[BOOK AND MOVIE RECOMMENDATION SYSTEM]

GROUP NAME: - GROUP B

GROUP MEMBERS: -

First Name	Last Name	Student Number
Gurpreet Singh	Virdi	C0762178
Manoj	Moond	C0761024
Parth Dhirenkumar	Patel	C0765143
Priyanshi	Chakraborty	C0765384

Submission Date: -
August 25, 2020

Submitted To: -
William Pourmajidi

Table of Contents

Abstract: -

At present people use the Internet is the primary source of entertainment. Movies are the priority by the consumer as part of the enjoyment as a source of indoor enjoyment. From the past 2 decades, we can identify that most of the hit movies and series in the cinema are based upon books, these books have different types of genres such as horror, fantasy, thriller, fiction, biography, novels, and many more. A good example of these books-based movies is Harry Potter, Clash of the Titans, Hunger Games, etc. Through this project, we would like to introduce a platform for moviemakers to make good content movies. There is always a chance for doubt whether the new movies or serial would be perceived by the consumers or not.

This project is based on the movie and the book recommendation system. The movie recommendation system would help the movie house to recognize the movie content based upon the ratings received from the consumer. On the other hand, the book recommendation system would help to categorize the book genre and rating that is based upon the user s feedback. The rating would help movie maker to select the content what users are looking forward to.

Introduction: -

We have seen great triumph in the e-commerce-based businesses such as Amazon, Alibaba, eBay those are already running recommendation system. A Recommendation System framework is a type of filtering system that helps the organization to predict the user preference based upon the feedback received on an item. That furthermore helps to make improved decision making.

The recommendation system is widely implemented by organizations to improve the utilization of the resources. The touchy development in the measure of accessible computerized data and the number of guests to the Web has made a possible test of data over-burden, which blocks opportune access to things of enthusiasm on the Web. Due to the high volume of digital content on the Internet and drastic increase in the internet-dependent user, it has showen potential challenges for the users. The legacy systems failed to cope with the traffic demands and the other frameworks like devilfinder partially able to resolve the time access searches. The implementation of the recommendation framework is an absolute necessity to respond to the users' requests in a timely and efficient manner. This new system is completely different as compared to traditional systems that were previously used by the organizations.

This system helped to minimize the stress to examine the crucial data from the large data that is produced by the user's preferences and interests. It has the capability to predict if the user will select that item. It helps to refine the effective procedures and the quality for the e-commerce business by cutting off the cost and selections available. The e-commerce business with the help of recommendation system involvement has significantly enriched the profits for the businesses with convincing techniques to sell the items. The organizations like Netflix, Amazon Prime, Google, Amazon Music, Apple Music, Spotify, Amazon Prime they all uses the recommendation system to identify what type of item customer like to watch, listen, buy and their opinions about it? The recommendation system is esteemed for both users and organizations (book and movies) as it will provide opportunity for new writers and authors as a platform to express their talent.

Forms of Recommendation Systems: -

It is described as decision-making system that helps to provide the recommendation inputs given by the user that the system then aggregates and directs to applicable recipients. It will be additionally portrayed as a system that produces personalized recommendations as output or has the result of guiding the user in an exceedingly customized way to fascinating objects in an exceedingly more significant house of available choices. Recommender system can become associate degree integral part of the media and show business within the close to future. There are majorly six forms of recommendation systems as mentioned below: -

A. Collaborative: - It uses product to product matrix to overcome scalability problems.
B. Content-Based: - It uses customer information which helps to perform forecast.
C. Demographic Based: - It uses person to person preferences based upon region without collecting user historical profiles data.
D. Utility Based: - It is based upon the calculation of utility and also aspect non-product characteristics. That enables to check real time supplies and present to the user.
E. *Knowledge Based*: - It works on the base of knowledge about an item and notice how that item meets the user necessities
F. Hybrid: - It is basically mixture of any two above recommender systems to meet the business needs.

Content Based Filtering Form: -

After the careful understanding of the above describe forms of recommender systems we have selected content-based filtering form to apply on this project. There were several important aspects those are suitable for our project. Such as it highlights on the examination of the qualities in order to create forecast. Th content-based filtering examining depends upon the suggestion developed on the basis of users' substance of the things those were assessed earlier by the user requirements. The content-based filtering form uses few other types of other models as well to find out the resemblance between the user needs based upon historical data to suggest future items. To find out the similarity the content-based form could use these below mentioned models: -

A. Vector Space Model further divide into two categories TF/IDF (term frequency inverse document frequency) that helps to retrieval of the user information, which is implemented on this model or Probabilistic model.
B. Decision Tree or Neural network.

An excellent example of content-based filtering system in operational is LinkedIn, YouTube and Netflix platform. The content-based filtering works with the information that is provided by the user in terms of likeness, rating (explicitly) or clicking on the link (implicitly). With the more interaction of user, the content-based filtering form works better to generate new suggestions to customers.

Advantages and Disadvantages of Content Based Filtering: -

Advantages: -

A. Recommendation precision is not affected without the user inclinations.
B. System will still provide recommendation, if the user stopped giving feedbacks for example in form of ratings.
C. Content-based filtering has capability to adjust according to user penchant in same duration of time automatically.
D. It also provides privacy for the user in other term, it does not share user profiles with others.
E. Provides the feedback to the user why the system is providing this as recommendation. A good example of this is Netflix and YouTube let the user know why this movie or song is recommended like based on genre and rating.

Disadvantages: -

A. It is dependent on the item metadata and explanatory information of item.
B. The item information has to be well structured before the recommendation starts operational.

STAGES OF RECOMMENDATION PROCESS: -

A. INFORMATION COLLECTION PHASE: -
This first stage includes the gathering of user's profile characteristics and feedbacks. It is mandatory to well-devised the profile to make the recommendation functional, which also helps to avoid mistakes and save time. As always, it is maintaining the integrity of the proof is of prevailing importance through the method.

B. LEARNING PHASE: -
The second stage includes the educational system algorithmic rule is applied to make sure the model filters the user attributes from the findings of earlier stage.

C. PREDICTION/RECOMMENDATION STAGE: -
This is the last stage of the process provides the suggestion based upon the information gathered in the stage 1 (data information) or it could suggest based upon the user's activity (browsing, watch history or clicking on the links).

Feedback: -

There are three types of input on which the recommender system depends upon:

A. Explicit Feedback: -
The recommendation system framework prompts the user towards the system interface to submit evaluations to things to improve and acquire the model. The evaluation submitted by the user solely impacts the user proposal. The main drawback of this method is it requires feedback in most of the scenarios which are specifically given by the user and most of the time the clients are not always ready or willing to provide the information. Although input from the user is required in large flow it can be seen that erasing the tendencies from events furnishes the quality and trust in the proposals.

B. Implicit Feedback: -
This framework consequently convinces user tendencies by noticing the multiple factors of the user such as purchase history, route history, duration of time spent over the page, and many more. This framework minimizes stress by understanding user partialities. The framework technique does not require much action from the user and accuracy is quite less as well. It can be understood inclination data might objective as it is no predisposition emerging from clients reacting in a socially attractive manner.

C. Hybrid Feedback: -
The hybrid feedback is the consolidated qualities version of both Explicit and Implicit feedback. To limit the shortcoming and get a better execution framework. This can be done by utilizing and understanding the information as a be careful of explicit rating or permitting a user to give express criticism just when he decides to communicate explicit intrigue.

BOOK RECOMMENDATION SYSTEM: -

The Content-Based Filtering framework is implemented in the book recommendation system that suggest similar items to the user. It considers the user history and profile to provide suggestions. For example: If the user likes “The Hunger Games” then the framework will suggest user to read “The Hunger Games: Catching Fire”, “The Hunger Games: Mockingjay Part 1-2”, based upon user review, rating and genre.

We are utilizing suggestion information and don't have clients understanding history. Henceforth, we have utilized a basic substance-based suggestion framework using goodread.com datasets. We are going to assemble two suggestion frameworks by utilizing title and portrayal of the book. The similarity measures were used to find out the equivalent books, those will be used to assign later to the user’s as a suggestion. In this project the Cosine Similarity was in placed in our framework to find similarity and provide recommendation of books.

```
In [13]: # Importing necessary libraries
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import RegexpTokenizer
import re
import string
import random
from PIL import Image
import requests
from io import BytesIO
import matplotlib.pyplot as plt
%matplotlib inline

In [14]: # Reading the file
df = pd.read_csv("goodreads.csv", encoding = "ISO-8859-1", engine='python')
```

```
In [15]: #Reading the first five records
df.head()
```

Out[15]:

	Id	title	genre	Unnamed: 3	authors	Rating	Desc	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	1	Harry Potter and the Half-Blood Prince (Harry ...	Fantasy Fiction	NaN	J.K. Rowling	4.57	The war against Voldemort is not going well; e...	NaN	NaN	NaN
1	2	Harry Potter and the Order of the Phoenix (Har...	Fantasy Fiction	NaN	J.K. Rowling	4.50	There is a door at the end of a silent corrido...	NaN	NaN	NaN
2	3	Harry Potter and the Sorcerer's Stone (Harry P...	Fantasy Fiction	NaN	J.K. Rowling	4.47	Harry□s perfectly normal life at number 4 priv...	NaN	NaN	NaN
3	4	Harry Potter and the Chamber of Secrets (Harry...	Fantasy Fiction	NaN	J.K. Rowling	4.42	Harry Potter is about to start his second year...	NaN	NaN	NaN
4	5	Harry Potter and the Prisoner of Azkaban (Harr...	Fantasy Fiction	NaN	J.K. Rowling	4.57	Harry Potter and the Prisoner of Azkaban is th...	NaN	NaN	NaN

```
In [16]: #Checking the shape of the file
df.shape
Out[16]: (18, 10)
```

Figure 1

The above figure 1 is the first step of creating book recommendation system was to import the required libraries and as the first step of “DISCOVER” we have obtained the data using panda’s read func(). It also includes gathering data, cleaning data, exploring data and baseline outcome and assumption the solution.

```
In [15]: #Reading the first five records
df.head()
```

Out[15]:

	Id	title	genre	Unnamed: 3	authors	Rating	Desc	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	1	Harry Potter and the Half-Blood Prince (Harry ...	Fantasy Fiction	NaN	J.K. Rowling	4.57	The war against Voldemort is not going well; e...	NaN	NaN	NaN
1	2	Harry Potter and the Order of the Phoenix (Har...	Fantasy Fiction	NaN	J.K. Rowling	4.50	There is a door at the end of a silent corrido...	NaN	NaN	NaN
2	3	Harry Potter and the Sorcerer's Stone (Harry P...	Fantasy Fiction	NaN	J.K. Rowling	4.47	Harry□s perfectly normal life at number 4 priv...	NaN	NaN	NaN
3	4	Harry Potter and the Chamber of Secrets (Harry...	Fantasy Fiction	NaN	J.K. Rowling	4.42	Harry Potter is about to start his second year...	NaN	NaN	NaN
4	5	Harry Potter and the Prisoner of Azkaban (Harr...	Fantasy Fiction	NaN	J.K. Rowling	4.57	Harry Potter and the Prisoner of Azkaban is th...	NaN	NaN	NaN

```
In [16]: #Checking the shape of the file
df.shape
Out[16]: (18, 10)
```

Figure 2

Due to the processing restrictions, we have used only 18 rows of data, which would be sufficient for a prototype data model. The cleaning of the data for such small dataset was done by hardwiring.

```
In [17]: #Exploratory Data Analysis

In [18]: # Genre distribution
         df['genre'].value_counts().plot(x = 'genre', y ='count', kind = 'bar', figsize = (10,5)  )

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x228da9ce488>
```

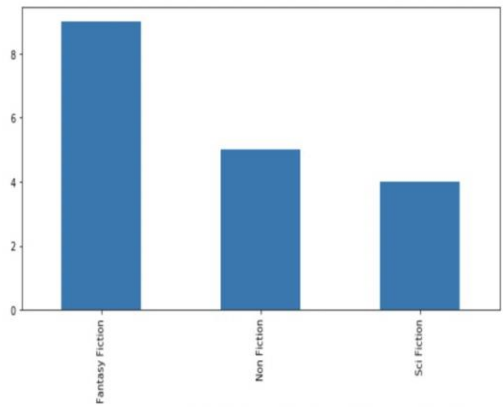

Figure 3

The above figure 3 shows that the dataset includes 3 genres those are fiction, non-fiction and sci-fiction.

```
In [64]: #Book description – Word count distribution

In [107]: # Calculating the word count for book description
          df['word_count'] = df['Desc'].apply(lambda x: len(str(x).split()))
          # Plotting the word count
          df['word_count'].plot(
              kind='hist',
              bins = 50,
              figsize = (12,8),title='Word Count Distribution for book descriptions')

Out[107]: <matplotlib.axes._subplots.AxesSubplot at 0x228e20bf608>
```

Figure 4

In the figure 4 we run a test to see the description of the word count. On that basis we plot a graph.

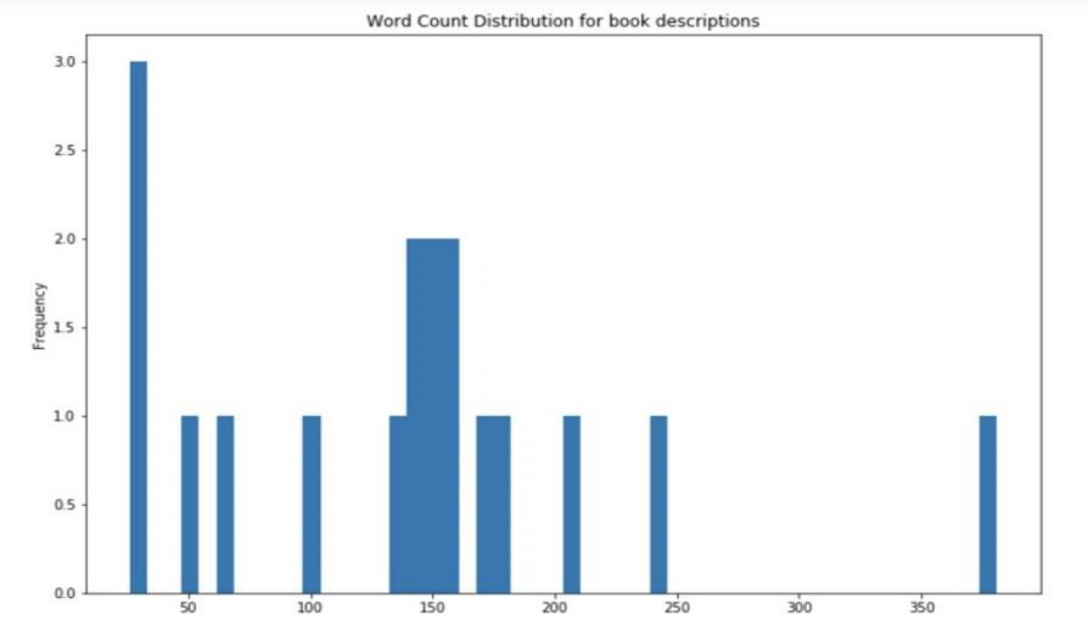

Figure 5

(NLP) Natural Language Processing: -

Natural Language Processing is part of Artificial Intelligence (AI). The NLP has been around for more than 5 decades in the computer industry. NLP is the human-made brainpower that manages the collaboration among PCs and users utilizing the analysis for a sentence, word segmentation (dividing a large piece of text to units), speak understanding, sentence breaking (sentence edges in large text). The main important objective of NLP is to examine, decode, grasp, and understand human languages.

There are various important functions of NLP in our daily life, few are described below: -

A. Grammarly and Google docs applications that engage Natural Language Processing to verify the precision of the writing.
B. Interactive Voice Response (IVR) is the telephony system that interacts with the user and transfers the calls to suitable receivers. It is widely used in the Banking call centers and Tax departments in the developed countries.
C. For traveller it is useful to take benefits of Google translate that help to translate almost all the languages around the world.
D. Google, Alexa, Siri, and Cortana personal assistant vastly deployed on all the latest technologies also uses NLP to translate the user's request.

We have used library for Natural Language Processing in the Python is NLTK well known as Natural Language Toolkit. It is widely used by most advanced developers around the world. The other library we have reader NLTK Corpus whose function is read corpus file which included a huge set of plain structured texts.

Figure 6

```
In [66]: #import nltk
         #nltk.download()
         #It will open a page for corpus download.Download all the packages

In [67]: #The distribution of top part-of-speech tags in the book descriptions
         from textblob import TextBlob
         blob = TextBlob(str(df['Desc']))
         pos_df = pd.DataFrame(blob.tags, columns = ['word' , 'pos'])
         pos_df = pos_df.pos.value_counts()[:20]
         pos_df.plot(kind = 'bar', figsize=(10, 8), title = "Top 20 Part-of-speech tagging for comments")

Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x228e15f6ac8>
```

Figure 7

The figure 5 gave us an estimate about the word count. We installed the NLTK packages that helps to match the word with the corpus library that can be find in the above figure 6 - 7.

```
In [68]: #Converting text descriptions into vectors using TF-IDF using Bigram
         tf = TfidfVectorizer(ngram_range=(2, 2), stop_words='english', lowercase = False)
         tfidf_matrix = tf.fit_transform(df['Desc'])
         total_words = tfidf_matrix.sum(axis=0)
         #Finding the word frequency
         freq = [(word, total_words[0, idx]) for word, idx in tf.vocabulary_.items()]
         freq =sorted(freq, key = lambda x: x[1], reverse=True)
         #converting into dataframe
         bigram = pd.DataFrame(freq)
         bigram.rename(columns = {0:'bigram', 1: 'count'}, inplace = True)
         #Taking first 20 records
         bigram = bigram.head(20)

In [69]: #Plotting the bigram distribution
         bigram.plot(x ='bigram', y='count', kind = 'bar', title = "Bigram disribution for the top 20 words in the book description", figs

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x228e1c601c8>
```

Figure 8

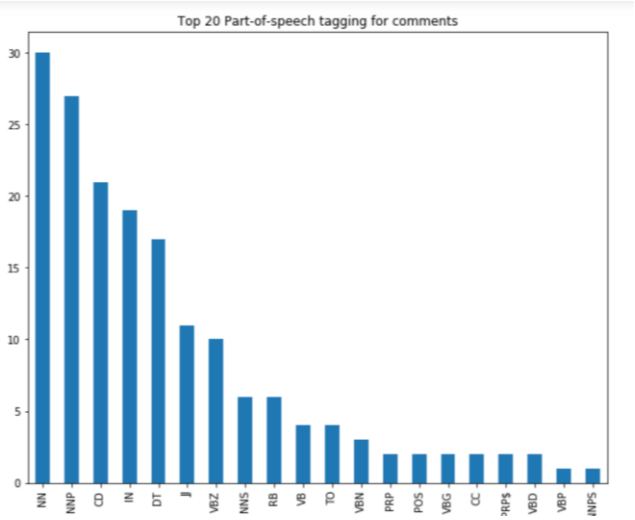

Figure 9 (used the function TextBlob to find Top 20 speech tags)

The machine learning processes does not function with raw text of data, but instead it converts the text into vector of numbers format. Term Frequency -Inverse Document Frequency (TF/IDF) helps to understand what the frequency of repetition of that word is and how important is that word for that document or sentence.

According to (Karbhari, 2019) from the Medium.com below is the formula to calculate TF/IDF.
TF(t) = (number of times term t appears in a document)/ (total number of terms in the document) (Karbhari, 2019)
IDF(t) = log10(Total number of documents/ number of documents with term t in it) (Karbhari, 2019)

The final formula that filters is TF*IDF.

Based on the findings we made, we know we have to use content and title of books for recommendation, thus we vectorize the desc using n- gram formula, we check both bi-gram and tri-gram using TF-IDF.

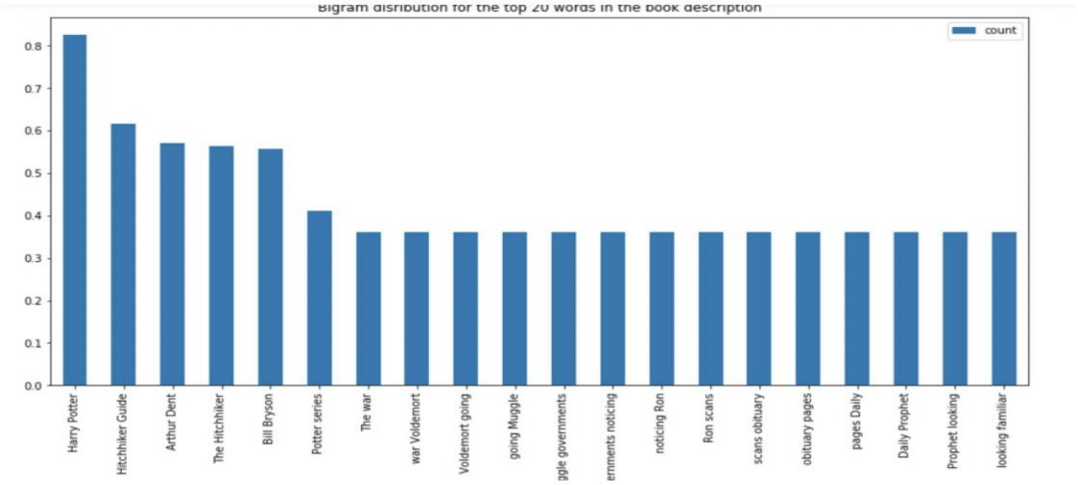

Figure 10 (shows the couple of words with the highest frequency)

```
In [70]: #Converting text descriptions into vectors using TF-IDF using Trigram
         tf = TfidfVectorizer(ngram_range=(3, 3), stop_words='english', lowercase = False)
         tfidf_matrix = tf.fit_transform(df['Desc'])
         total_words = tfidf_matrix.sum(axis=0)
         #Finding the word frequency
         freq = [(word, total_words[0, idx]) for word, idx in tf.vocabulary_.items()]
         freq =sorted(freq, key = lambda x: x[1], reverse=True)
         #converting into dataframe
         trigram = pd.DataFrame(freq)
         trigram.rename(columns = {0:'trigram', 1: 'count'}, inplace = True)
         #Taking first 20 records
         trigram = trigram.head(20)
         #Plotting the trigramn distribution
         trigram.plot(x ='trigram', y='count', kind = 'bar', title = "Trigram disribution for the top 20 words in the book description", f

Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x228e1c86648>
```

Figure 11

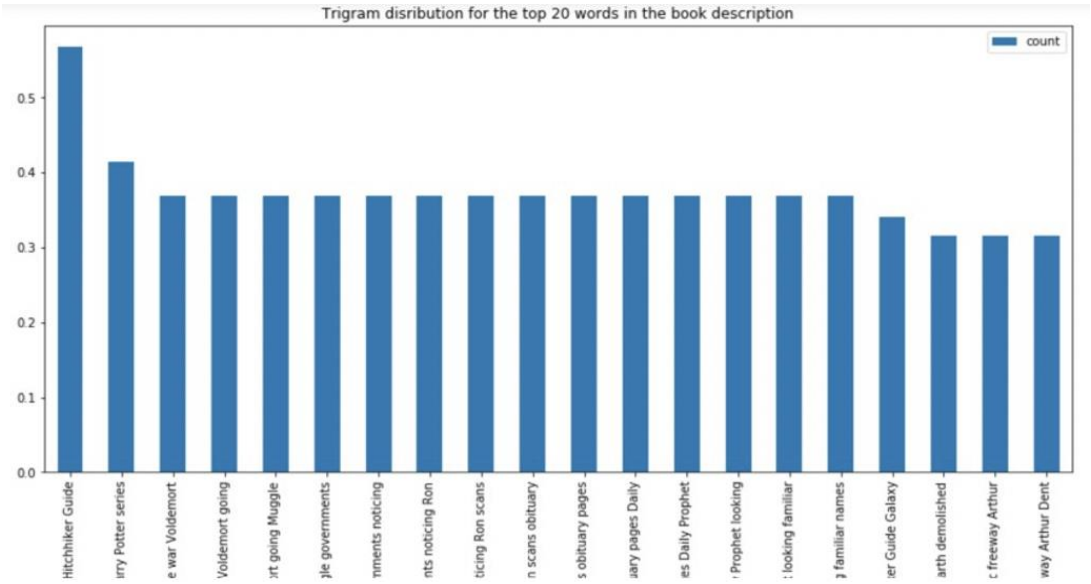

Figure 12

Based on the Trigram, we see triplets words with highest frequency in the whole dataset within this graph.

```
In [71]: #Text Preprocessing
         #Cleaning the book description.
         # Function for removing NonAscii characters
         def _removeNonAscii(s):
             return "".join(i for i in s if  ord(i)<128)
         # Function for converting into lower case
         def make_lower_case(text):
             return text.lower()
         # Function for removing stop words
         def remove_stop_words(text):
             text = text.split()
             stops = set(stopwords.words("english"))
             text = [w for w in text if not w in stops]
             text = " ".join(text)
             return text
         # Function for removing punctuation
         def remove_punctuation(text):
             tokenizer = RegexpTokenizer(r'\w+')
             text = tokenizer.tokenize(text)
             text = " ".join(text)
             return text
         #Function for removing the html tags
         def remove_html(text):
             html_pattern = re.compile('<.*?>')
             return html_pattern.sub(r'', text)
         # Applying all the functions in description and storing as a cleaned_desc
         df['cleaned_desc'] = df['Desc'].apply(_removeNonAscii)
         df['cleaned_desc'] = df.cleaned_desc.apply(func = make_lower_case)
         df['cleaned_desc'] = df.cleaned_desc.apply(func = remove_stop_words)
```

Figure 13

Since, the description, is written by different people, before we apply it to the algorithm it needs to be cleaned, thus we remove, all upper case, punctuation marks, stop words and html from the description. Now applying the cleaned data to below model where we used vectors and apply cosine similarity function to find the similarities, between two titles and two descriptions.

Recommendation Engine: -

We have developed book title and description recommendation model, which uses vectors TF-IDF and bi-gram. The model suggests a comparable book dependent on title and description. Using cosine similarity to determine likeness between books.

Cosine Similarity: -

While working on natural language processing (NLP) in this project and other module in this semester (Data Mining and Analysis) we have learned about the Cosine Similarity. The Cosine Similarity metric is used to determine the similarity between the two documents, word or in terms to calculate the cosine angle between the two vectors in several direction. The purpose of using cosine similarity is to over the Euclidean distance problem, which means comparing the common word between two documents regardless the size of the document (as the documents size increases the number of common word count goes up). The two vectors mean the array word counts of both documents or sentence.

Cosine similarity is different than the number of common words appear in the two documents. The reason is when they are drawn in the multi-directional the path relates to the words present in the document. Over here the cosine similarity does not apprehend the magnitude but uses the angle of file. The magnitude is calculated by the Euclidean Distance. The benefit of implementing cosine similarity is if the

two documents or books are at the Euclidean distance (distant) the similarity of word is 20:100 the cosine angle will be small. It means smaller the angle greater is the similarity.

```
In [72]: from sklearn.metrics.pairwise import cosine_similarity

In [73]: #Recommendation based on book title
         # Function for recommending books based on Book title. It takes book title and genre as an input.
         def recommend(title, genre):

             # Matching the genre with the dataset and reset the index
             data = df.loc[df['genre'] == genre]
             data.reset_index(level = 0, inplace = True)

             # Convert the index into series
             indices = pd.Series(data.index, index = data['title'])

             #Converting the book title into vectors and used bigram
             tf = TfidfVectorizer(analyzer='word', ngram_range=(2, 2), min_df = 1, stop_words='english')
             tfidf_matrix = tf.fit_transform(data['title'])

             # Calculating the similarity measures based on Cosine Similarity
             sg = cosine_similarity(tfidf_matrix, tfidf_matrix)

             # Get the index corresponding to original_title

             idx = indices[title]
         # Get the pairwsie similarity scores
             sig = list(enumerate(sg[idx]))
         # Sort the books
             sig = sorted(sig, key=lambda x: x[1], reverse=True)
         # Scores of the 5 most similar books
```

Figure 14

```
         # Scores of the 5 most similar books
             sig = sig[1:6]
         # Book indicies
             movie_indices = [i[0] for i in sig]

             # Top 5 book recommendation
             rec = data[['title', 'url']].iloc[movie_indices]

             # It reads the top 5 recommend book url and print the images

             for i in rec['url']:
                 response = requests.get(i)
                 img = Image.open(BytesIO(response.content))
                 plt.figure()
                 print(plt.imshow(img))

In [80]: recommend("In a Sunburned Country", "Non Fiction")
```

Figure 15

Here we recommend, book by title, thus we gave input of “In a sunburned country” which is a non - fiction and authored by “Bill Bryson” and as in output we can see that it recommended another book which is a non-fiction and written by the same author.

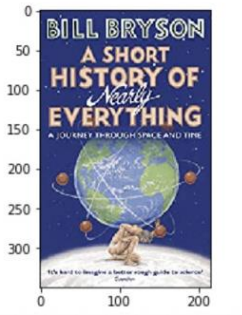

Figure 16

```
In [103]: #Recommendation based on book description
#We are using the same above function by converting book description into vectors.
# Function for recommending books based on Book title. It takes book title and genre as an input.
def recommend(title, genre):

    global rec
    # Matching the genre with the dataset and reset the index
    data = df.loc[df['genre'] == genre]
    data.reset_index(level = 0, inplace = True)
# Convert the index into series
    indices = pd.Series(data.index, index = data['title'])
    #Converting the book description into vectors and used bigram
    tf = TfidfVectorizer(analyzer='word', ngram_range=(2, 2), min_df = 1, stop_words='english')
    tfidf_matrix = tf.fit_transform(data['cleaned_desc'])
     # Calculating the similarity measures based on Cosine Similarity
    sg = cosine_similarity(tfidf_matrix, tfidf_matrix)
    # Get the index corresponding to original_title

    idx = indices[title]
    # Get the pairwsie similarity scores
    sig = list(enumerate(sg[idx]))
    # Sort the books
    sig = sorted(sig, key=lambda x: x[1], reverse=True)
# Scores of the 5 most similar books
    sig = sig[1:6]
# Book indicies
    movie_indices = [i[0] for i in sig]
```

Figure 17

```
    # Top 5 book recommendation
    rec = data[['title']].iloc[movie_indices]

    # It reads the top 5 recommend book url and print the images

    for i in rec['url']:
        response = requests.get(i)
        img = Image.open(BytesIO(response.content))
        plt.figure()
        print(plt.imshow(img))

In [105]: recommend("Harry Potter and the Prisoner of Azkaban","Fantasy Fiction")
```

Figure 18

Here we recommend, book by desc, thus we gave input of “Harry Potter and Prisoner of Azkaban” which is a fantasy fiction and as in output we can see that it recommends all other books in “Harry Potter series”.

Result 1: -

```
Harry Potter and the Goblet of Fire
Harry Potter and the Deathly Hallows
Harry Potter and the Chamber of Secrets
Harry Potter and the Sorcerer's Stone
Harry Potter and the Order of the Phoenix
```

Figure 19

As the part discovering we discovered that this system can work on big data if it is implemented on cloud. Due to processing restrictions we were only able to make a Prototype based model and thus, cannot be deployed at this moment.

Movie Recommendation System: -

Importing the Libraries: -

```python
# Libraries
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from ast import literal_eval
```

Figure 20

Loading the Data: -

```python
# Dataset
data = pd.read_csv("C:/Users/Vinay Moond/PycharmProjects/TermProject/movies_metadata.csv", low_memory=False)

#top rows of data
print('Top Rows of Data: ','\n', data.head())
```

Figure 21

Weighted Rating Function: -

```
# Mean of average vote
mean_vote = data['vote_average'].mean()
print('The mean vote across the whole report: ', mean_vote)

# Minimun vote required
min_vote = data['vote_count'].quantile(0.90)
print('The minimum votes required to be listed in the chart: ',min_vote)

# new dataset with qualified movies
movies = data.copy().loc[data['vote_count'] >= min_vote]
print('Size of Movies dataset: ','\n',movies.shape)
print('Size of Original dataset: ','\n',data.shape)

#Weighted Rating Fumction
def WR_function(x, min_vote=min_vote, mean_vote=mean_vote):
    vote_count = x['vote_count']
    vote_avg = x['vote_average']
    return (vote_count/(vote_count+min_vote) * vote_avg) + (min_vote/(min_vote+vote_count) * mean_vote)
```

Figure 22

In this equation,

- Vote_count is no. of votes for the movie.
- Min_vote is the min votes required to be in the chart.
- Vote_avg is the average rating of the movie.
- Mean_vote is the mean vote across the whole dataset.

```
# New column defing the weighted rating
movies['WR'] = movies.apply(WR_function, axis=1)
print('Top Rows of Data: ','\n',movies.head())

#Sorting according to WR
movies = movies.sort_values('WR', ascending=False)

#top 15 movies
print('Top 15 Rows of Data: ','\n',movies[['title', 'vote_count', 'vote_average', 'WR']].head(15))

#overview from data
print('Overview from Data: ','\n',movies['overview'].head())
```

Figure 23

Term Frequency–Inverse Document Frequency: -

```
#Replacing NaN with an empty string
movies['overview'] = movies['overview'].fillna('')

#term frequency-inverse document frequency vectorizer object
tfidf = TfidfVectorizer(stop_words='english')

#term frequency-inverse document frequency matrix
t_matrix = tfidf.fit_transform(movies['overview'])
print('Term Frequency-Inverse Document Frequency Matrix: ','\n',t_matrix)
```

Figure 24

Cosine Similarity Matrix: -

```
#cosine similarity matrix
c_matrix = linear_kernel(t_matrix, t_matrix)
print('Cosine Similarity Matrix: ','\n',c_matrix)
```

Figure 25

Recommendation Function: -

```
#reverse map of indices and movie titles
indices = pd.Series(movies.index, index=movies['title']).drop_duplicates()
print('Reverse Map of Indices and Movie Titles: ','\n',indices[:10])

# Recommendation function
def recommendations(title, c_matrix=c_matrix):
    index = indices[title]

    similarity_scores = list(enumerate(c_matrix[index]))

    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    similarity_scores = similarity_scores[1:11]

    movies_title = [i[0] for i in similarity_scores]

    return movies['title'].iloc[movies_title]
```

Figure 26

Result 2: -

```
#trying if recommendation system works
print('Recommendations based on the movie are: ','\n',recommendations('Pulp Fiction'))
```

Figure 27

```
..
Recommendations based on the movie are:
 1983                      The Return of Jafar
1034          Aladdin and the King of Thieves
28929                           A Little Chaos
18832                            Mirror Mirror
21349                             Blue Jasmine
15265    Prince of Persia: The Sands of Time
160                                  Desperado
3524                       Big Momma's House
23110                  The Legend of Hercules
1987                          Sleeping Beauty
Name: title, dtype: object
```

Figure 28

Conclusion: -

This project was developed with both the books and movie recommendation framework. This project will help in providing the users by providing them the suggestion for the movies and the books. These suggestions will be based upon the title and genre of the movies and books. We just used small part from the datasets due to limited operational resources. This model is prototype and was not pushed to cloud platform due to lack of finances. However, all the group members enjoyed it through the remote working and the most important things is we have learned a lot about the recommendation system.

Future Possibilities: -

As examined before,

A. This project has the wide scope of implementation on the industrial level, once we moved to cloud environment.
B. Both the books and movie recommendation can be combined together and uploaded to cloud to work as industrial model.
C. The movie proposal will help movie maker to observe the trend based upon the genre through customer preferences
D. The book recommendation also helps authors to appraisals to help film producers settle on the content.

References: -

- DAS, S. (2015, August 11). *www.analyticsvidhya.com/.* Retrieved from www.analyticsvidhya.com/: https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/
- F.O.Isinkayea, Y. B. (2015, November). *www.sciencedirect.com/.* Retrieved from www.sciencedirect.com/: https://www.sciencedirect.com/science/article/pii/S1110866515000341
- Grimaldi, E. (2018, October 01). *towardsdatascience.com/*. Retrieved from towardsdatascience.com/: https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243
- Gupta, S. (2018, May 15). *towardsdatascience.com/*. Retrieved from towardsdatascience.com/: https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50
- Jiawei Han, J. P. (2012). *www.sciencedirect.com.* Retrieved from www.sciencedirect.com: https://www.sciencedirect.com/topics/computer-science/cosine-similarity
- Karbhari, V. (2019, july 9). *medium.com.* Retrieved from medium.com: https://medium.com/acing-ai/what-is-tf-idf-in-feature-engineering-7f1ba81982bd
- kdnuggets.com. (2019, September). *https://www.kdnuggets.com*. Retrieved from https://www.kdnuggets.com: https://www.kdnuggets.com/2019/09/machine-learning-recommender-systems.html
- Prabhakaran, S. (n.d.). *www.machinelearningplus.com/*. Retrieved from www.machinelearningplus.com/: https://www.machinelearningplus.com/nlp/cosine-similarity/
- Rocca, B. (2019, June 02). *https://towardsdatascience.com*. Retrieved from https://towardsdatascience.com: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada