CONTENT-BASED MOVIE RECOMMENDATION SYSTEM

A Mini Project Report Submitted by

NAMRATHA (4NM18CS099)

NIDHI RAI (4NM18CS103)

UNDER THE GUIDANCE OF

Mrs.Shabari Shedthi B

Assistant Professor

Department of Computer Science and Engineering

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Engineering in
Computer Science & Engineering

from

Visveshvaraya Technological University, Belgaum

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

N.M.A.M. INSTITUTE OF TECHNOLOGY

(An Autonomous Institution under VTU, Belgaum) (AICTE approved, NBA Accredited, ISO 9001:2008 Certified) NITTE -574 110, Udupi District, KARNATAKA.

May 2021

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
☎: 08258 - 281039 – 281263, Fax: 08258 – 281265
Department of Computer Science and Engineering
B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

"Content-Based Movie Recommendation system"

is a bonafide work carried out by

NAMRATHA(4NM18CS099) NIDHI RAI(4NM18CS103)

in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belgaum during the year 2020-2021.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report.

The Mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide Signature of HOD

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to make this project successful.

First and foremost, our sincere thanks to our beloved principal, **Dr. Niranjan N. Chiplunkar** for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.

We express our deep sense of gratitude and indebtedness to our guide **Mrs.Shabari Shedthi B**, Assistant Professor, Department of Computer Science and Engineering, for her inspiring guidance, constant encouragement, support and suggestions for improvement during the course of our project.

We sincerely thank **Dr.Jyothi Shetty**, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyanthaya Memorial Institute of Technology, Nitte.

We also thank all those who have supported us throughout the entire duration of our project.

Finally, we thank the staff members of the Department of Computer Science and Engineering and all our friends for their honest opinions and suggestions throughout the course of our project.

NAMRATHA(4NM18CS099)
NIDHI RAI(4NM18CS103)

ABSTRACT

Recommender System is a tool helping users find content and overcome information overload. It predicts interests of users and makes recommendation according to the interest model of users. They recommend an item to a user based upon a description of the item and a profile of the user's interests.

Our recommendation engine filters the data using certain algorithms and recommends the most relevant items to users. They offer generalized recommendations to every user, based on certain features of the movie which the user has previously watched and liked. The System recommends the similar movies to users according to their interest.

There are various kinds of recommendation engines. Firstly, the popularity-based recommendation system which uses the basic idea that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience and recommends those movies to the users. Second is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user. Lastly the system we will be implementing is the content-based filtering, where we try to profile the user's interests using information collected, and recommend items similar to that item.

Our recommender system uses a simple algorithm whose aim is to provide the most similar movie depending on various features to a user by discovering patterns in a dataset. The algorithm calculates the similarity of the movie provided with all the movies present in the dataset and recommends a similar movie depending on the content of the movie. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommend music that you might like.

TABLE OF CONTENTS

INTRODUCTION

Content-based recommendation systems may be used in a variety of domains ranging from recommending web pages, news articles, restaurants, television programs, and items for sale. Although the details of various systems differ, content-based recommendation systems share in common a means for describing the items that may be recommended, a means for creating a profile of the user that describes the types of items the user likes, and a means of comparing items to the user profile to determine what to recommend. The profile is often created and updated automatically in response to feedback on the desirability of items that have been presented to the user.

Content-based systems use metadata such as genre, producer, actor, musician to recommend items say movies or music. Such a recommendation would be for instance recommending Infinity War that featured Vin Diesel because someone watched and liked The Fate of the Furious. Similarly, you can get music recommendations from certain artists because you liked their music. Content-based systems are based on the idea that if you liked a certain item you are most likely to like something that is similar to it.

Content-based recommendation systems try to recommend items similar to those a given user has liked in the past. Indeed, the basic process performed by a content-based recommender consists in matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of a content object (item), in order to recommend to the user new interesting items.

Content based recommendation engines collects data regarding the movie the user has watched and liked in the past and looks for similar movies from the collection of movies in the dataset and recommends most similar movies to the user which will be advantageous for the user to watch movies according to his interest rather than being confused about what to watch among a wide variety of new movies releasing worldwide everyday. Rather than asking people for recommendations regarding good movies the user can easily get recommendations from such movie recommendation engines.

LITERATURE SURVEY

- Kim Mucheol et al. [1] described an interactive RS. This paper projected a theoretical model and sample of the projected interactive movie RS. This proposed method constructs adapted suggestion of movies in online community systems. The presented method develops the grouping conscious community network model which is considered as the approach that is capable to confine the dynamics of socially-mediated information communicated in communal networks. This proposed replica may examine the fondness of user methodically and which can show the quick and constant change in social network.

- Sharma et al. [2] this paper reviewed the several approaches used for RS. Approaches may be categorized into three parts CF. Hybrid Recommendations and Content Based recommendation. Also this paper describes the merits and demerits of the recommendation approaches. Furthermore, problem occurs in RS also discussed in this paper.

- Rattanajit et al. [3] presented pseudo ratings which is relying on multi criteria and also focuses on the related information as multidimensional. The Naiumlve Bayes approach has been applied for doing the pseudo rating depending on multi criteria. This approach has been used to categorize the multi criterion of client's preferences. The multi regression is useful to examine the appropriate information of client in order to include multidimensional. According to the evaluation of experiment, the RS is created on movie domain known as Modernize Movie and demonstrates that the multi criterion pseudo ratings and multi-dimensional client report increase the worth and accurateness of recommender outcome.

- Christakou et al. [4] a clustering technique has been proposed which is depending on semi supervised learning. In this paper, presented approach is utilized to create a system for recommending movies which merge collaborative and content-based information. The presented system is checked on the Movie Lens DS, providing recommendation of high precision.

DESIGN

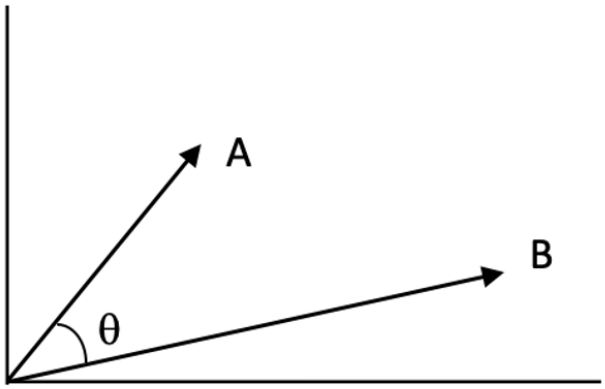
Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

What is cosine similarity?

Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between the two vectors.

Cosine similarity is one of the most widely used and powerful similarity measure in Data Science. It is used in multiple applications such as finding similar documents in NLP, information retrieval, finding similar sequence to a DNA in bioinformatics, detecting plagiarism and may more.

Cosine similarity is calculated as follows,

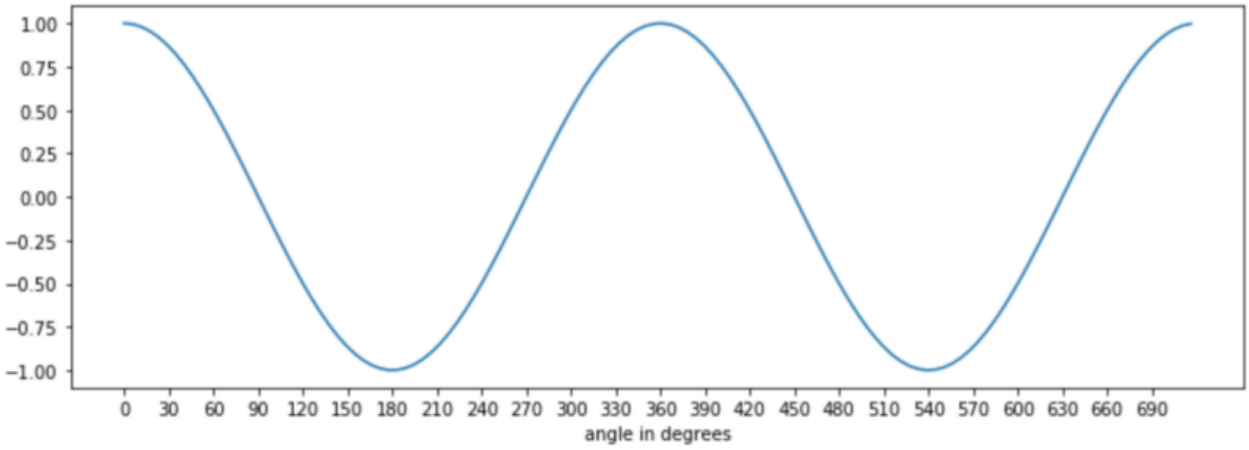


Angle between two 2-D vectors A and B

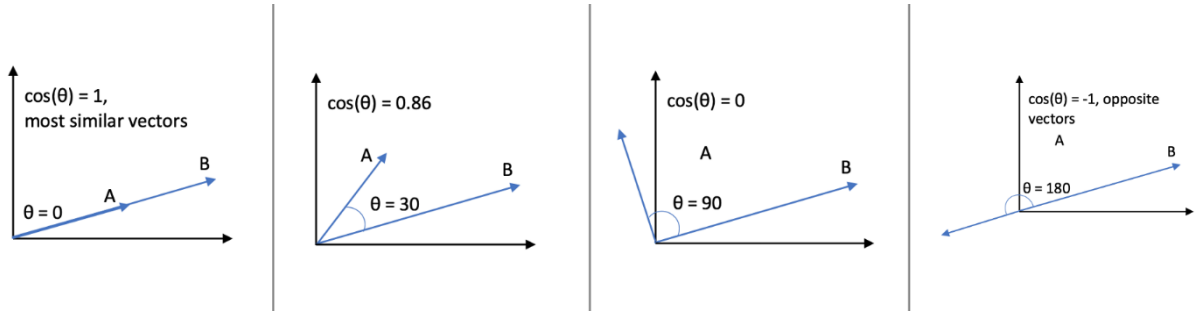
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Calculation of cosine of the angle between A and B

Why cosine of the angle between A and B gives us the similarity?
If you look at the cosine function, it is 1 at theta = 0 and -1 at theta = 180, that means for two overlapping vectors cosine will be the highest and lowest for two exactly opposite vectors. You can consider 1-cosine as distance.



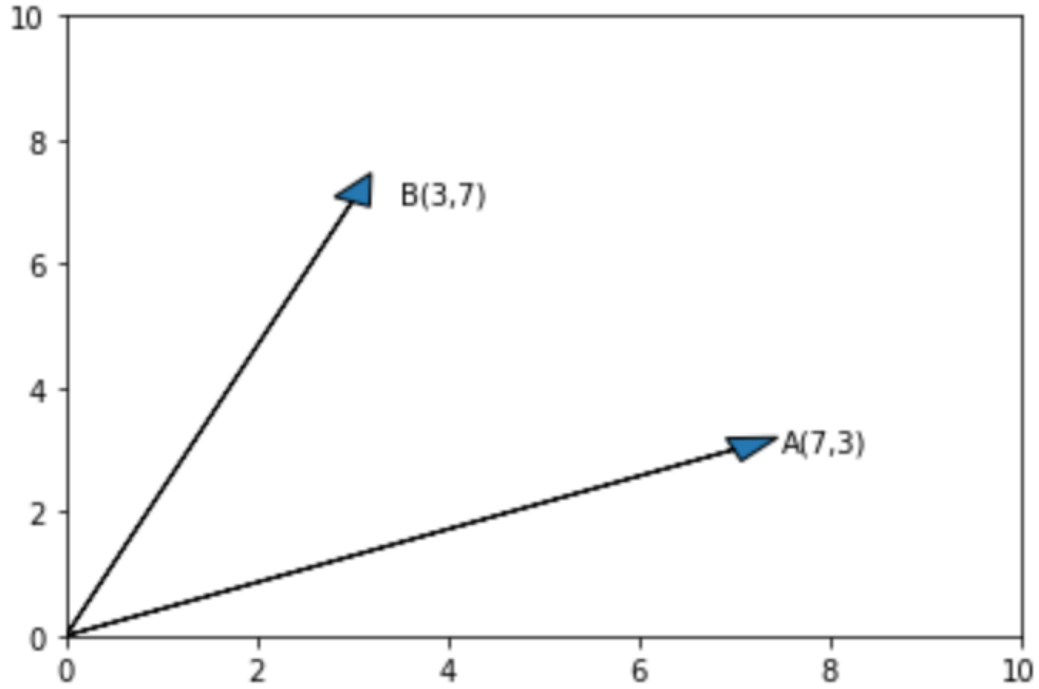
Cosine



Values of cosine at different angles

How to calculate it in Python?
The numerator of the formula is the dot product of the two vectors and denominator is the product of L2 norm of both the vectors. Dot product of two vectors is the sum of element wise multiplication of the vectors and L2 norm is the square root of sum of squares of elements of a vector.
We can either use inbuilt functions in Numpy library to calculate dot product and L2 norm of the vectors and put it in the formula or directly use the cosine_similarity from sklearn.metrics.pairwise.

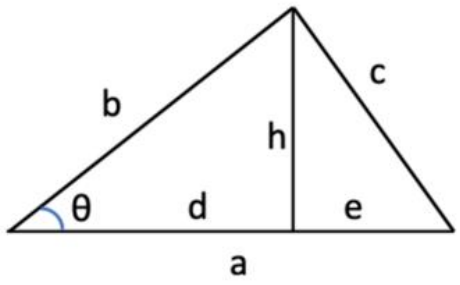
Consider two vectors A and B in 2-D, following code calculates the cosine similarity,




```
Cosine Similarity between A and B:0.7241379310344827
Cosine Distance between A and B:0.27586206896551735
```

Proof of the formula

Cosine similarity formula can be proved by using Law of cosines,



$$c^2 = b^2 + a^2 - 2abcos(\theta)$$

$cos(\theta) = d/b$
$d = bcos(\theta)$

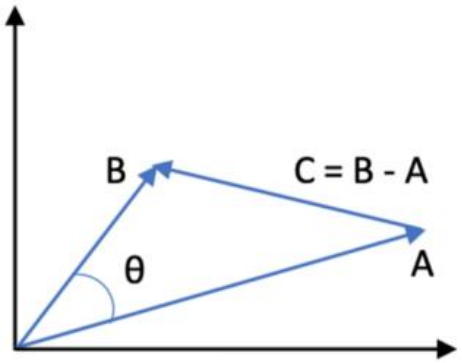
$sin(\theta) = h/b$
$h = bsin(\theta)$

$c^2 = h^2 + e^2$
$c^2 = (bsin(\theta))^2 + (a-d)^2$
$c^2 = (bsin(\theta))^2 + (a - bcos(\theta))^2$
$c^2 = b^2sin^2(\theta) + a^2 - 2abcos(\theta) + b^2cos^2(\theta)$
$c^2 = b^2(sin^2(\theta) + cos^2(\theta)) + a^2 - 2abcos(\theta)$

$c^2 = b^2 + a^2 - 2abcos(\theta)$

Law of cosines

Consider two vectors A and B in 2-dimensions, such as,



$$\vec{A} = \begin{matrix} a1 \\ a2 \end{matrix} \quad \vec{B} = \begin{matrix} b1 \\ b2 \end{matrix} \quad \vec{B} - \vec{A} = \begin{matrix} b1 - a1 \\ b2 - a2 \end{matrix}$$

$$\vec{A}.\vec{B} = a1.b1 + a2.b2$$

Length of vector $\vec{A} = ||\vec{A}|| = \sqrt[2]{a1^2 + a2^2}$

Two 2-D vectors

Using Law of cosines,

$$||\vec{B} - \vec{A}||^2 = ||\vec{B}||^2 + ||\vec{A}||^2 - 2||\vec{A}||.||\vec{B}||.\cos(\theta)$$

$$2||\vec{A}||.||\vec{B}||.\cos(\theta) = ||\vec{B}||^2 + ||\vec{A}||^2 - ||\vec{B} - \vec{A}||^2$$
$$2||\vec{A}||.||\vec{B}||.\cos(\theta) = b1^2 + b2^2 + a1^2 + a2^2 - ((b1 - a1)^2 + (b2 - a2)^2)$$
$$2||\vec{A}||.||\vec{B}||.\cos(\theta) = b1^2 + b2^2 + a1^2 + a2^2 - ((b1^2 - 2b1.a1 + a1^2) + (b2^2 - 2b2.a2 + a2^2))$$
$$2||\vec{A}||.||\vec{B}||.\cos(\theta) = 2(a1.b1 + a2.b2)$$

$$\cos(\theta) = \frac{a1.b1 + a2.b2}{||\vec{A}||.||\vec{B}||}$$

$$\cos(\theta) = \frac{\vec{A}.\vec{B}}{||\vec{A}||.||\vec{B}||}$$

Cosine similarity using Law of cosines

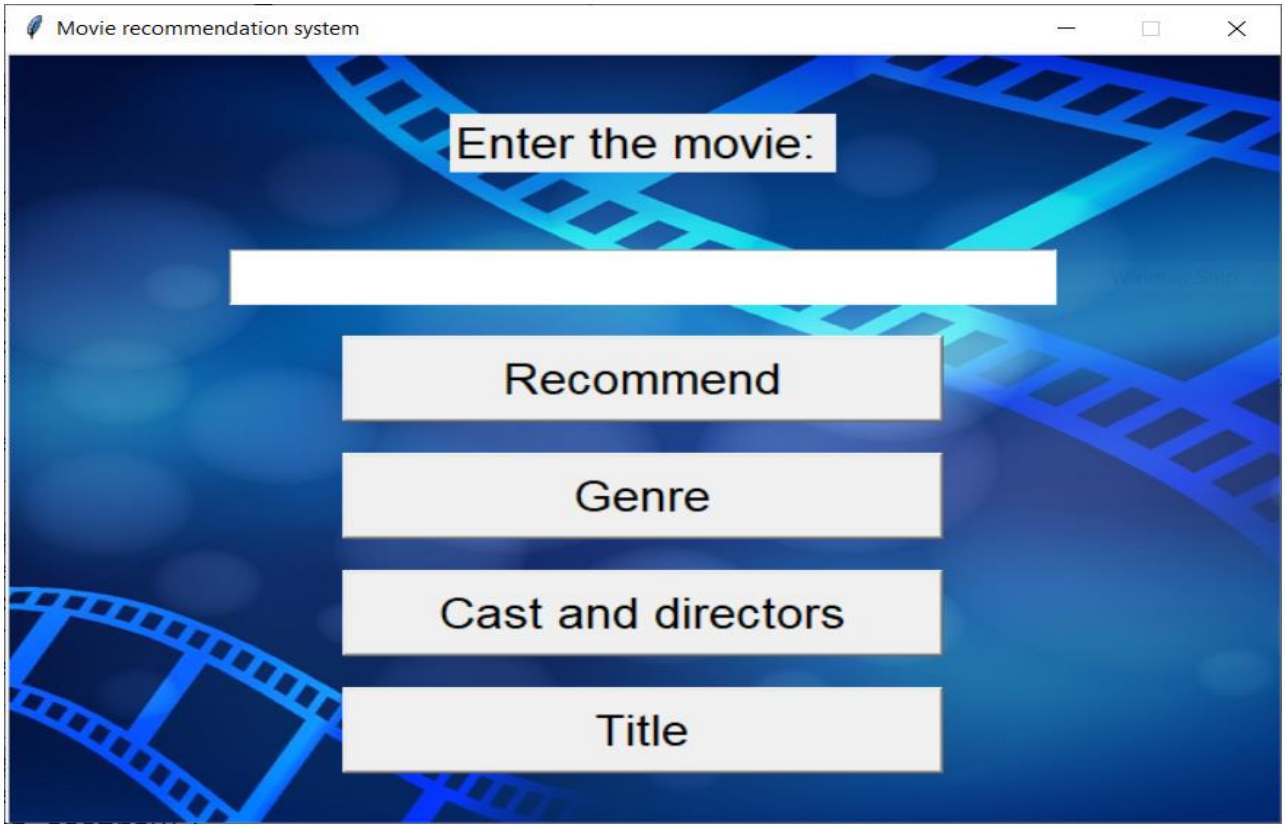
You can prove the same for 3-dimensions or any dimensions in general. It follows exactly same steps as above

IMPLEMENTATION

Importing Libraries

```
import tkinter
import pandas as pd
import numpy as np
import math
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

User Interface



We have developed UI using Tkinter library. Here we can enter the movie whichever is there in our dataset and click on any of the four buttons.

We have designed our GUI using tkinter library

```
result=""
general,genre,castdir,title=0,0,0,0
top = tkinter.Tk()
top.title("Movie recommendation system")
top.geometry("700x500")
top.resizable(False,False)
bg = tkinter.PhotoImage(file="backg.png")
my_label=tkinter.Label(top,image=bg)
my_label.place(x=0,y=0)
label=tkinter.Label(top,text="Enter the movie: ",font=("Helvetica",20),fg="black")
label.pack(pady=40)
```

```
def getTextInput():
    global result
    global general
    result=entry.get("1.0","end-1c")
    general = 1
    top.destroy()
    print(result)
def getGenreInput():
    global result
    global genre
    result=entry.get("1.0","end-1c")
    genre = 1
    top.destroy()
    print(result)
def getCastInput():
    global result
    global castdir
    result=entry.get("1.0","end-1c")
    castdir = 1
    top.destroy()
    print(result)
def getTitleInput():
    global result
    global title
    result=entry.get("1.0","end-1c")
    title = 1
    top.destroy()
    print(result)
entry=tkinter.Text(top,height=1,width=30,font=("Helvetica",20),fg="black")
entry.pack(pady=10)
button=tkinter.Button(top,text="Recommend",width=20,font=("Helvetica",20),fg="black",command=getTextInput)

button.pack(pady=10)
buttongenre = tkinter.Button(top,text="Genre",width=20,font=("Helvetica",20),fg="black",command=getGenreInput)
buttongenre.pack(pady=10)
buttoncast = tkinter.Button(top,text="Cast and directors",width=20,font=("Helvetica",20),fg="black",command=getCastInput)
buttoncast.pack(pady=10)
buttontitle = tkinter.Button(top,text="Title",width=20,font=("Helvetica",20),fg="black",command=getTitleInput)
buttontitle.pack(pady=10)
top.mainloop()
```

Reading the dataset

```
[4] df = pd.read_csv("movie_dataset.csv")
```

```
df.head()
```

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	production_coun
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "name": "United
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "name": "United
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": "name": " Kingc
						dc						

Basically our dataset has 24 columns but our similarity algorithm we don't require all the 24 columns. Because there are many columns which are numeric. So we choose 5 columns from the dataset as our feature which we will be using for the similarity algorithm.

```
[6] df.shape
```

```
(4803, 24)
```

```
[7] df.describe()
```

	index	budget	id	popularity	revenue	runtime	vote_average	vote_count
count	4803.000000	4.803000e+03	4803.000000	4803.000000	4.803000e+03	4801.000000	4803.000000	4803.000000
mean	2401.000000	2.904504e+07	57165.484281	21.492301	8.226064e+07	106.875859	6.092172	690.217989
std	1386.651002	4.072239e+07	88694.614033	31.816650	1.628571e+08	22.611935	1.194612	1234.585891
min	0.000000	0.000000e+00	5.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	1200.500000	7.900000e+05	9014.500000	4.668070	0.000000e+00	94.000000	5.600000	54.000000
50%	2401.000000	1.500000e+07	14629.000000	12.921594	1.917000e+07	103.000000	6.200000	235.000000
75%	3601.500000	4.000000e+07	58610.500000	28.313505	9.291719e+07	118.000000	6.800000	737.000000
max	4802.000000	3.800000e+08	459488.000000	875.581305	2.787965e+09	338.000000	10.000000	13752.000000

```
df.columns.values
```

```
array(['index', 'budget', 'genres', 'homepage', 'id', 'keywords',
       'original_language', 'original_title', 'overview', 'popularity',
       'production_companies', 'production_countries', 'release_date',
       'revenue', 'runtime', 'spoken_languages', 'status', 'tagline',
       'title', 'vote_average', 'vote_count', 'cast', 'crew', 'director'],
      dtype=object)
```

Checking for null values

```
df.isnull().sum()
```

```
index                      0
budget                     0
genres                    28
homepage                3091
id                         0
keywords                 412
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title                      0
vote_average               0
vote_count                 0
cast                      43
crew                       0
director                  30
dtype: int64
```

Now in those 5 columns we have to check if there are any null values.

Replacing null values with empty string

```
[10] features = ['genres', 'keywords', 'title', 'cast', 'director']
     for feature in features:
         df[feature] = df[feature].fillna('')
```

```
[12] df[features].head()
```

	genres	keywords	title	cast	director
0	Action Adventure Fantasy Science Fiction	culture clash future space war space colony so...	Avatar	Sam Worthington Zoe Saldana Sigourney Weaver S...	James Cameron
1	Adventure Fantasy Action	ocean drug abuse exotic island east india trad...	Pirates of the Caribbean: At World's End	Johnny Depp Orlando Bloom Keira Knightley Stel...	Gore Verbinski
2	Action Adventure Crime	spy based on novel secret agent sequel mi6	Spectre	Daniel Craig Christoph Waltz L\u00e9a Seydoux ...	Sam Mendes
3	Action Crime Drama Thriller	dc comics crime fighter terrorist secret ident...	The Dark Knight Rises	Christian Bale Michael Caine Gary Oldman Anne ...	Christopher Nolan
4	Action Adventure Science Fiction	based on novel mars medallion space travel pri...	John Carter	Taylor Kitsch Lynn Collins Samantha Morton Wil...	Andrew Stanton

```
df[features].isnull().sum()
```

```
genres      0
keywords    0
title       0
cast        0
director    0
dtype: int64
```

We don’t require null values. So we replace all the null values with empty strings.

Combining into a single column

```
def combine_features(row):
    return row['title']+' '+row['genres']+' '+row['director']+' '+row['keywords']+' '+row['cast']
def combine_genrefeatures(row):
    return row['genres']+' '+row['keywords']
def combine_castfeatures(row):
    return row['director']+' '+row['cast']
def combine_titlefeatures(row):
    return row['title']+' '+row['keywords']

cv = CountVectorizer()
if general == 1:
    df['combined_features'] = df.apply(combine_features, axis = 1)
    count_matrix = cv.fit_transform(df['combined_features'])
if genre == 1:
    df['combined_features1'] = df.apply(combine_genrefeatures, axis = 1)
    count_matrix = cv.fit_transform(df['combined_features1'])

if castdir == 1:
    df['combined_features2'] = df.apply(combine_castfeatures, axis = 1)
    count_matrix = cv.fit_transform(df['combined_features2'])

if title == 1:
    df['combined_features3'] = df.apply(combine_titlefeatures, axis = 1)
    count_matrix = cv.fit_transform(df['combined_features3'])
```

The features we need are all in different columns, we need them all in a single column together as a string in each row.

So, basically we have 4 functions to combine one will be basic one which will combine all the 5 features. The first function will combine only 2 features genre and keyword. The second function will combine director and cast and the last function will combine title and keywords. There functions are applied on each row which adds the respective feature values.

Example for countVectorizer.fit_transform

```
[36] example = ['London Paris London','Paris Paris London']

[37] eg_matrix = cv.fit_transform(example)

[38] print(eg_matrix)

  (0, 0)        2
  (0, 1)        1
  (1, 0)        1
  (1, 1)        2

print(eg_matrix.toarray())

[[2 1]
 [1 2]]
```

CountVectorizer is basically a class which convert a collection of text document to a matrix of token counts. The function fitTransform of the class CountVectorizer learn the vocabulary dictionary and return a document term matrix.

Based on the button we click we have 4 if statements and respective combine features function is called.

```
print(count_matrix)
```

```
  (0, 1036)     1
  (0, 170)      1
  (0, 238)      1
  (0, 5284)     1
  (0, 13728)    1
  (0, 5437)     1
  (0, 7809)     1
  (0, 2471)     1
  (0, 3707)     1
  (0, 3100)     1
  (0, 5854)     1
  (0, 14528)    2
  (0, 16754)    1
  (0, 3261)     1
  (0, 14425)    1
  (0, 13480)    1
  (0, 17158)    1
  (0, 17448)    1
  (0, 13450)    1
  (0, 14211)    1
  (0, 16831)    1
  (0, 14767)    1
  (0, 8839)     1
  (0, 10322)    1
  (0, 13148)    1
  :     :
```

Basically fitTransform will count the number of occurence of each word in the document and generate a matrix.

It basically generates a sparse matrix. Sparse matrix is a matrix which contains very few non-zero elements. Then it is converted into an n-dimensional array ,then later it is converted into numpy array as it is fast and better for manipulations.

```
[26]
count_array = count_matrix.toarray()
print(count_array)
```

```
[[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]]
```

```
nparray = np.array(count_array)
print(nparray)
```

```
[[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]]
```

Here get_index_from_title is used to get the index of a particular movie from the dataset.

```
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]

def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

def cosine_similarity(v1,v2):
    sumxx, sumxy, sumyy = 0, 0, 0
    for i in range(len(v1)):
        x = v1[i]; y = v2[i]
        sumxx += x*x
        sumyy += y*y
        sumxy += x*y
    return sumxy/(math.sqrt(sumxx)*math.sqrt(sumyy))
```

Cosine similarity example

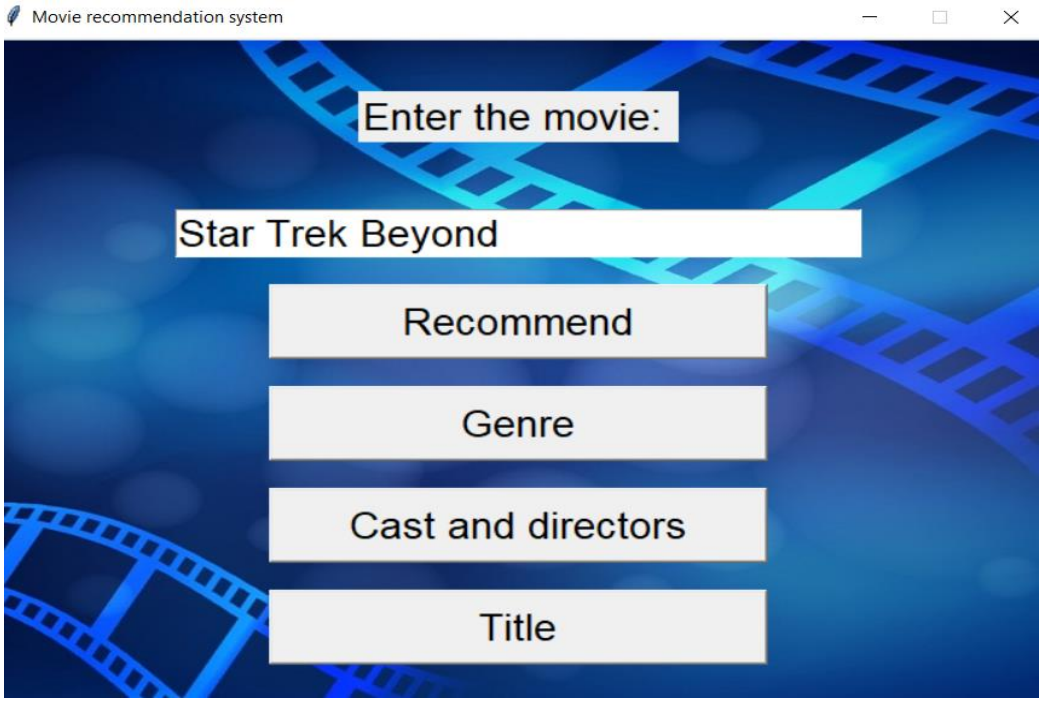
```
[82] example = ['London Paris London','Paris Paris London']

[83] eg_matrix = cv.fit_transform(example)

[84] doc_vectors = eg_matrix.toarray()

cosine_sim=cosine_similarity(doc_vectors[0],doc_vectors[1])
print(cosine_sim)

0.7999999999999998
```

Cosine similarity is used to measure the text similarity between 2 documents. Mathematically, it measures the cosine of the angle between 2 vectors projected in a multi-dimensional space.

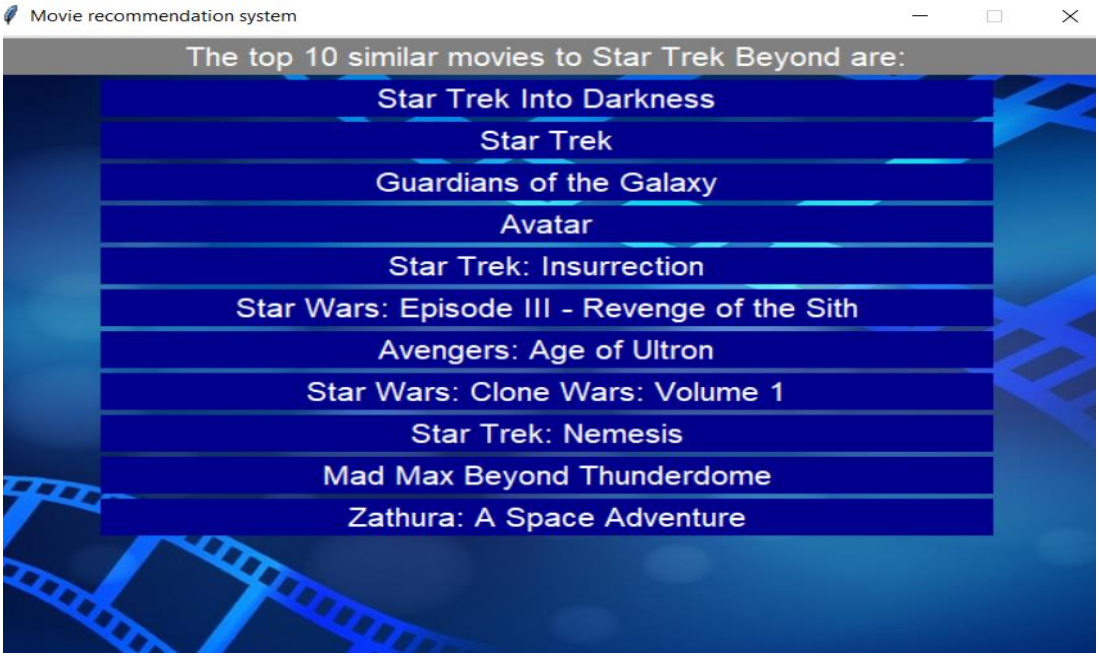
The smaller the angle, higher the cosine similarity. The cosine similarity will be 1 if the two vectors are identical and it will be 0 if the two are orthogonal.

RESULTS

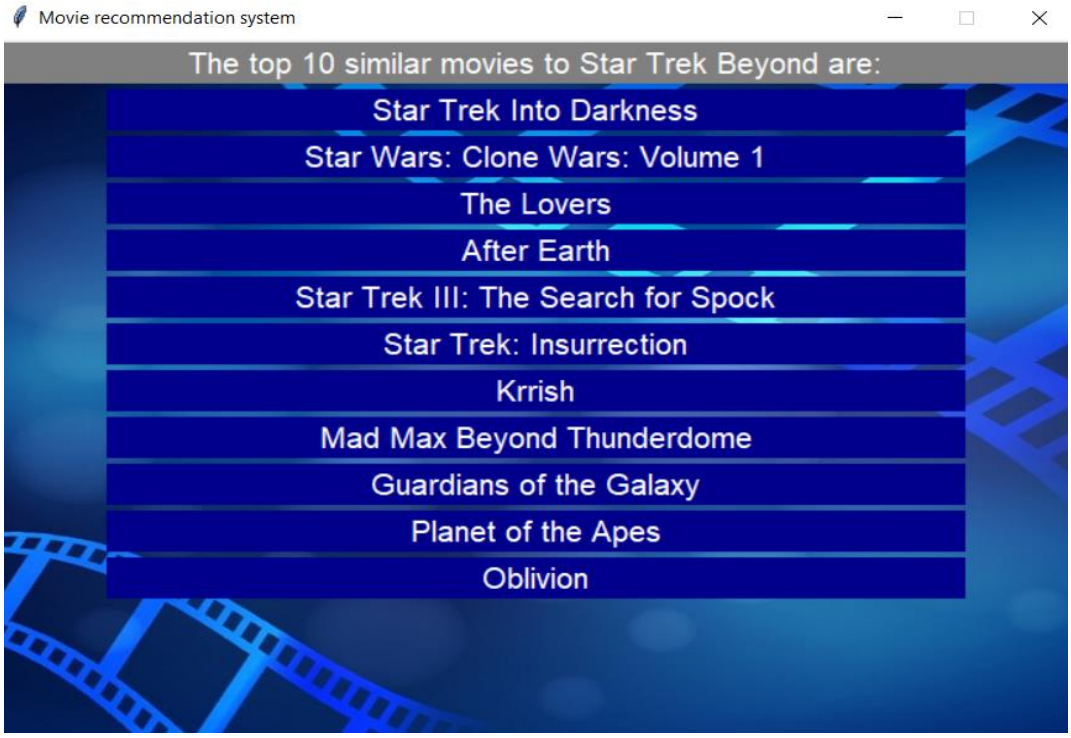
Here we can enter the movie whichever is there in the dataset and click on any of the 4 buttons.



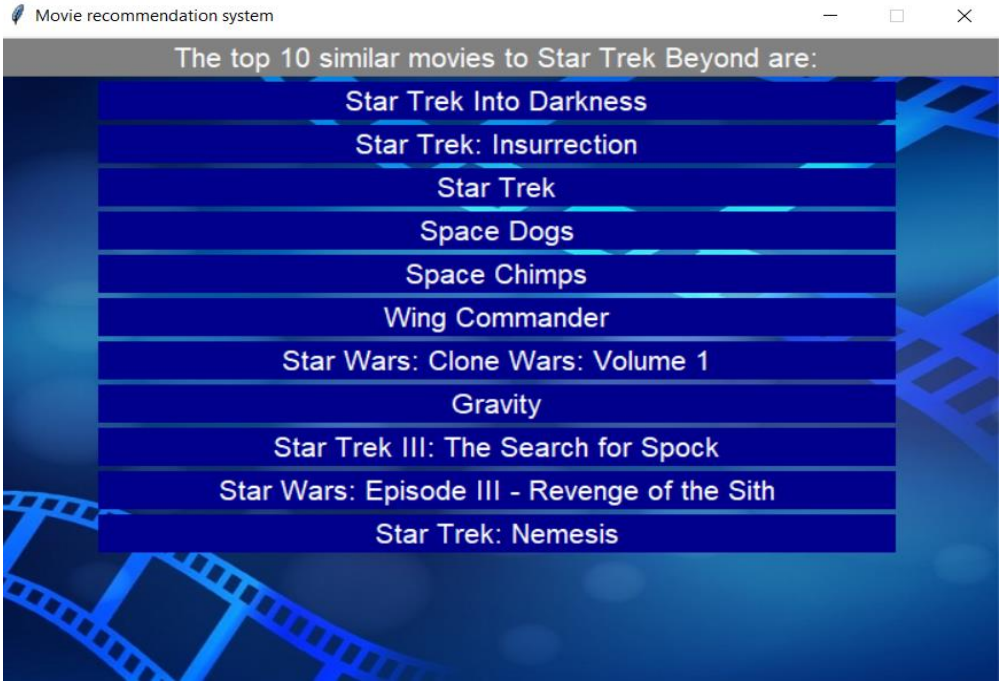
Recommend: It is general recommendation which will include many features.



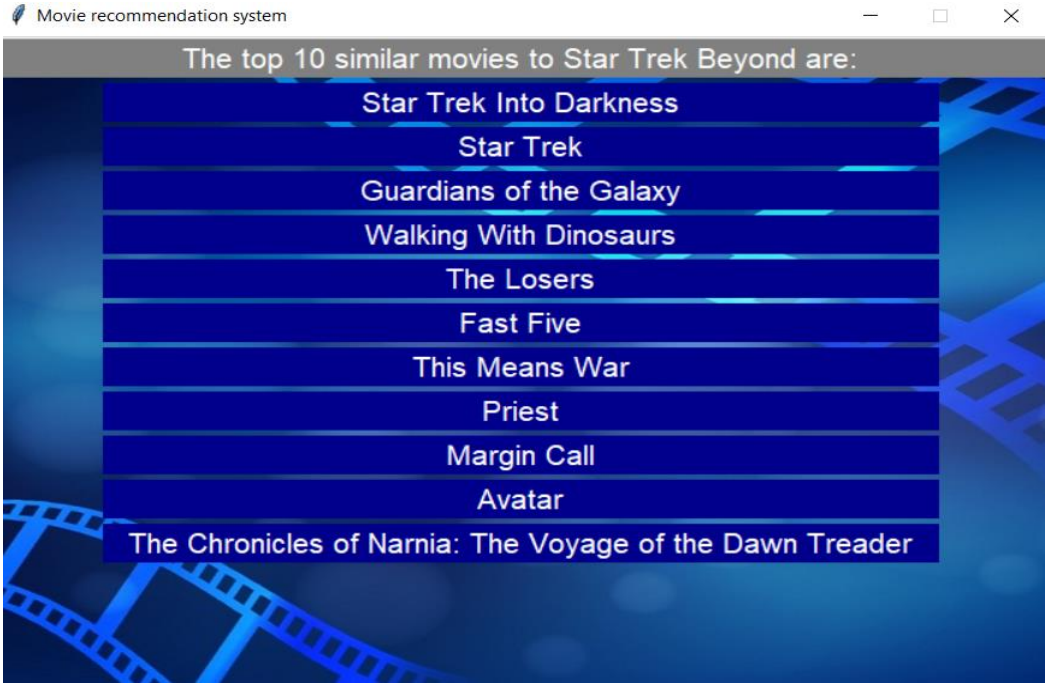
Genre: It is recommendation particularly based on the genre of the movie whichever we have entered.



Cast and Director: It is for recommendation based on the cast and director of the movie we have entered.



Title: It will be recommending based on the title we have entered.



CONCLUSION

Content-based recommendation systems recommend an item to a user based upon a description of the item and a profile of the user's interests. While a user profile may be entered by the user, it is commonly learned from feedback the user provides on items. A variety of learning algorithms have been adapted to learning user profiles, and the choice of learning algorithm depends upon the representation of content. Although there are different approaches to learning a model of the user's interest with content-based recommendation, no content-based recommendation system can give good recommendations if the content does not contain enough information to distinguish items the user likes from items the user doesn't like.

The recommendation system implemented in this paper aims at providing movie recommendation based on the genres of the movies. If a user highly rates a movie of a particular genre, movies containing similar genres will be recommended to him. Recommendation systems are widely used in today's era of Web 2.0 for searching for reliable and relevant information. While simple recommendation systems recommend users based on a few parameters, complex ones take many parameters into consideration.

By implementing machine learning in recommender systems, intelligent recommendations can be made for customers. Given the potential of such systems, they have a huge commercial value. Several MNCs have been exploiting the potential of recommendation system to lure customers into using their products. This also impacts greatly on the field of data mining and web mining. Mobile cloud computing (mcc) is able to save energy, improve application and experience of the users. All frameworks mentioned above have their own benefits and issues but still not up to level to address all issues related to security, energy and user experience. Security issues are key problem in mcc, they need to be focused more compare to other issues.

REFERENCE

- https://developers.google.com/machine-learning/recommendation/content-based/basics
- https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421
- https://www.educative.io/edpresso/what-is-content-based-filtering
- https://www.geeksforgeeks.org/ml-content-based-recommender-system/
- https://www.upwork.com/resources/what-is-content-based-filtering
- https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/
- [1] Kim, Mucheol, and Sang Oh Park, "Group affinity based social trust model for an intelligent movie recommender system", Multimedia tools and applications 64, no. 2, 505 516, 2013
- [2] Sharma, Meenakshi, and Sandeep Mann, "A survey of recommender systems: approaches and limitations", Int J InnovEng Technol. ICAECE-2013, ISSN, 2319-1058, 2013
- [3] Rattanajit banjong, Nutcha, and Saranya Maneeroj, "Multi criteria pseudo rating and multidimensional user profile for movie recommender system", In Computer Science and Information Technology (ICCSIT) 2009, 2nd IEEE International Conference on IEEE, pp. 596-601, 2009
- [4] Christakou, Christina, Leonidas Lefakis, Spyros Vrettos, and Andreas Stafylopatis, "A movie recommender system based on semi-supervised clustering", in Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on IEEE, vol. 2, pp. 897-903, 2005.