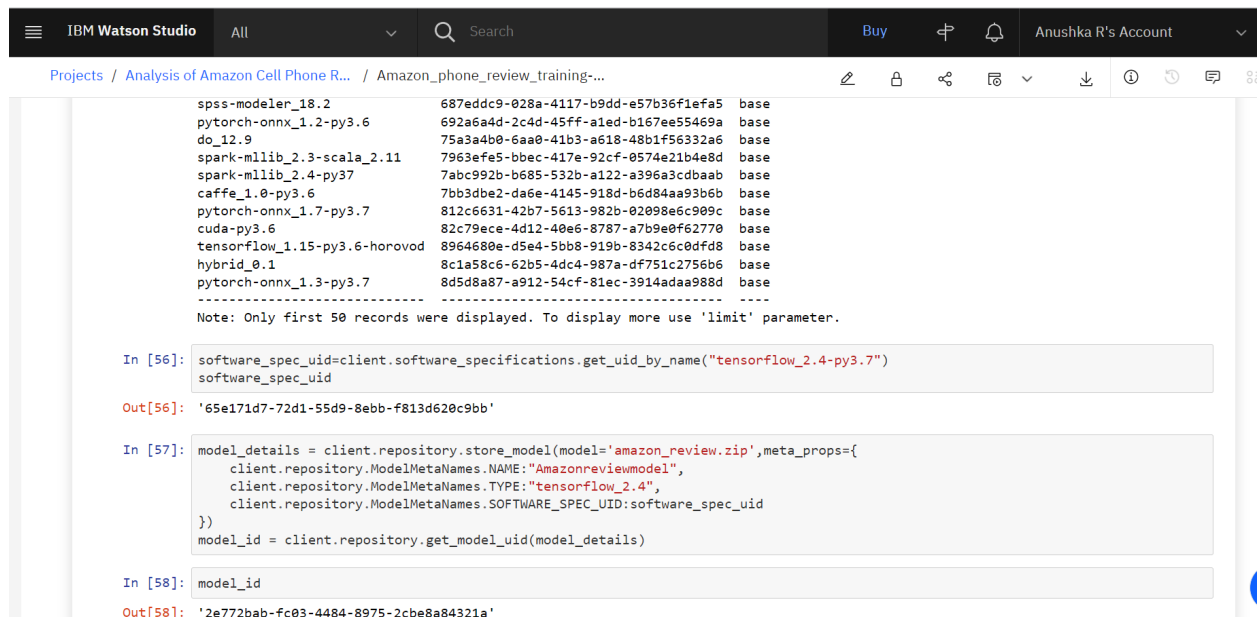
0 assets selected.

<input type="checkbox"/>	Name	Type	Created by	Last modified
<input type="checkbox"/>	CSV Reviews.csv	Data Asset	Anushka R	Dec 15, 2021, 02:07 PM

Notebooks

New Notebook

Name	Shared	Scheduled	Status	Language	Last editor	Last modified
<input type="checkbox"/> Amazon_phone_review_training-ibm				Python 3.8	Anushka R	Dec 15, 2021



The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', a search bar, and user account information 'Anushka R's Account'. Below the navigation bar, the breadcrumb path is 'Projects / Analysis of Amazon Cell Phone R... / Amazon_phone_review_training-...'. The main area displays a Jupyter Notebook with the following content:

```
spss-modeler_18.2      687eddc9-028a-4117-b9dd-e57b36f1efa5  base
pytorch-onnx_1.2-py3.6 692a6a4d-2c4d-45ff-a1ed-b167ee55469a  base
do_12.9                75a3a4b0-6aa0-41b3-a618-48b1f56332a6  base
spark-mllib_2.3-scala_2.11 7963efe5-bbec-417e-92cf-0574e21b4e8d  base
spark-mllib_2.4-py37      7abc992b-b685-532b-a122-a396a3cdbaab  base
caffe_1.0-py3.6          7bb3dbe2-da6e-4145-918d-b6d84aa93b6b  base
pytorch-onnx_1.7-py3.7    812c6631-42b7-5613-982b-02098e6c909c  base
cuda-py3.6              82c79ece-4d12-40e6-8787-a7b9e0f62770  base
tensorflow_1.15-py3.6-horovod 8964680e-d5e4-5bb8-919b-8342c6c0dfd8  base
hybrid_0.1              8c1a58c6-62b5-4dc4-987a-df751c2756b6  base
pytorch-onnx_1.3-py3.7    8d5d8a87-a912-54cf-81ec-3914adaa988d  base
-----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

Below the table, there are three code blocks:

```
In [56]: software_spec_uid=client.software_specifications.get_uid_by_name("tensorflow_2.4-py3.7")
software_spec_uid

Out[56]: '65e171d7-72d1-55d9-8ebb-f813d620c9bb'
```

```
In [57]: model_details = client.repository.store_model(model='amazon_review.zip',meta_props={
        client.repository.ModelMetaNames.NAME:"Amazonreviewmodel",
        client.repository.ModelMetaNames.TYPE:"tensorflow_2.4",
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
    })
model_id = client.repository.get_model_uid(model_details)

In [58]: model_id

Out[58]: '2e772bab-fc03-4484-8975-2cbe8a84321a'
```

7. ADVANTAGES

The proposed system does not restrict its scope over polarity or sentiment prediction, it digs into the polarity intensity such as whether the review is just positive or very positive and same for negative reviews. The proposed system results also show that the different aspects of the product about which positive is being said or negative is being said are also identified. The efficiency of the model is based on performance evaluation; it is seen that it outperforms Machine Learning model performance parameters.

DISADVANTAGES

Some of the limitations are low performance and highly time consuming. The proposed model fails to give accurate results sometimes and hence is inefficient.

8 .APPLICATIONS

- Social media monitoring
- Customer support

- Customer feedback
- Brand monitoring and reputation management
- Voice of customer (VoC)
- Voice of employee
- Product analysis
- Market and competitor research

9. CONCLUSION

Product review platform, provided by Amazon, describes that a major number of reviewers have set 4-star and 3-star ratings to the unlocked phones. The average length of the reviews comes close to 230 characters. It can be seen that review with more lengthy text tends to be more useful and there is a direct correlation between rating and price. Sentiment analysis shows that positive sentiment is established among the reviews and in terms of emotions, 'trust', 'anticipation' and 'joy' have highest scores. Hence it is observed that the proposed model gives more accurate and elaborative details about the reviews which is helpful in terms of analyzing the aspects of the products whose polarity is also identified. So because of this, consumers and service providers will get better clarification on products' market value and help them make important business decisions.

10. FUTURE SCOPE

Some algorithms that remain to be applied in future work include LSTM, KNN, and Maximum entropy. Then, we will compare the result to the result we performed in this current study. Our research has some limitations: NLP is relatively a new topic, and highly advanced; hence, it needs a lot of research to understand the field and how it works. Furthermore, we faced some problems with computer memory causing experiments to be highly time consuming.

APPENDIX

1. source code

index.html

```
<!DOCTYPE html>
<html>
<title >Amazon Cellphone Sentimental Review Analysis</title>
<head>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFIdvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
  <link rel="stylesheet" type="text/css" href="[url_for("static", filename="style/index12.css")] }}">

  <style>

  body {

    /* background: #acb6e5;

    background: -webkit-linear-gradient (to right, #86fde8, #acb6e5);
    background: linear-gradient(to right, #86fde8, #acb6e5); */
    background: rgb(172,182,229);
background: linear-gradient(90deg, rgba(172,182,229,1) 35%, rgba(134,253,232,1) 100%);
  }
</style>

</head>

<body class="body1">

<form method="POST">

<div class="container-fluid">

<h1 style="color:black; font-size:50px;text-align:center" >Amazon Cellphone Sentiment Analysis</h1>

<p align="center" style="font-size:25px">Enter your Review here</p>
```

```

<form action="/tpredict" method="POST">
  <p align="center">
    <input type="text" align="Center" placeholder="Enter the review" name="tweet" id="rcorners1"
required>
  </p>
  <table align='center' style='padding:0.5px; border-spacing: 10px; font-size:20 px'>
    <tr>
      <td colspan="2" align="center"><button class="btn btn-dark" type="submit" nane="predict"
align="center">PREDICT</button>
    </tr>
    <tr>
      <td colspan="2" align="center"><b>{{ypred}}</b></td>
    </tr>
  </table>
</form>
<p>
{% if ypred == "Positive Review" %}
<p>Positive</p>

{% else %}

{% if ypred == "Negative Review" %}
<p>Negative</p>

{% endif %}
{% endif %}
</p>
</div>
</form>
</body>
</html>

```

Temp.py

```

import re
import nltk
import pandas as pd
import numpy as np

dataset=pd.read_csv("20191226-reviews.csv")
print(dataset.head())
print(dataset.isnull().sum())
dataset['body']=dataset ['body'].fillna('').apply(str)

dataset['name'] = dataset ['name'].fillna('').apply(str)
dataset['title'] = dataset ['title'].fillna('').apply(str)
dataset['helpfulVotes'] = dataset [ 'helpfulVotes' ].fillna('').apply(str)
print(dataset.isnull().sum())
dataset=dataset.drop(columns=['asin','name','helpfulVotes','date'],axis=1)
a=dataset['rating'].tolist()
d=[]
for i in range(len(a)):
    if a[i]>=3:
        d.append(1)
    else:
        d.append(0)
print(d)
dt=pd.DataFrame(d,columns=['emotion'])
print(dt)
data1=pd.concat([dataset,dt],axis=1)
data1.head()
data1.drop(['verified'],axis=1,inplace=True)
data1['Review'] = data1['title'].str.cat(data1['body'],sep=" ")
data1.drop(['title','body','rating'],axis=1,inplace=True)
print(data1.head())
print(data1.shape)
y=data1.iloc[:,0].values
x=data1.iloc[:,1].values
print(y)

```

```

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer #create an object for stemming
ps=PorterStemmer()
#library used for stem the words
from nltk.stem import WordNetLemmatizer #create an object for wordnet Lemmatizer
wordnet=WordNetLemmatizer()
data=[]
for i in range(len(x)):
    review=data1['Review'][i]
    review=re.sub('[^a-zA-Z]', ' ',str(review))
    review=review.lower()
    review=review.split()
    review=[ps.stem(word) for word in review if not word in stopwords.words('english')]
    review=[wordnet.lemmatize(word) for word in review if not word in set(stopwords.words('english'))]
    review=' '.join(review)
    data.append(review)
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=2000)
x=cv.fit_transform(data).toarray()
print(x)
import pickle
pickle.dump(cv,open('count_vec.pkl','wb'))
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print(x_train.shape)
print(x_test.shape)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(units=13264,activation = 'relu'))
model.add(Dense(units= 2000,activation = 'relu'))
model.add(Dense(units= 2000,activation = 'relu'))
model.add(Dense(units= 2000,activation = 'relu'))
model.add(Dense(units=1,activation = 'sigmoid'))

```



```

model.compile(optimizer='adam',loss='binary_crossentropy', metrics = ['accuracy'])
model.fit(x_train,y_train,batch_size=128,epochs=50)
y_pred = model.predict(x_test)
text = "The phone is okay. average "
text = re.sub ('[^a-zA-Z]', ' ',text)
text = text.lower()
text = text.split()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
y_p = model.predict(cv.transform( [text]))
# saving the model
model.save("review_analysis.h5")

```

App.py

```

from flask import render_template, Flask, request,url_for
from tensorflow.keras.models import load_model
import pickle
import tensorflow as tf
graph =tf.compat.v1.get_default_graph()
with open(r'count_vec.pkl','rb') as file:
    cv=pickle.load(file)
app=Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/tpredict')

@app.route('/', methods=['GET', 'POST'])

def page2():
    if request.method == 'GET':

        return render_template('index.html')

    if request.method == 'POST' :

```

```
topic= request.form['tweet']
print("Hey " +topic)
topic=cv.transform([topic])
print("\n"+str(topic.shape)+"\n")
with graph.as_default():
    cla = load_model('review_analysis.h5')
    cla.compile(optimizer='adam',loss='binary_crossentropy')
    y_pred = cla.predict(topic)
    print("pred is "+str(y_pred))

if(y_pred > 0.7):
    topic="Positive Review"

else:
    topic = "Negative Review"
return render_template('index.html', ypred = topic)

if __name__=="__main__":
    app.run(debug=False)
```