Customer Recommendation based on Sentiment Analysis on Product Reviews

PROJECT REPORT

for

Natural Language Processing(CSE4022)

Submitted by

Kaustubh Jha 18BCE1043

Ayanabha jana 18BCE1044

To

DR. Subhra Rani Patra

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

November 2020

Table of Content

ABSTRACT

The problem of buying products online is that the consumer cannot touch, try, or even see the product directly. Then how does the consumer believe the product they like, is the worthy product to be bought. Moreover, the user would like a general recommendation on a product based on popularity and not just by the number of buyers.

The main key is product review from the consumer who has bought and tried the product.

Based on their reviews we will train our model categorizing each customer Review as either positive or negative through their natural language or features we have extracted.

Conversational and question-based recommender systems have gained increasing attention in recent years, with users enabled to converse with the system and better control recommendations.

Managing the negative reviews by checking similarity of other high rated reviews with the user-defined reviews in order to understand the type of problem being experienced by the user for a particular product and reply to them with a suitable solution or product. Here we have shown a recommendation system that learns from the data and problems given in review ,based on that it provides recommendations to users. Without the user specifically searching for that item, that item was brought automatically by the system.

INTRODUCTION

During online product search, customers are constantly looking for proof before interacting with any product or specific brand. Customer reviews boost the purchase transparency that can drive higher online trust on a product . Customer review can act as a third-party validation tool that can build user trust in your product and online promotions.

We are using sentiment analysis to analyse customer review,basically sentiment analysis is an algorithm-driven process that can categorize user feedback as positive or negative. Sentiment analysis algorithms have access to a large dictionary of words each of which has either a positive or negative sentiment attached to them.

Based on the included words and the associated sentiment in the user feedback, the sentiment analysis method assigns a sentiment score to them. As a result, positive feedback gets a higher sentiment score while negative feedback gathers a lower score.

While a sentiment can indicate either a positive or negative review , a group of similar words can help to analyze the actual words used to convey user sentiment. It can help in the understanding of the reason behind a particular response. For example, words that commonly convey positive feedback include “good,” “trustworthy,” “innovative” and “great” which can give customers more confidence to go for the product.

Algorithms used for sentiment analysis:

- **Naive Bayes:**

The Naive Bayes Classifier is a well-known machine learning classifier with applications in Natural Language Processing (NLP) and other areas. Despite its simplicity, it is able to achieve above average performance in different tasks like sentiment analysis.

- **Logistic Regression:**

For all data entries we will convert review.ratings to binary ,showing 1,0 as positive and negative review.

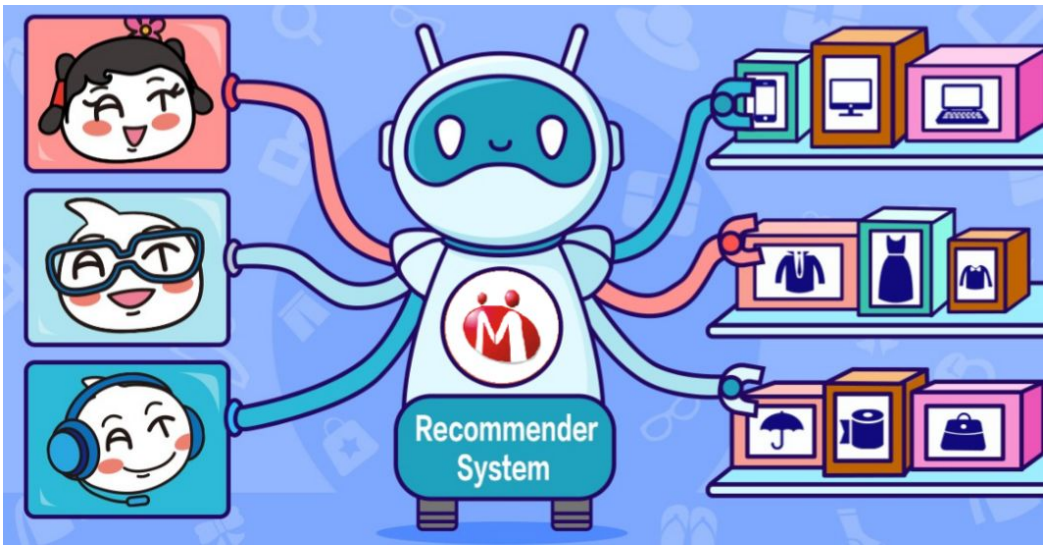
So the idea behind classification using logistic regression is minimizing the cost function which is a representation for the relation between the real output and the output predicted using sigmoid function.

The logistic or sigmoid function uses the features as input to calculate the probability of a review being labeled as positive, if the output is greater or equal 0.5, we classify the review as positive. Otherwise, we classify it as negative

Algorithm for Recommendation System:

- **K nearest neighbors:**

The nearest neighbors model computes the distance between every interaction vector in the query set against every interaction vector in the reference set. For each vector in the query set, the k closest reference vectors are returned. We evaluated two dissimilarity metrics, Euclidean distance and cosine dissimilarity. In our approach, we average the interaction vectors of the k neighbors to get an interaction score between 0 and 1 for each review.



Literature Review

- **Sentiment Analysis Of Product Reviews – A Survey**

  -Dishi Jain, Bitra Harsha Vardhan, Saravanakumar Kandasamy

With expanding development of the web network, informal communities, online gateways, audits, reviews, suggestions, evaluations, and input are produced by clients and it can be about anything like books, individuals, items, explore, occasions, and so on. These opinions become advantageous for organizations, political bodies, and people. While this is intended to be valuable, a greater part of this client produced content requires utilizing the opinion mining methods for sentiment analysis. Sentiment Analysis is a field in which we study about feelings, conclusions and subjectivity of opinions. This review paper is a similar investigation of numerous as of late proposed calculations' upgrades and different slant examination applications. Mainly for product reviews that how sentiment analysis can be used to recommend a product based on reviews by the client. The related fields to estimation investigation that pulled in specialists as of late are examined. The fundamental objective of this review is to give a nearly full picture of sentiment analysis, its sorts and characterization. The primary commitments of this work include the complex orders of late articles and the outline of the ongoing pattern of research in the sentiment analysis and its related territories.

METHODOLOGY

The study approach is as per the following: brief clarification to the renowned feature selection methods and sentiment classification calculations speaking to some related fields to sentiment analysis are examined. At that point the commitment of these articles to these calculations is displayed representing how they utilize these calculations to tackle uncommon issues in sentiment analysis. The principal focus of this study is to see how sentiment analysis is used in product reviews.

- # Recommendation System using kNN

  -by Sofia Porta

A common task of recommender systems is to improve customer experience through personalized recommendations based on prior implicit feedback. These systems passively track different sorts of user behavior, such as purchase history, watching habits and browsing activity, in order to model user preferences. Unlike the much more extensively researched explicit feedback, we do not have any direct input from the users regarding their preferences.

kNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors.

Recommendation system should be on Implicit data and Explicit data :-

Explicit data :

The dictionary meaning of explicit is to state clearly and in detail. Explicit feedback data as the name suggests is an exact number given by a user to a product. Some of the examples of explicit feedback are ratings of movies by users on Netflix, ratings of products by users on Amazon. Explicit feedback takes into consideration the input from the user about how they liked or disliked a product. Explicit feedback data are quantifiable.

Explicit data should be like in the form of ratings.

When was the last time you rated a movie on Netflix? People normally rate a movie or an item on extreme feelings—either they really like the product or when they just hated it. The latter being more prominent. So, chances are your dataset will be largely filled with a lot of positive ratings but very less negative ratings.

Explicit feedback is hard to collect as they require additional input from the users. They need a separate system to capture this type of data. Then you've to decide whether you should go with ratings or like/dislike option to collect the feedback. Each having their merits/demerits.

Explicit feedback doesn't take into consideration the context of when you were watching the movie. Let us understand with an example. You watched a documentary and you really liked it and you rated it well. Now, this doesn't mean you would like to see a lot many documentaries but with explicit ratings, this becomes difficult to take into

consideration. I like to binge watch The Office tv series while having dinner and I would give it a high rating 4.5/5 but that doesn't mean that I would watch it at any time of the day.

There is another problem with explicit ratings. People rate movies/items differently. Some are lenient with their ratings while others are particular about what ratings they give. You need to take care of bias in ratings from users as well.

Implicit data :

The dictionary meaning of implicit is suggested though not stated clearly. And that's exactly what implicit feedback data represents. Implicit feedback doesn't directly reflect the interest of the user but it acts as a proxy for a user's interest.

Examples of implicit feedback datasets include browsing history, clicks on links, count of the number of times a song is played, the percentage of a webpage you have scrolled—25%, 50% or 75 % or even the movement of your mouse.

If you just browsed an item that doesn't necessarily mean that you liked that item but if you have browsed this item multiple times that gives us some confidence that you may be interested in that item. Implicit feedback collects information about the user's actions.

Implicit feedback data are found in abundance and are easy to collect. They don't need any extra input from the user. Once you have the permission of the user to collect their data, your dependence on the user is reduced.

Existing work

Every day thousands of people leave their opinion online. They are being posted, tweeted, shared, left on online retailers' own sites and on platforms like Amazon,flipkart,snapdeal,myntra ,Nykaa and many more other ecommerce websites. Consumers are leaving their opinions about products they are considering, items they have bought or services they are using.

All of this information lives on the internet and is informing not only the close network of people giving these opinions, but also other internet users who are discovering these opinions when carrying out their own product research.

If you are running an e-commerce website then it can be a huge challenge to discover what is being said about your site, your brand or the products you sell. Even harder is to sort and make sense of the mountain of feedback that exists online. Harder still is the ability to draw upon granular levels of feedback about specific product aspects within the data.

How Apple is doing it

The way Apple presents its products and establishes them on the market is a fine example of sentiment analysis application for the benefit of market research and competitor analysis.

Product Analytics

The use of sentiment analysis in product analytics stems from reputation management. Conceptually, it is very similar to brand monitoring. But instead of brand mentions, it goes for specific comments and remarks regarding the product and its performance in specific areas (user interface, feature performance, etc).

This kind of insight is very important at the initial stages with MVP when you need to try the product by fire (i.e. actual users) and make it as polished as possible.

At this stage, the most basic way to apply sentiment analysis is to gather and categorize feedback for further improvements.

Sentiment analysis algorithms can do the dirty work and show what kind of feedback goes from which segment of the audience and at what it points.

Usually, the whole thing is divided between the following types:

Brand keywords
Brand-adjacent keywords
Customer needs
Customer sentiment
Competitors analysis (based on similar criteria)

As a result, this can be a significant factor in the product's successful establishment on the market.

At the later stages, the use of sentiment analysis in product analytics merges with brand monitoring and provides a multi-dimensional view on the product and its brand.How the brand/product is perceived by various target audience segments?

Which elements of the product or its presentation are the points of contention and in what light?

Requirements

DATA REQUIREMENT

A large dataset with the ratings and reviews columns is necessary in order to cater to the need of a large sample size. The categories column can help categorise the problems for more modularity

TOOLS USED FOR ANALYSIS

1. NLTK toolkit for natural language processing
2. Pandas and matplotlib for data manipulation and visualisation
3. Scikit-learn for constructing ML models and making predictions

Language :

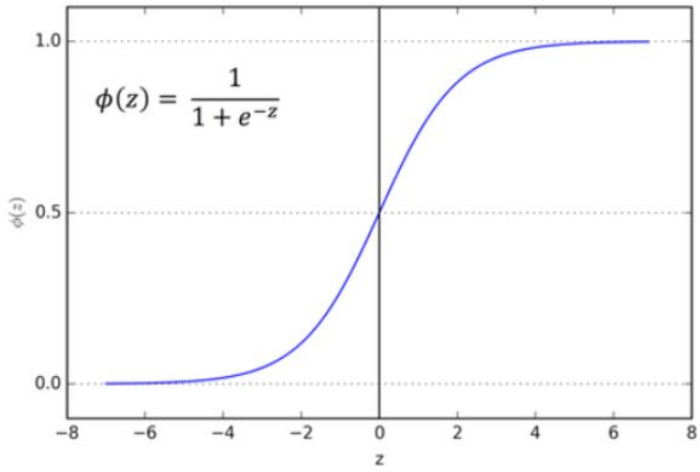
Python

Proposed Work

The idea for analyzing customer reviews and making accurate predictions out of it in this project follows a two-fold approach:

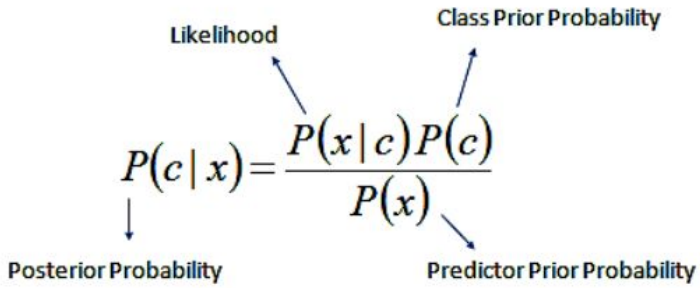
Sentiment Analysis

The basic Sentiment Analysis methodology is a supervised technique and extracts features from textual data in the form of quantitative attributes to arrive at an understanding of the social sentiment or feeling towards a particular product, either positive, negative and neutral. But what if we don't have this target sentiment in our dataset. This is where our proposal comes in where we move forward with product ratings as the predicted outcome that is required to estimate the sentiment that a particular review is trying to convey. Working under the assumption that a rating above 3.0 is positive and below 3.0 is negative, the ratings are marked as either 1(positive) or 0(negative). This binary data column is now our target variable that is to be modelled using the textual reviews. This prediction model has been build using two machine learning techniques – logistic regression that is designed specifically to operate on binary outcome data by constraining the upper and lower bounds of the numerical target between 0 and 1 using a sigmoid function as follows,



and the second technique is Multinomial Naïve Bayes which is a probabilistic model that calculates class labels, in this case, sentiments by training on vectors of feature

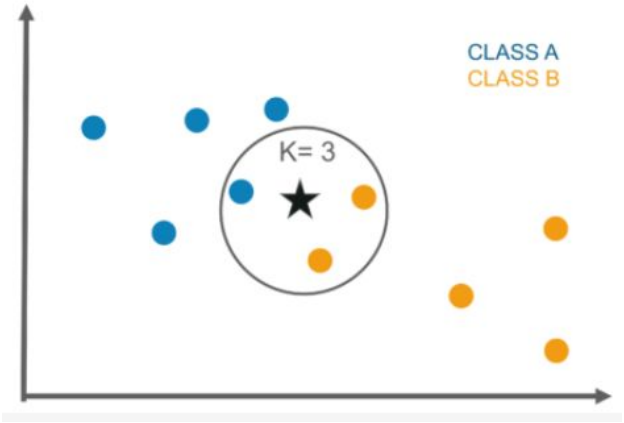
values, in this case, unique word counts in every review under consideration, using the following formula as follows,



where, P(c|x) is the probability of a particular sentiment given the feature values of the word vectors in the documents. Having predicted the corresponding sentiments, we sieve in the negative reviews for computing the similarity with a given negative review, to segment the customer issue with a group of similar issues throwing light on the solution that can be provided for the said customer and product.

Customer Recommendation

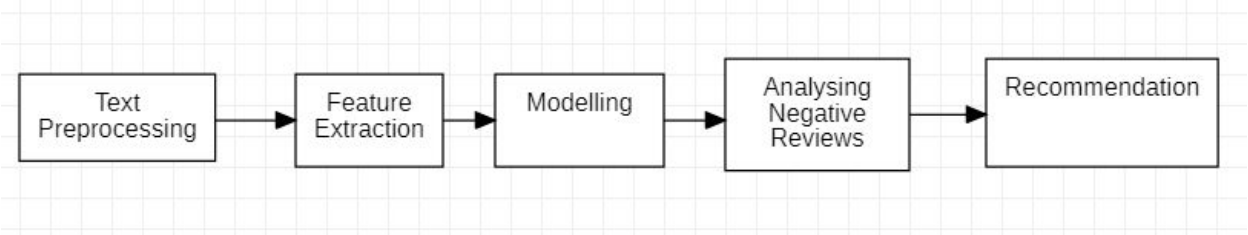
Once the sentiments have been collected using the predictor model, these outcomes are used to build a recommendation engine for the customers, which uses a collaborative filtering approach that utilizes a user's past ratings as well as similar ratings made by other users to suggest items that the user may be interested in. This is achieved by constructing a pivot table of users and products with the table contents as the predicted sentiments. In this pivot table, each row is a vector of user sentiment for that particular product. The core idea is that for a product selected by a user, the system should recommend similar products. This has been implemented using the K-nearest neighbors approach which predicts the values of new data-points based on how close they are to other points. This closeness is measured in terms of a similarity measure which in this case, is the cosine similarity between the product vectors. The KNN algorithm is depicted in the following graph:



In the graph, the star point is the product that a user has rated, and when we specify the no. of nearest neighbors as 3, the engine returns the data-points that are enclosed within the circle drawn around the point.

Flowchart and Module Description

The project is designed to follow the linear flow of the following flowchart diagram where each major language processing task is represented as separate modules:



Let us now elaborate each of the modules and see the functionalities involved in them.

Text Preprocessing

Before we can build the models for sentiment analysis and recommendation, we need to decide on the data columns that will be used for the computation. In the dataset under consideration, we select the columns 'reviews.rating' and 'reviews.text' that will be used for sentiment analysis. Working under the assumption that we will be using the ratings as the supervised labels for the sentiments, we duplicate this column for secondary computation and replace the main column values with 0 and 1.

Since our main target for analyzing the sentiments will be the user textual reviews, we need to preprocess this column. The first step is to remove the various punctuation, numbers and special characters as these do not contribute as much as the words itself. The second step is for removing the stop words from the text, which include common words like 'the', 'an', etc., along with tokenizing the text into separate words. We have used the NLTK toolkit for this purpose. Having acquired the individual words in each review, we can now the stem the words to by removing the prefixes using the PorterStemmer in NLTK, as we are more focused on the occurrence of the actual root word rather than its variants. Once these stemmed words are acquired for each review,

we concatenate them back together into a new column called ‘Cleaned reviews’, which will be used for feature extraction in the next module.

Feature Extraction

It is an important step in any machine-learning methodology, where the available input is transformed into a non-redundant and organized set of features or a feature vector that is used for further computations. As we are dealing with raw text in our project, the models need some representation of these text data which are quantifiable in order to process. In this project, we will be using the two classical feature extraction techniques in natural language processing as follows:

- **Bag of Words (BoW)**
  It transforms tokens into a set of features, where each word is used as a feature for training the classifier. In this case, it takes the collection of text reviews and creates a vocabulary of all unique words to form a training corpus. This is achieved through a process called Text Vectorization where each word corresponds to a column and each row corresponds to a review and the values in this matrix is the count of all these unique words in each of the reviews. This matrix is essentially the bag of words which quantities can be fed into a classifier to predict the outcome sentiment. We have used the CountVectorizer in the scikit-learn package to obtain the bag-of-words by using the bigram language model i.e. considering two adjacent tokens for counting as it provides more context for a particular piece of text rather than only considering the words individually. For example, considering three reviews with a unigram model, the bag of words would look something like this:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

- **TF – IDF**
  Term frequency – inverse document frequency is a measure that depicts how important a word is in a review as well as the entire training corpus. It is a weighting factor that increases proportionally to the word occurrence count in the review and decreases with the count of reviews in the corpus that contains the word. It comprises of two parts –

Term Frequency – measures how frequently a word occurs in the entire document or review and can be formulated as:

$$tf(w_i, r_j) = \frac{No.\,of\,times\,w_i\,occurs\,in\,r_j}{Total\,no.\,of\,words\,in\,r_j}$$

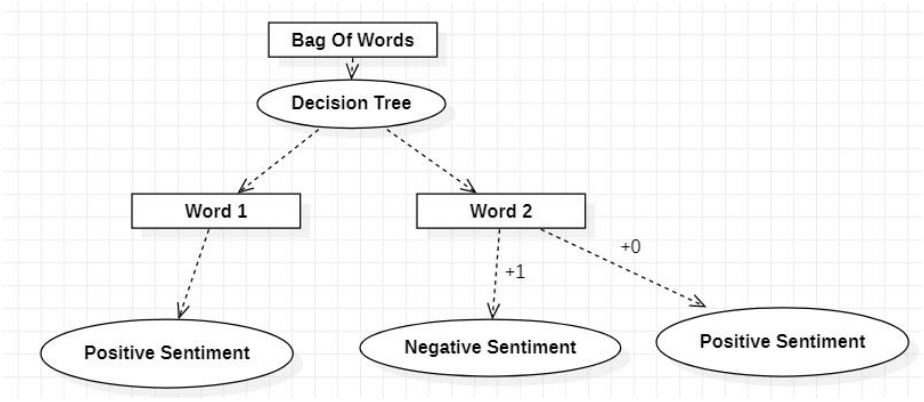
Inverse Document Frequency – it is a measure of whether a term is rare or frequent across all the reviews in the entire corpus. It is normalized by a logarithmic transform and is acquired by dividing the total number of reviews by the number of reviews containing a particular word.

$$idf(d, D) = \log \frac{|D|}{\{d \epsilon D : t \epsilon D\}}$$

The TF-IDF is the product of the term frequency and the inverse document frequency with a score approaching 1 if the word is rare across the entire corpus.
In this project, after acquiring the bag-of-words, we feed this as an input to the TfidfTransformer in the scikit-learn package to obtain the TF-IDF measure for our text reviews training corpus, which is used to build classifier models.

Modelling
This is the main part of our project where we perform sentiment analysis by constructing supervised models that can predict the sentiment with the features that are obtained from the previous module. Three models are used for this purpose, in order to test the level of accuracy that is obtainable:

- **Decision Tree**
  It is a ML classification model that follows a tree structure where each internal node performs a test on an attribute, in this case each unique word of the training corpus (bag-of-words) and based on the results, the nodes branch out and eventually reach the leaves which are the predicted sentiments (0 or 1). For a sample corpus with only two words the decision tree classifier for predicting sentiments would look something like this,



As can be seen from the figure, the bag-of-words that is obtained previously is fed to the DecisionTreeClassifier in the scikit-learn package which makes use of the 'entropy' method to split the nodes at each level of the decision tree that can be defined as the measure of the purity of each split calculated as follows:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

where p_i is the probability of occurrence of sentiment 'i'. The branches in the decision tree represent the various outcomes of the splitting attribute, like in this diagram, an outcome of 1 for 'Word 2' gives a negative sentiment while a value of 0 gives a positive sentiment. One shortcoming of decision tree however is that for a new document, it is possible that it can return a wrong sentiment value which can be resolved by analyzing the probabilities of being a positive and negative sentiment and this probability value is not provided by decision tree and hence we can use the model mentioned next.

- **Naïve Bayes Classifier**
  It is a conditional probability model, where given a vector of n features, in this case a particular instance of occurrence of each unique word of same dimension as the training corpus, the model finds the probability of both positive and negative sentiment and whichever comes out to be greater the input is assigned that particular sentiment.
  To train the model, we have used the TF-IDF feature set as input. One thing to notice is that we have used Multinomial Naïve Bayes instead of normal Naïve Bayes so that the algorithm is scalable for more than two sentiments as well. Having trained the model, we now provide two sample input reviews and estimate the probability of both positive and negative sentiments. Thereafter, a comparison can be drawn regarding the overall nature of the predicted sentiment between the two reviews.

- **Logistic Regression**

  We will use **LogisticRegression** for model development as for high dimensional sparse data like ours, **LogisticRegression** often works best.
  Grid Search: for parameter tuning of LogisticRegression. We want to determine what value of coefficient 'C' provides better accuracy.Like Naive Bayes here also we have used the TF-IDF feature set as input.

Having acquired the predicted sentiments, we then measure the accuracy for each of the three models using the F1 score in scikit-learn metrics given by:

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

TP = number of true positives
FP = number of false positives
FN = number of false negatives

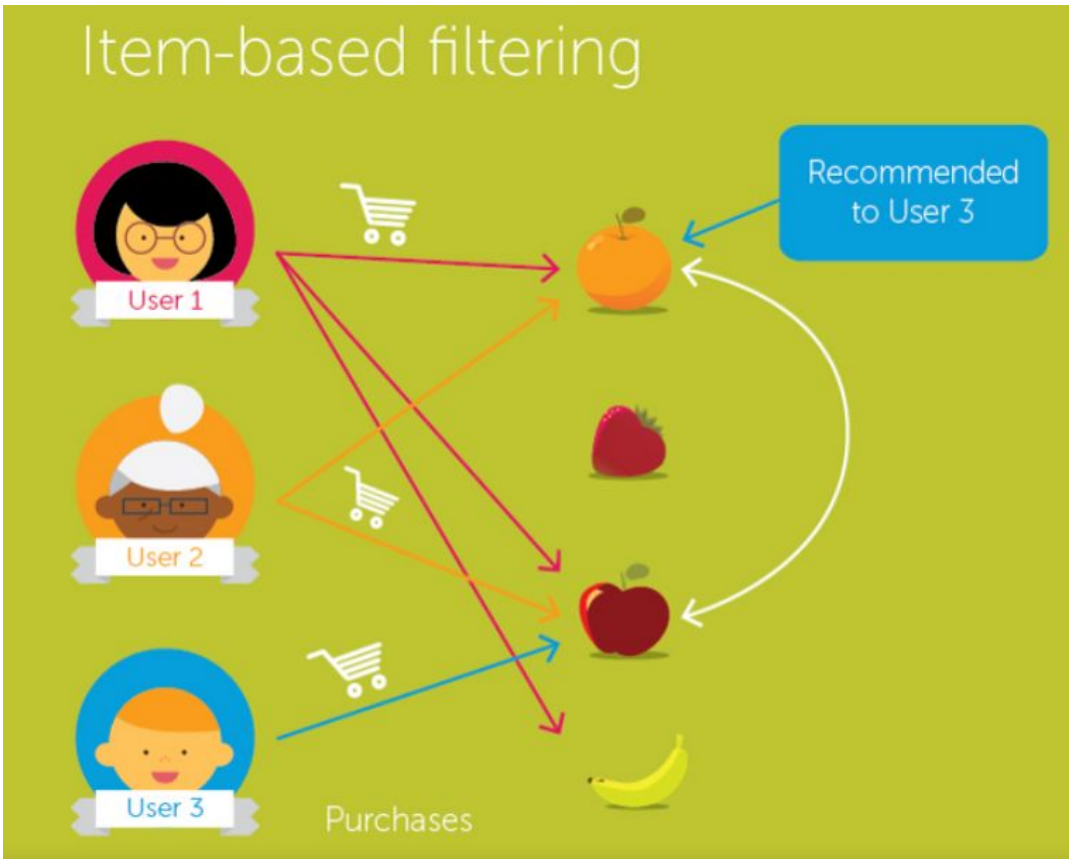
Analyzing Negative Reviews
Having filtered the negative reviews, we find the top similar reviews compared with a target review using cosine similarity which is computed as follows:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

where A and B are individual word vectors for two reviews that are being compared using CountVectorizer. The cosine similarity gives us similar reviews for a target negative review that is necessary to draw a comparison of the problems faced by a particular customer for better analysis. This helps in segmenting the issues faced by a customer to a small set of similar issues and suggest the solutions that were recommended for those issues to the customer.

Recommendation
We have built a recommendation engine in this module from the predicted sentiments for all products from all available users, which filters the data and recommends the most relevant products to the user by training on the past data of the customers. Also

known as collaborative filtering, this approach makes recommendations based on users with similar tastes.
We have aimed for item-item collaborative filtering in this module which computes the cosine similarity score between each pair of items that the range of available users might have used. The idea is that for a given input item, we can now suggest a list of similar items that the user might like. This is achieved by the K-nearest neighbors algorithm.
First, a pivot table is created using the items as rows and the customers as columns, with the values as the predicted sentiments. In order to feed this to a KNN model, we first convert it to a sparse matrix and give this as an input with the similarity metric as 'cosine' to train the model. Using this model, we now obtain the indices of the top 5 similar items and suggest it to the user with its average ratings calculated from the duplicated ratings column. The sparse matrix is actually a set of item vectors whose distance from other vectors is estimated and the engine suggests the one with the lowest distance. Item-item collaborative filtering for customers would look something like this:



Program Code

(Sentiment Analysis using Decision Tree and Naïve Bayes with Recommendation)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
%matplotlib inline
#Reading amazon products review dataset
df=pd.read_csv('D:\\5th sem materials\\NLP\\amazon.csv')
df.head(5)
#duplicating the reviews rating column and replacing the original column values with sentiments 1.0 and 0.0
dt1=df[["name","categories","reviews.rating","reviews.text","reviews.username"]]
dt1["ratings"]=dt1["reviews.rating"]
dt1["reviews.rating"].replace({1.0:0.0},inplace=True)
dt1["reviews.rating"].replace({2.0:0.0},inplace=True)
dt1["reviews.rating"].replace({3.0:0.0},inplace=True)
dt1["reviews.rating"].replace({4.0:1.0},inplace=True)
dt1["reviews.rating"].replace({5.0:1.0},inplace=True)
dt1.head(5)
#checking count of missing values
dt1.isnull().sum()
#removing missing values
dt=dt1.dropna(axis=0,how='any')
dt.isnull().sum()
#removing punctuation,numbers and special characters
dt['reviews.text']=dt['reviews.text'].str.replace("[^a-zA-Z#]"," ")
dt.head(5)
#Removing stop words and tokenizing the text
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
dt['reviews.text']=dt['reviews.text'].astype(str)
s=dt['reviews.text']
```

```
stop_words=stopwords.words('english')
filtered_reviews=[]
for text in s:
   word_tokens=word_tokenize(text)
   word_tokens
   filtered_sentence=[w for w in word_tokens if not w in stop_words]
   filtered_reviews.append(filtered_sentence)
filtered_reviews
#Stemming the reviews by removing the suffixes
from nltk import PorterStemmer
ps=PorterStemmer()
stemmed_reviews=[]
for text in filtered_reviews:
   L=[]
   for w in text:
      L.append(ps.stem(w))
   stemmed_reviews.append(L)
stemmed_reviews
#creating a column of cleaned reviews
clean=[]
for text in stemmed_reviews:
   st=""
   for w in text:
      st=st+w+" "
   clean.append(st)
#stemmed_reviews
dt['Cleaned reviews']=[rev for rev in clean]
dt.head(5)
#extracting bag-of-words feature
from sklearn.feature_extraction.text import CountVectorizer
bvect=CountVectorizer(max_df=0.9,min_df=2,stop_words='english',ngram_range=(1,2))
bag_of_words=bvect.fit_transform(dt['Cleaned reviews'])
#train-test splitting
train_bag=bag_of_words[:27867]
Y=dt.iloc[0:27867,2]
from sklearn.model_selection import train_test_split
x_train_bag,x_valid_bag,y_train_bag,y_valid_bag=train_test_split(train_bag,Y,test_size=0.3,ran
dom_state=2)
```

```
#training a decision tree classifier
from sklearn.tree import DecisionTreeClassifier
dc=DecisionTreeClassifier(criterion='entropy',random_state=1).fit(x_train_bag,y_train_bag)
dc_pred=dc.predict(x_valid_bag)
#finding F1 score for decision tree
from sklearn.metrics import f1_score
f1_score(y_valid_bag,dc_pred,average='weighted')
#Finding sentiment of new review using Decision Tree
sent=["Bad product","Good product"]
X_new_counts=bvect.transform(sent)
predicted=dc.predict(X_new_counts)
predicted
#extracting Tf-idf feature
from sklearn.feature_extraction.text import TfidfTransformer
tf_transformer=TfidfTransformer(use_idf=False).fit(bag_of_words)
X_train_tf=tf_transformer.transform(bag_of_words)
#training multinomial naive bayes classifier
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train_tf, dt['reviews.rating'])
#finding F1 score for naive bayes
pred2=clf.predict(tf_transformer.transform(x_valid_bag))
f1_score(y_valid_bag,pred2,average='weighted')
#finding probability of sentiments for new reviews
docs_new=['Bad product','Good product']
X_new_counts=bvect.transform(docs_new)
X_new_tfidf=tf_transformer.transform(X_new_counts)
predicted=clf.predict_proba(X_new_tfidf)*100
predicted
#Grouped bar chart for comparing probabilities of positive and negative reviews
pro1=[predicted[0][0],predicted[1][0]]
pro2=[predicted[0][1],predicted[1][1]]
ypos=np.arange(2)
plt.bar(ypos,tuple(pro1),width=0.3,label="Negative",color="red")
plt.bar(ypos+0.3,tuple(pro2),width=0.3,label="Positive",color="green")
plt.title("Probability of positive and negative sentiment")
plt.xlabel("Review")
plt.ylabel("Probability")
plt.xticks(ypos+0.3/2,(docs_new[0],docs_new[1]))
```

```
plt.legend(labels=["Negative","Positive"])
plt.show()
#segregating the negative reviews
dts=dt[dt['reviews.rating']==0.0]
dts.head()
import string
def clean_string(text):
   text=''.join([word for word in text if word not in string.punctuation])
   text=text.lower()
   text=' '.join([word for word in text.split() if word not in set(stopwords.words("english"))])
   return text
sentences=dts['reviews.text']
cleaned=list(map(clean_string,sentences))
sentences
cleaned
#creating bag-of-words for negative reviews
from sklearn.metrics.pairwise import cosine_similarity
vectorizer=CountVectorizer().fit_transform(cleaned)
vectors=vectorizer.toarray()
#calculating cosine_similarity between two reviews
def cosine_sim_vectors(vec1,vec2):
   vec1=vec1.reshape(1,-1)
   vec2=vec2.reshape(1,-1)
   return cosine_similarity(vec1,vec2)[0][0]
csim_max=[]
for i in range(0,len(vectors)):
   csim_max.append(cosine_sim_vectors(vectors[0],vectors[i]))
csim_max
dts["cos sim"]=csim_max
rstdf=dts.sort_values(by="cos sim",ascending=False)
rstdf.head(5)
d1=rstdf.iloc[1:6,]
#comparing similar negative reviews
plt.rcdefaults()
fig,ax=plt.subplots(figsize=(10,10))
y_pos=np.arange(len(d1["reviews.text"]))
error=np.random.rand(len(d1["reviews.text"]))
ax.barh(y_pos,d1["cos sim"],xerr=error,align='center')
```

```
ax.set_yticks(y_pos)
ax.set_yticklabels(d1["reviews.text"])
ax.invert_yaxis()
ax.set_xlabel("Cosine similarity")
ax.set_title("Similar reviews")
plt.show()
#checking count of unique products
dt['name'].value_counts()
dt.shape
#predicting sentiment for recommendation engine
pred_red=clf.predict(tf_transformer.transform(bag_of_words))
dt['pred_sentiment']=pred_rec
dt.head()
#creating pivot table with item vectors
dp=dt.pivot_table(index='name',columns='reviews.username',values='pred_sentiment').fillna(0
)
dp
#forming sparse matrix for input to KNN
from scipy.sparse import csr_matrix
m=csr_matrix(dp.values)
print(m)
#training KNN model with cosine similarity and created sparse matrix
from sklearn.neighbors import NearestNeighbors
model=NearestNeighbors(metric='cosine',algorithm='brute')
model.fit(m)
#selecting an item for recommendation
selected_item=dp.index[1]
selected_item
q=0
for i in range(dp.shape[0]):
  if(dp.index[i]==selected_item):
    q=i
#finding distance and indices for recommended items
dist,indices=model.kneighbors(dp.iloc[q,:].values.reshape(1, -1),n_neighbors=5)
#displaying recommended items
for i in range(0,len(dist.flatten())):
  if i==0:
    print('Recommendations for {0}:\n'.format(dp.index[q]))
```

```
    else:
        print('{0}: {1}'.format(i,dp.index[indices.flatten()[i]]))
rec_list=[]
for i in dp.index[indices.flatten()]:
    rec_list.append(i)
#finding mean ratings for recommended items
av_sent=[]
for rec in rec_list:
    sumA=0
    countA=0
    for i in range(dt.shape[0]):
        if(rec==dt.iloc[i,0]):
            sumA+=dt.iloc[i,5]
            countA+=1
    av_sent.append(sumA/countA)
av_sent
#comparing the recommended items through a bar chart via their mean ratings
plt.figure()
plt.barh(rec_list,av_sent)
```

(Sentiment Analysis using Logistic Regression)

#Loading the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv('./1429_1.csv')
df.head(5)
#duplicating the reviews rating column and replacing the original column values with
sentiments 1.0 and 0.0
df = pd.read_csv('./1429_1.csv')
df.head(5)
dt=df[["reviews.text","reviews.rating"]]
dt["reviews.rating"].replace({1.0:0.0},inplace=True)
dt["reviews.rating"].replace({2.0:0.0},inplace=True)
dt["reviews.rating"].replace({3.0:0.0},inplace=True)
dt["reviews.rating"].replace({4.0:1.0},inplace=True)
```

```
dt["reviews.rating"].replace({5.0:1.0},inplace=True)
dt.head(5)
```
#Bag of words / Bag of N-grams model
```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer()
docs = np.array([
      'The sun is shining',
      'The weather is sweet',
      'The sun is shining, the weather is sweet, and one and one is two'])
bag = count.fit_transform(docs)
print(count.vocabulary_)
print(bag.toarray())
```
#Word relevancy using term frequency-inverse document frequency
```
np.set_printoptions(precision=2)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf = TfidfTransformer(use_idf=True, norm='l2', smooth_idf=True)
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
```

#Calculate tf-idf of the term *is*:
```
tf_is = 3
n_docs = 3
idf_is = np.log((n_docs+1) / (3+1))
tfidf_is = tf_is * (idf_is + 1)
print('tf-idf of term "is" = %.2f' % tfidf_is)
tfidf = TfidfTransformer(use_idf=True, norm=None, smooth_idf=True)
raw_tfidf = tfidf.fit_transform(count.fit_transform(docs)).toarray()[-1]
raw_tfidf
l2_tfidf = raw_tfidf / np.sqrt(np.sum(raw_tfidf**2))
l2_tfidf
```
#Data Preparation
```
df.loc[0, 'reviews.text'][-50:]
import re
def preprocessor(text):
   text = re.sub('<[^>]*>', '', text)
   emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', text)
   text = re.sub('[\W]+', ' ', text.lower()) +\
      ' '.join(emoticons).replace('-', '')
```

```
    return text
preprocessor(df.loc[0, 'reviews.text'][-50:])
preprocessor("</a>This :) is :( a test :-)!")
df['review.text'] = df['review.text'].apply(preprocessor)
```

#Tokenization of documents

```
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()
def tokenizer(text):
    return text.split()
def tokenizer_porter(text):
    return [porter.stem(word) for word in text.split()]
tokenizer('runners like running and thus they run')
tokenizer_porter('runners like running and thus they run')
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')
[w for w in tokenizer_porter('a runner likes running and runs a lot')[-10:]
if w not in stop]
```

#Document classification via a logistic regression model

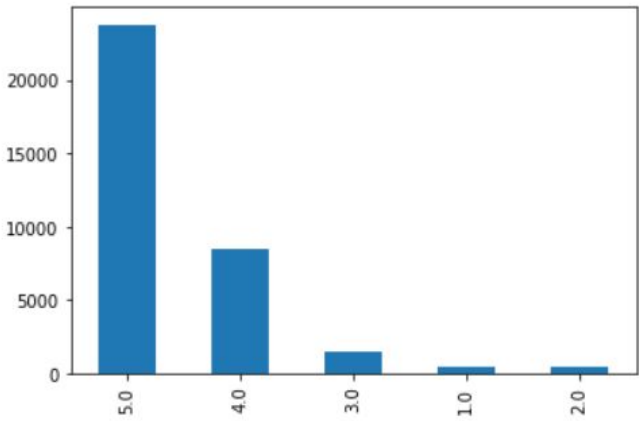
```
X_train = df.loc[:25000, 'reviews.text'].values
y_train = df.loc[:25000, 'reviews.rating'].values
X_test = df.loc[25000:, 'reviews.text'].values
y_test = df.loc[25000:, 'reviews.rating'].values
print("Review: ", X_train[1:2], "\n")
print("Sentiment: ", y_train[1])
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import GridSearchCV

tfidf = TfidfVectorizer(strip_accents=None,
                        lowercase=False,
                        preprocessor=None)

param_grid = [{'vect__ngram_range': [(1, 1)],
               'vect__stop_words': [stop, None],
               'vect__tokenizer': [tokenizer, tokenizer_porter],
```

```
              'clf__penalty': ['l2'],
              'clf__C': [1.0, 10.0, 100.0]},
             {'vect__ngram_range': [(1, 1)],
              'vect__stop_words': [stop, None],
              'vect__tokenizer': [tokenizer, tokenizer_porter],
              'vect__use_idf':[False],
              'vect__norm':[None],
              'clf__penalty': ['l2'],
              'clf__C': [1.0, 10.0, 100.0]},
             ]

lr_tfidf = Pipeline([('vect', tfidf),
                     ('clf', LogisticRegression(random_state=0))])

gs_lr_tfidf = GridSearchCV(lr_tfidf, param_grid,
                           scoring='accuracy',
                           cv=5,
                           verbose=1,
                           n_jobs=-1)
#Model accuracy
print('Best parameter set: %s' % gs_lr_tfidf.best_params_)
print('CV Accuracy:%.3f' % gs_lr_tfidf.best_score_)
clf = gs_lr_tfidf.best_estimator_
print('Test Accuracy: %.3f' % clf.score(X_test, y_test))
#Load saved model from disk
import pickle
filename = 'saved_model.sav'
gs_lr_tfidf = pickle.load(open(filename, 'rb'))
```

Results and Output

Sentiment analysis

Rating count



Missing values in dataset

```
Out[4]: name                    6760
        categories                 0
        reviews.rating            33
        reviews.text               1
        reviews.username           2
        ratings                   33
        dtype: int64
```

Preprocessed dataset for computation

Out[10]:

	name	categories	reviews.rating	reviews.text	reviews.username	ratings	Cleaned reviews
0	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	This product so far has not disappointed My c...	Adapter	5.0	thi product far disappoint My children love us...
1	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	great for beginner or experienced person Boug...	truman	5.0	great beginn experienc person bought gift love
2	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	Inexpensive tablet for him to use and learn on...	DaveZ	5.0	inexpens tablet use learn step nabi He thrill ...
3	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	I ve had my Fire HD two weeks now and I love...	Shacks	4.0	I fire HD two week I love thi tablet great val...
4	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	I bought this for my grand daughter when she c...	explore42	5.0	I bought grand daughter come visit I set user ...

Accuracy of Decision Tree Classifier

```
Out[13]: 0.8995341075764522
```

Predicted sentiment of new review using Decision Tree

```
Out[14]: array([1., 1.])
```

Accuracy of Naïve Bayes Classifier

```
Out[17]: 0.89053343547326
```

Sentiment Probabilities for new reviews

```
Out[18]: array([[ 6.24922457, 93.75077543],
                [ 0.75524601, 99.24475399]])
```

TF-IDF

```
tf_is = 3
n_docs = 3
idf_is = np.log((n_docs+1) / (3+1))
tfidf_is = tf_is * (idf_is + 1)
print('tf-idf of term "is" = %.2f' % tfidf_is)
```

```
tf-idf of term "is" = 3.00
```

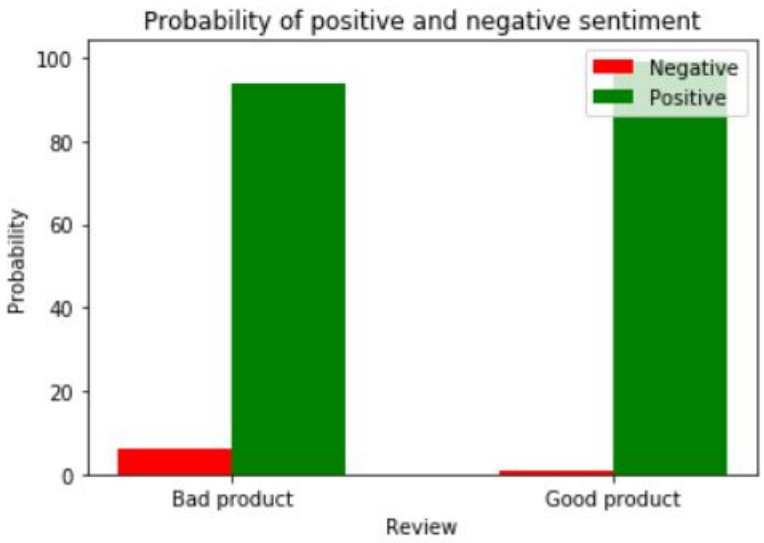
CV-Accuracy

```
Best parameter set: {'clf__C': 10.0, 'clf__penalty': 'l2', 'vect__ngram_range'
enizer': <function tokenizer at 0x000001544E5C0488>}
CV Accuracy:0.897
```

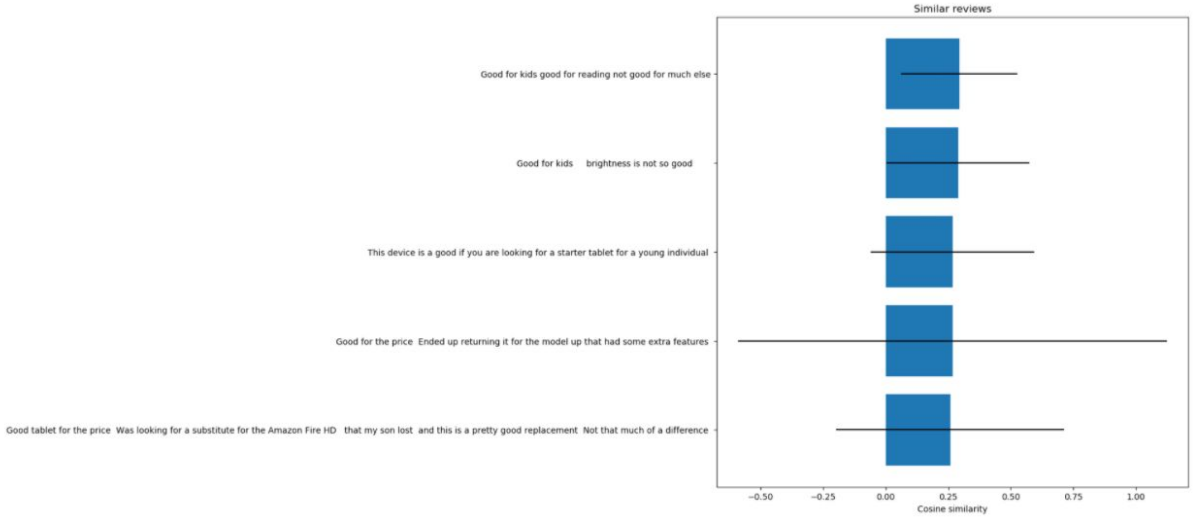
Accuracy of Logistic Regression

```
clf = gs_lr_tfidf.best_estimator_
print('Test Accuracy: %.3f' % clf.score(X_test, y_test))
```

```
Test Accuracy: 0.899
```



Generating top 5 negative reviews with respect to cosine similarity to understand user problem experienced



Recommendation

Dataset with predicted sentiments for recommendation

Out[37]:

	name	categories	reviews.rating	reviews.text	reviews.username	ratings	Cleaned reviews	pred_sentiment
0	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	This product so far has not disappointed My c...	Adapter	5.0	thi product far disappoint My children love us...	1.0
1	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	great for beginner or experienced person Boug...	truman	5.0	great beginn experienc person bought gift love	1.0
2	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	Inexpensive tablet for him to use and learn on...	DaveZ	5.0	inexpens tablet use learn step nabi He thrill ...	1.0
3	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	I ve had my Fire HD two weeks now and I love...	Shacks	4.0	I fire HD two week I love thi tablet great val...	1.0
4	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	Electronics,iPad & Tablets,All Tablets,Fire Ta...	1.0	I bought this for my grand daughter when she c...	explore42	5.0	I bought grand daughter come visit I set user ...	1.0

Pivot table of item vectors

Out[38]:

reviews.username	.	000G	00100z	00Gambler	00worm	01-Jan	01bandit12
name							
All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Magenta	0.0	1.0	0.0	1.0	0.0	0.0	0.0
All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 32 GB - Includes Special Offers, Magenta	0.0	0.0	0.0	0.0	0.0	0.0	0.0
All-New Kindle E-reader - Black, 6 Glare-Free Touchscreen Display, Wi-Fi - Includes Special	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Sparse matrix for KNN

```
(0, 1)        1.0
(0, 3)        1.0
(0, 17)       1.0
(0, 22)       1.0
(0, 23)       1.0
(0, 27)       1.0
(0, 28)       1.0
(0, 40)       1.0
(0, 46)       1.0
(0, 52)       1.0
(0, 69)       1.0
(0, 76)       1.0
(0, 78)       1.0
```

Trained KNN model for recommendation

```
Out[40]: NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine',
                          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                          radius=1.0)
```

Selected item for recommendation

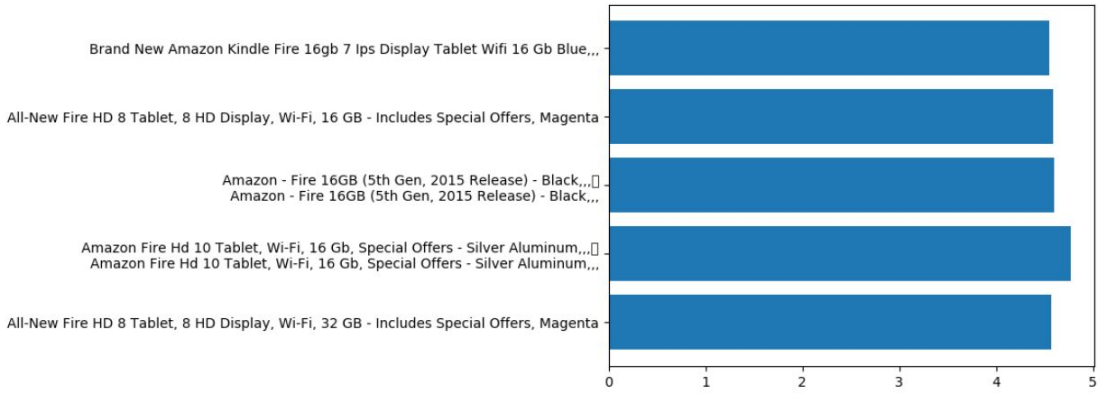
```
Out[42]: 'All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 32 GB - Includes Special Offers, Magenta'
```

Recommendations for the selected item

```
Recommendations for All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 32 GB - Includes Special Offers, Magenta:

1: Amazon Fire Hd 10 Tablet, Wi-Fi, 16 Gb, Special Offers - Silver Aluminum,,,
Amazon Fire Hd 10 Tablet, Wi-Fi, 16 Gb, Special Offers - Silver Aluminum,,,
2: Amazon - Fire 16GB (5th Gen, 2015 Release) - Black,,,
Amazon - Fire 16GB (5th Gen, 2015 Release) - Black,,,
3: All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi, 16 GB - Includes Special Offers, Magenta
4: Brand New Amazon Kindle Fire 16gb 7 Ips Display Tablet Wifi 16 Gb Blue,,,
```

Average ratings for recommended items

```
Out[48]: [4.568493150684931, 4.7734375, 4.6, 4.5867093105899075, 4.55082284607938]
```



Conclusion

From the project, we can see that the Decision Tree Classifier has an accuracy of about 89.9% , Naïve Bayes Classifier has an accuracy of 89% while logistic regression has accuracy around 89.9% on the reviews corpus. However, the decision tree classifier is prone to predicting incorrect sentiments. This issue is resolved by comparing the probabilities of two input reviews using Naïve Bayes, where the review with a greater positive probability is considered positive. The cosine similarities between a target negative review and other reviews are almost identical, enabling a much better understanding of the customer problem. Lastly, for the recommended items, a measure of average ratings shows little or negligible difference indicating a higher accuracy of the recommendation engine.

Future Work

This project can be enhanced and modified in the following ways:

- The models for sentiment analysis can be extended to include neutral reviews as well in the dataset
- The negative reviews suggested using cosine similarity can be enhanced to include the item recommendation to suggest items without the problems experienced by the customer
- The recommendation engine is prone to a cold start when enough data is not available and can be solved by using hybrid recommenders

References

- Dataset : https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products
- Twitter sentiment analysis : https://www.academia.edu/44189555/Twitter_Sentiment_analysis_using_different_Algorithms
- Sentiment Classification using Logistic Regression : https://towardsdatascience.com/sentiment-classification-using-logistic-regression-in-pytorch-e0c43de9eb66
- Derek G Bridge. 2002. Towards Conversational Recommender Systems: A Dialogue Grammar Approach.. In ECCBR Workshops.
- https://medium.com/@rnbrown/more-nlp-with-sklearns-countvectorizer-add577a0b8c8

****