MOVIE RECOMMENDER SYSTEM

JUNE 7

Authored by:
Rutvi Gohel

Content-Based Movie Recommender System

Rutvi Gohel

Final Project Document

SpringBoard

Contents:

1. Introduction:

In the era of information overloaded, it is very difficult for users to get information that they are really interested in. It is very hard for the content provider to make their content stand out from the crowd.

Have you ever wondered how Netflix or Hotstar recommends new movies? These suggestions or recommendations are done by a system called a recommendation system. We tend to watch similar content on YouTube or similar movie on Netflix. Recommender system suggest/ recommend the movie, content, product by learning and understanding the pattern in user's watch history or buy history, and then applies those learning to make new suggestions to watch/buy; that is known as 'Recommendation'.

Data from Netflix's site shows that 80% of what people watch comes from their recommendation system. As a result, recommender systems generate over $1 billion per year of Netflix's revenue.

2. Problem Statement:

For building a recommender system from scratch, I have several questions in my minds,

- What approach should I use to recommend movies, Content based, or Collaborative based?
- How to recommend movies when there is no user information?
- What kind of movie features can be used for the recommender system?
- How to calculate the similarity between two movies.

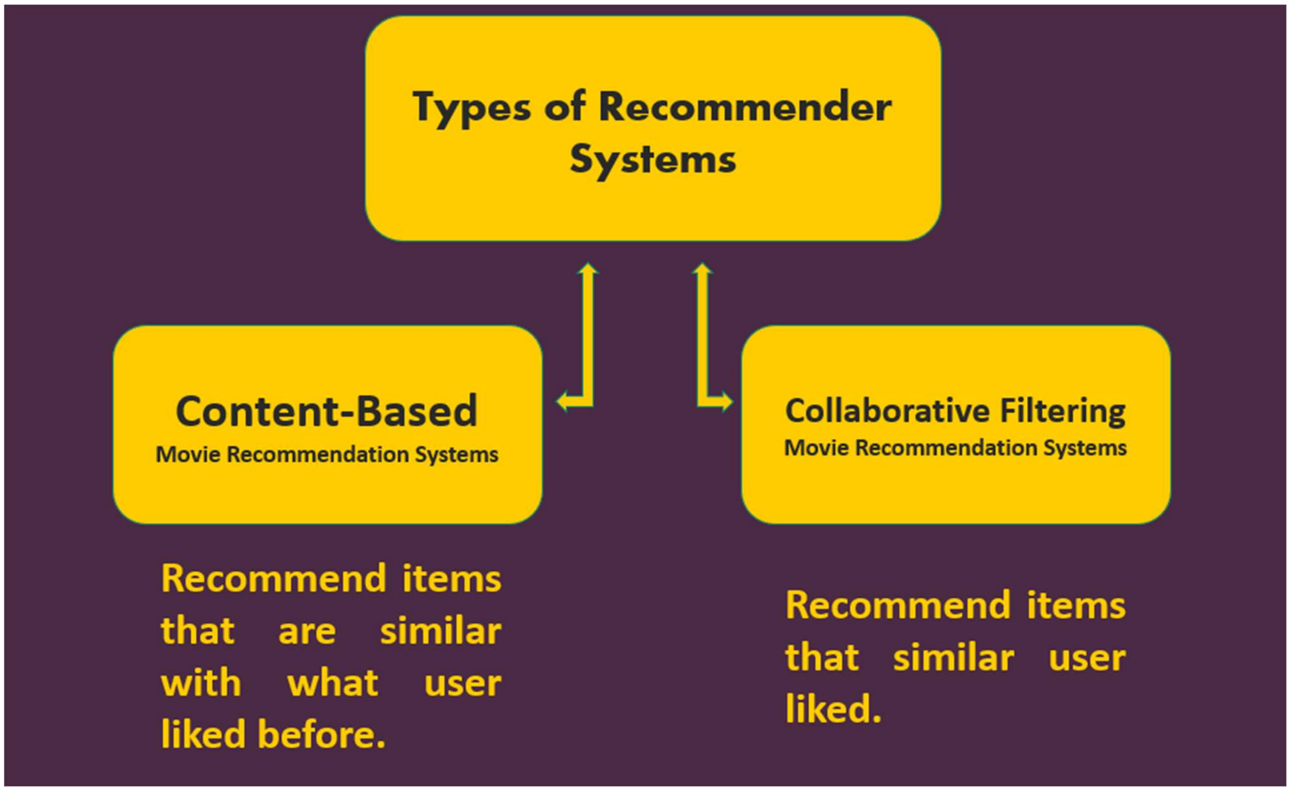
3. Goal:

The goal of this project is to generate meaningful movie recommendation to the user that they are really interested in.

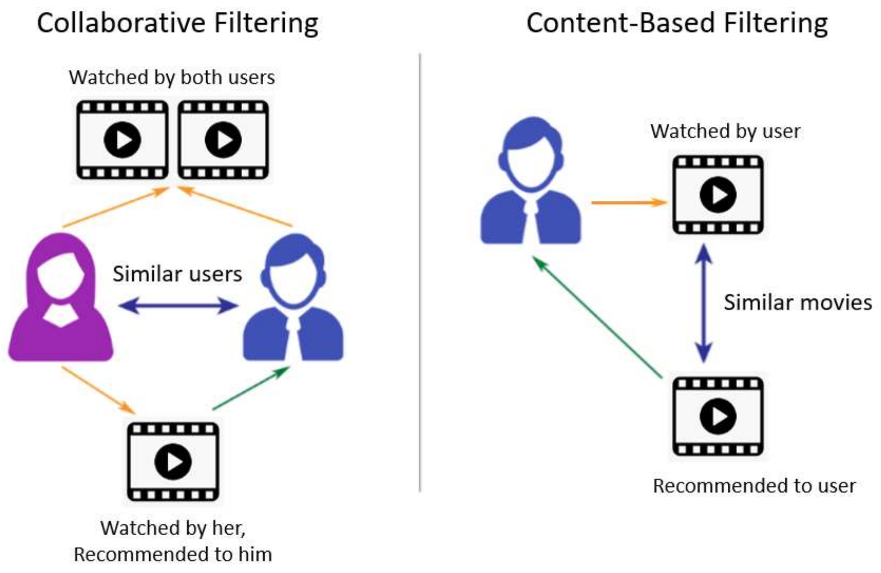
4. Approach

Recommender system is a very hot research topic in recent years. Many researchers raised a lot of different recommendation approaches. The most famous category of these approaches is:

- Content-based Recommendation.
- Collaborative-filtering Recommendation.
- Hybrid Recommendation.


Types of Recommender Systems
Content-Based
Movie Recommendation Systems
Collaborative Filtering
Movie Recommendation Systems
Recommend items that are similar with what user liked before.
Recommend items that similar user liked.

Content-based recommendation is an important approach in recommender systems. The basic idea is to recommend items that are similar with what user liked before. The core mission of content-based recommender system is to calculate the similarity between items. The tastes of a user can be modeled according to the history of what the user liked.



Here, we used Content based approach to recommend movies. The content-based approach uses information about users and movie’s category of movie, overview, genres, main actors, and director. This filtering method uses movie features to recommend other movies similar to what the user likes and based on their previous watch history. The main idea of content-based methods is to try to build a model, based on the available “features”, that explain users’ behavior to choose movie.

Recommendation Filtering	Advantages	Disadvantages
Content Based Filtering	• Recommendation results are intuitive and easy to understand • The system can recommend new movies	• Cannot recommend movie to new users
Collaborating Filtering	• Having movie knowledge is not necessary. • Performance improving as the users increasing	• Can not recommend movie to new users • New movie cannot be categorized. • Need many users to improve recommendation.

5. Famous Recommender Systems

There are a lot of websites using recommender system in the world. Personalized recommender system analyzes a huge amount of user behavior data and provides personalized content to different users, which improves the click rate and conversions of the website. The fields that widely use recommender system are e-commerce, movie, video, music, social network, reading, local based service, personalized email, and advertisement.

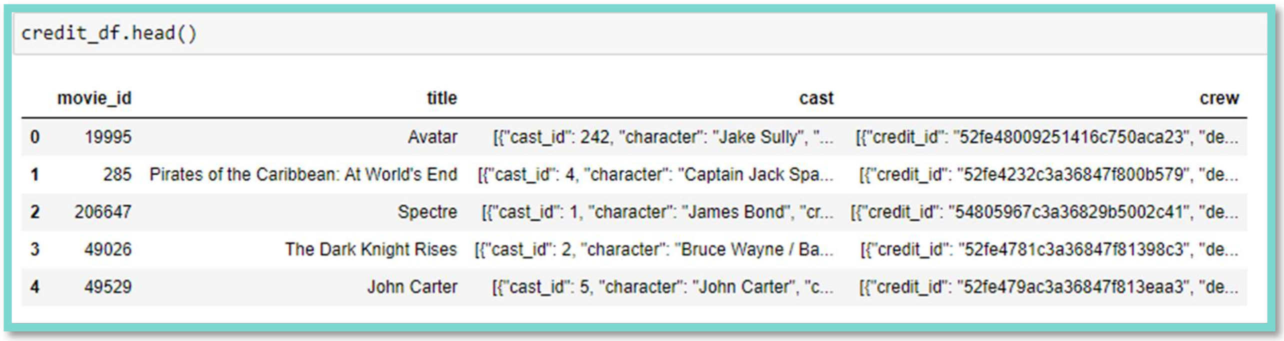
Movie and Video Website Personalized recommender system is a very important application for movie and video website, which can help users to find what they really like among the vast of videos. Netflix is the most successful company in this field.

6. Dataset: Kaggle Dataset

The Movie Database (TMDB) is a community-built movie and TV database. This contains two datasets, named

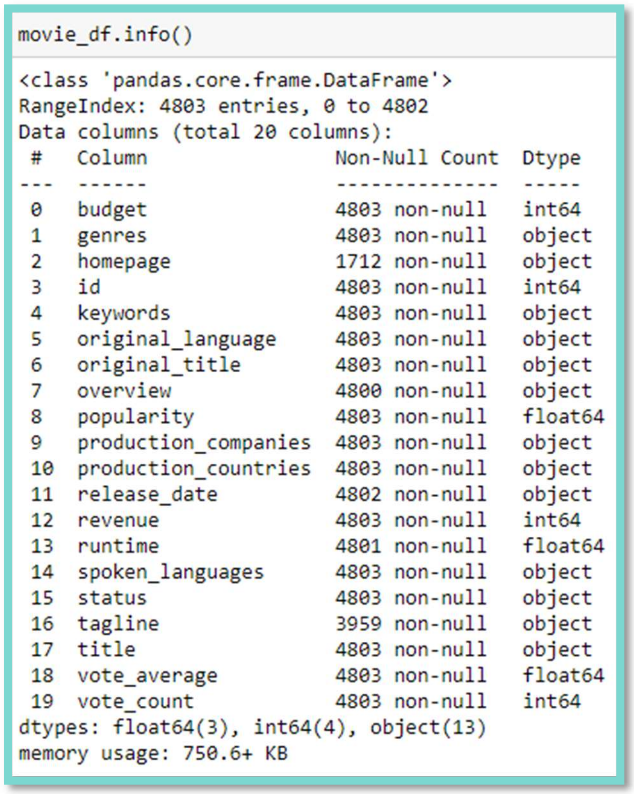
1. credit_df = tmdb_5000_credits.csv
2. movie_df = tmdb_5000_movies.csv

Credit dataframe contains Movie id, Title, Cast and Crew. Cast and crew is in form of list of dictionaries, and Movie dataframe contains 20 columns. We combined these 2 dataframes and new dataframe, merged has 4809 rows and 23 columns.



```
credit_df.head()
```

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...



```
movie_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title                 4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

7. Feature Extraction from Dataframe

What encourages user's decision in choosing a movie to watch?

- Genres
- Main Actors (Top 3 casts)
- Directors
- Overview (summary)

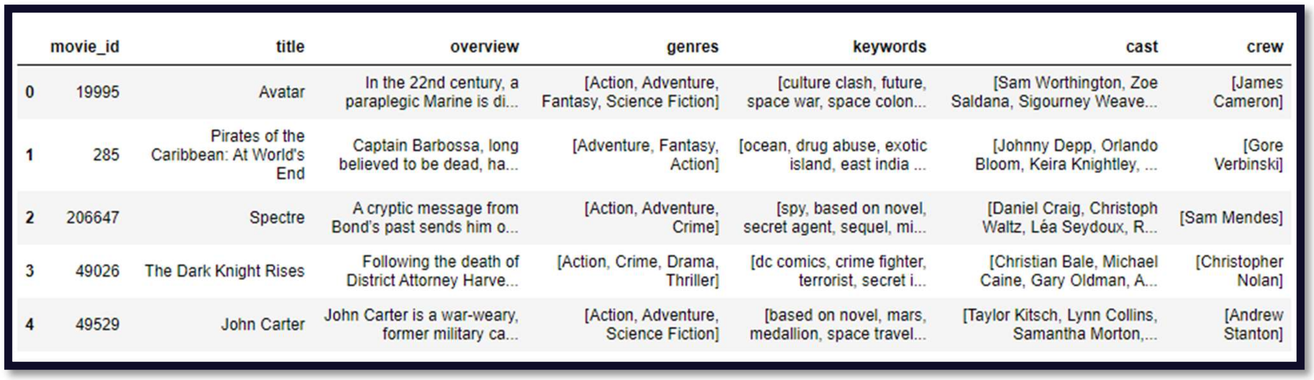
So, I have created tags (documents) using overview, genres, keywords, cast and crew features.

8. Pre-Processing

Genres and keywords features are in form of list of dictionaries, and it contains genres id-name and keyword id-name. As we need only name of the genres and name of the keywords, we created a function, named convert, to fetch names.

Cast means the actors in a movie. Mainly user focus on top 3 actors. So, we are going to fetch first 3 actors from the list of the cast.

Crew is a group of people who are involved in the practical and technical business of shooting a film. There are many crew members hired for a film. We only need to fetch name of the director in this list. Now, our dataframe looks very clean.



	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley, ...	[Gore Verbinski]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux, R...	[Sam Mendes]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman, A...	[Christopher Nolan]
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton,...	[Andrew Stanton]

Are we ready to create tags? Not yet!

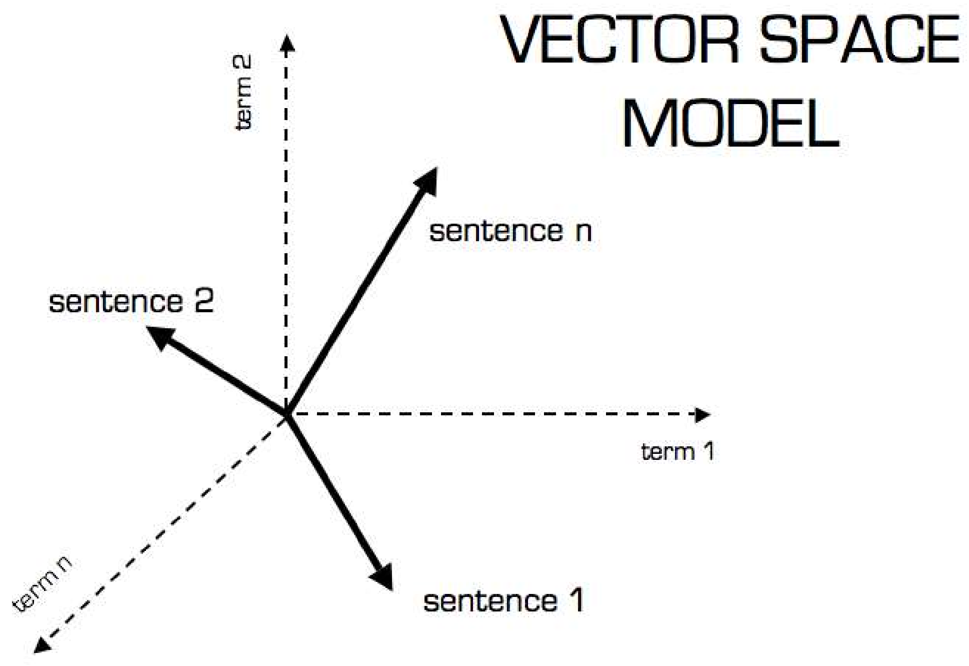
If we are creating tag at this moment, we have Sam Worthington as a cast and Sam Mendes as crew. Our system will create 3 tags from these 2 names. The system will think, Sam (Worthington) and Sam (Mendes) is a same person. Let's say, a user likes Sam (Worthington), our system will recommend Sam (Mendes) movie as well. To solve this problem, I will replace space between first name and last name with no space. So, now our system will create two tags.

Before combining the name	After combining the name
1. sam 2. worthington **3.** mendes	1. samworthington 2. sammendes

Now we are ready to create Documents(tags). For that, we combined ‘genres’, ‘keywords’, ‘cast’ and ‘crew’ features.

9. Text-Vectorization

For Natural Language Processing (NLP) to work, it always requires transforming natural language (text and audio) into numerical form. Text vectorization techniques namely Bag of Words and TF-IDF vectorization, which are very popular choices for traditional machine learning algorithms can help in converting text to numeric feature vectors.



Here, we are using TF-IDF, that is common weighting technique for information retrieval and text mining, which reflects how important a word is for a document. The importance of a word increases proportionally to the times the word appears in the document, but it also decreases inversely proportional to the frequency the word appears in the whole corpus.

TF-IDF consists of TF and IDF, which are ‘Term Frequency’ and ‘Inverse Document Frequency’ respectively. TF represents the frequency a word appears in the document. The main idea of IDF is: if a term appears more in other documents, the term will be less important.

TF-IDF is better text vectorization technique because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions.

Term frequency is about how many times a term i appears in document j , which can be represented by TF(ij). In the condition of removing stop-words, the more i appears in the document, the more important term i is for the document. Inverse document frequency (IDF) shows how frequently a keyword is used within a collection of documents.

It can be defined as:

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2\left(\frac{\text{Total documents}}{\text{documents with term i}}\right)$$

10.TF-IDF Parameters

We applied TF-IDF techniques with 3 parameters,

- stop_words = English
  Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information.

- max_features = 5000
  Build a vocabulary that only consider the top max_features ordered by term frequency across the corpus.

- lowercase=True
  Convert all characters to lowercase.

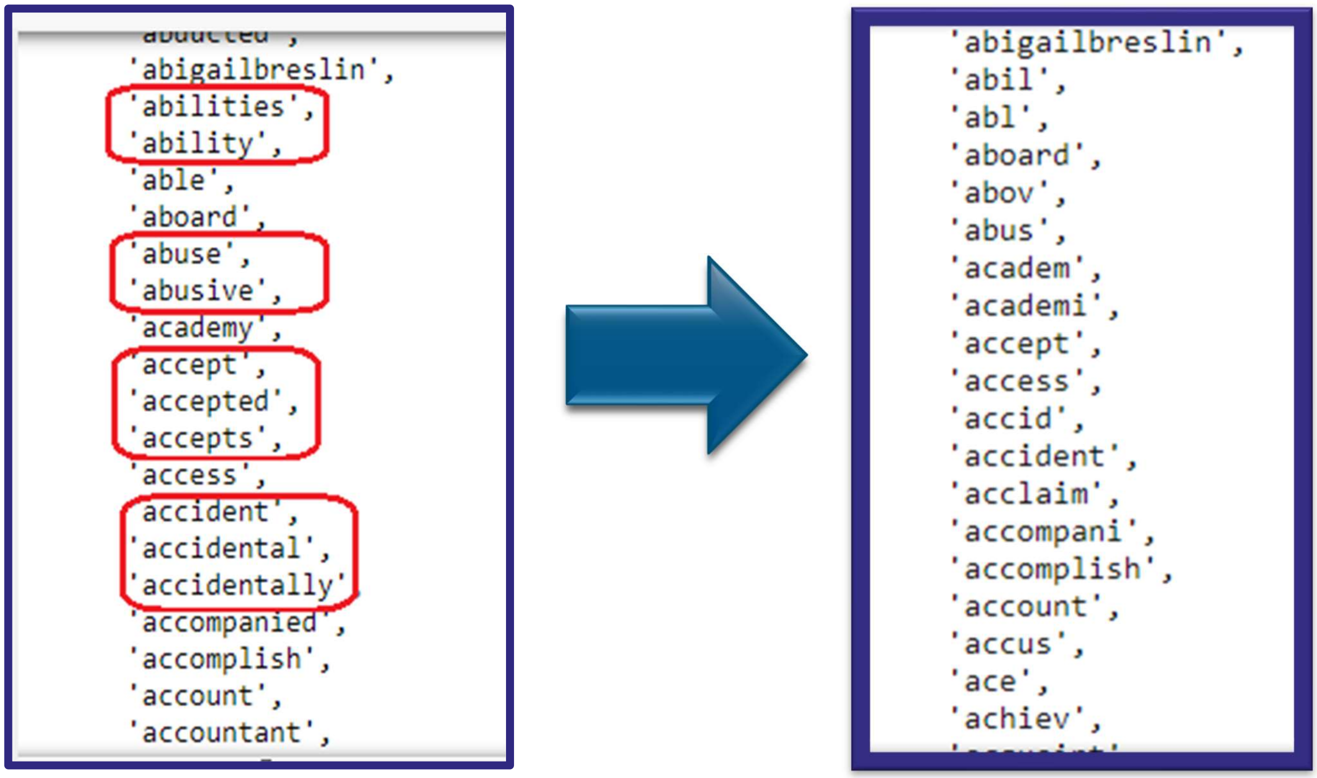
11. Documents

In many other content-based recommender systems, the genre is used as vector to calculate similarity. But this is only one aspect of the movie and there are a lot of other features of movie such as Overview, Actor, Directors, Genres etc. So, we added all these features. Some of them are unique because they are extracted from list of dictionaries. instead using the genres, we used other features to get our documents

Tags (Documents) = Genres + Top 3 casts + Directors + Overview + Keywords

12. Stemming

By applying TF-IDF parameters to corpus, we got our 5000 features as shown in the figure. But, consider words, “abilities and ability”, “abuse and abusive”, “accept, accepted, and accepts” etc. They all can be reduced to the same word “ability, abuse, accept”. After all, all these convey the same meaning. This helps reduce complexity while retaining the essence of meaning carried by these words. Stemming is the process of reducing words like these to its stem or root format. Finally, we got highest occurring 5000 words.

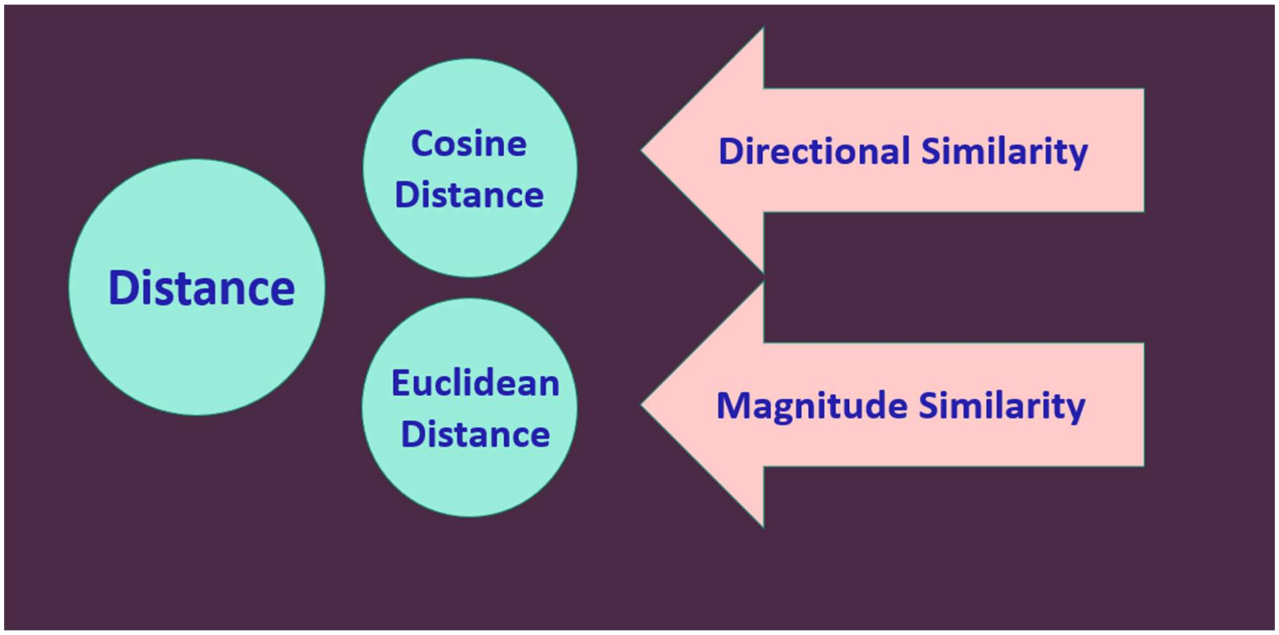


```
'abducted',
'abigailbreslin',
'abilities',
'ability',
'able',
'aboard',
'abuse',
'abusive',
'academy',
'accept',
'accepted',
'accepts',
'access',
'accident',
'accidental',
'accidentally'
'accompanied',
'accomplish',
'account',
'accountant',
```

```
'abigailbreslin',
'abil',
'abl',
'aboard',
'abov',
'abus',
'academ',
'academi',
'accept',
'access',
'accid',
'accident',
'acclaim',
'accompani',
'accomplish',
'account',
'accus',
'ace',
'achiev',
```

13.Distance Measurement

Calculate the similarity between movies is the objective of content-based recommender systems. The content can be anything such as text, video, and image. In our project, each movie is represented by a feature vector. I will introduce the cosine similarity algorithm then.



The cosine similarity looks at "directional similarity" rather than 'magnitude differences'. Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter. This happens for example when working with text data represented by word counts. We could assume that when a word (e.g., Suspense) occurs more frequent in document-1 than it does in document-2, that document-1 is more related to 'Suspense' movies. However, it could also be the case that we are working with documents of uneven lengths (Overview of the movie, for example). Then, 'Suspense' probably occurred more in document-1 just because it was way longer than document 2. Cosine similarity corrects for this. Text data is the most typical example for when to use this metric.

To calculate the similarity between two features, we can calculate the cosine of the angle between the feature vector using this Equation,

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

The cosine similarity between two vectors is measured in θ.

- If θ = 0,
  Movie A and Movie B overlap, thus proving they are similar.
- If θ = 90,
  Movie A and Movie B are dissimilar.
- If θ = 180,
  Movie A and Movie B are opposite.

The range of the similarity is between -1 and 1. -1 means that the direction of the two vectors are totally opposite (θ = 180°) and 1 means they are in the same direction (θ = 0°).

Actually, we have 4803 Movies, and we are comparing these movies with one-another. So, shape of this similarity matrix is (4803, 4803).

Here is an example for illustrating cosine similarity.

14.Example

Given two sentences:

- Sentence 1: I like watching TV, but I don't like watching films.
- Sentence 2: I don't like watching TV and films.

How can we calculate the similarity between the two sentences? The basic idea is: the more similar the words used by the two sentences are, the more similar the sentences are.

First step: Word segmentation.

- Sentence 1: I / like / watching / TV, but / I / don't / like / watching / films.
- Sentence 2: I / don't / like / watching / TV / and / films.

Second step: List all the words (Corpus).

I / like / watching / TV/ but / I / don't / like / watching / films/ I / don't / like / watching / TV / and / films

Third step: List unique words from corpus (Vocabulary).

I
like
watching
TV
but
don’t
films
and

Forth step: TF-IDF calculation

Sentence 1: I like watching TV, but I don’t like watching films.
Total text in ‘sentence1’ Document = 10

Text in Sentence1	TF
I	2/10
like	2/10
watching	2/10
TV	1/10
But	1/10
Don’t	1/10
films	1/10

Sentence 1: I don’t like watching TV and films.
Total text in ‘sentence1’ Document = 7

Text in Sentence2	TF
I	1/10
Don’t	1/10
like	1/10
watching	1/10
TV	1/10
and	1/10
films	1/10

And, now let’s find IDF

Vocabulary	IDF Calculation	IDF
I	$\text{Log}_e(2/2)$	0
like	$\text{Log}_e(2/2)$	0
watching	$\text{Log}_e(2/2)$	0
TV	$\text{Log}_e(2/2)$	0
but	$\text{Log}_e(2/1)$	0.3010
don’t	$\text{Log}_e(2/2)$	0
films	$\text{Log}_e(2/2)$	0
and	$\text{Log}_e(2/1)$	0.3010

If Text is rare (not occurs in all documents), IDF will increase.
If Text is common (occurs in all/most documents), IDF will decrease.

And TF X IDF will be,

TF-IDF	I	like	watching	Don’t	TV	films	and	but
Sent1	2/10 X 0	2/10 X 0	2/10 X 0	1/10 X 0	1/10 X 0	1/10 X 0	0 X 0.3010	1/10X0.3010
Sent2	1/10 X 0	1/10 X 0	1/10 X 0	1/10 X 0	1/10 X 0	1/10 X 0	1/10X0.3010	1/10X0.3010

This is our TF-IDF table.

Note,
But, to understand cosine similarity, we are considering another easy example with easy numbers.

Corpus	Document 1	Document 2
I	2	1
like	2	1
watching	2	1
TV	1	1
but	1	0
Don’t	1	1
films	1	1
and	0	1

Fifth Step: Text Vectorization

- Document 1 = [2,2,2,1,1,1,1,0]
- Document 2 = [1,1,1,1,0,1,1,1]

Then, we calculate the cosine of the two vectors,

$$\cos\theta = \frac{A \cdot B}{|| A || \; || B ||}$$

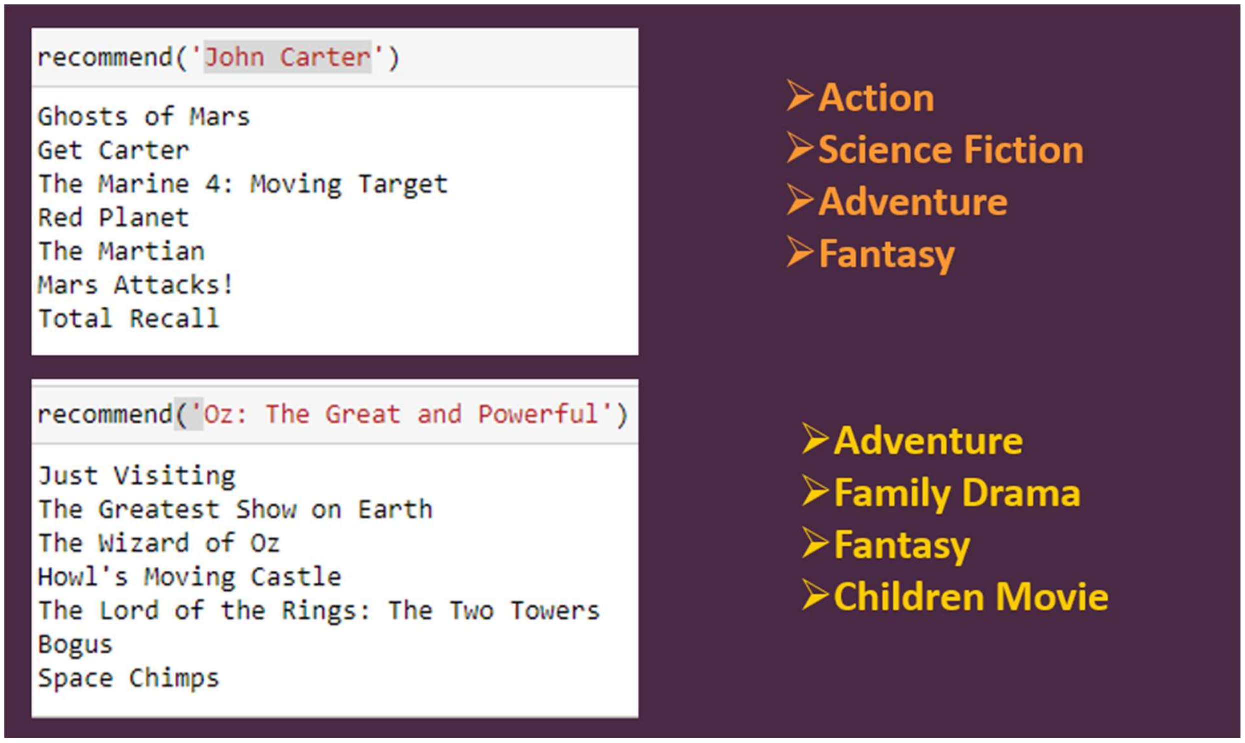
$$\frac{(2\times1)+(2\times1)+(2\times1)+(1\times1)+(1\times0)+(1\times1)+(1\times1)+(0\times1)}{\sqrt{2^2+2^2+2^2+1^2+1^2+1^2+1^2+0^2}\sqrt{1^2+1^2+1^2+1^2+0^2+1^2+1^2+1^2}}$$

$$= \frac{9}{\sqrt{16}\sqrt{7}} \approx 0.85 \qquad (\ldots)$$

The value is 0.85 so that the two sentences are quite different.

15.Recommendation Result:

We converted movie to vector space model and every movie is represented by a vector. Then we use cosine similarity to calculate similarity for each movie. The final recommendation of ‘John Carter’ and ‘Oz the Great and Powerful’,



```
recommend('John Carter')

Ghosts of Mars
Get Carter
The Marine 4: Moving Target
Red Planet
The Martian
Mars Attacks!
Total Recall
```

```
recommend('Oz: The Great and Powerful')

Just Visiting
The Greatest Show on Earth
The Wizard of Oz
Howl's Moving Castle
The Lord of the Rings: The Two Towers
Bogus
Space Chimps
```

John carter is a Action, science fiction, adventure and fantasy movie and the recommendation has same genres to John carters.

Oz: The Great and Powerful is a Adventure, fam*ily drama (kind of children)* and fantasy movie and the recommendation has same genres to Oz: The Great and Powerful.

16.Future Work:

I will consider the following aspects in future work.

- Use Hybrid filtering recommendation.
  When we have enough user data, we can introduce collaborative filtering recommendation with content based. Collaborative filtering is based on the social information of users, analyzing user's information can improve our recommendation system.
- Introduce user dislike movie list.
  The user data is always useful in recommender systems. Collect dislike movie list from user and generate similar movies to dislike movies. Do not recommend those movies!

17.Acknowledgement

I would like to *Thank* Mr. Daniel Wu for guiding me in this project.

Thank You,

Rutvi Gohel
Rutvimgohel@gmail.com
https://www.linkedin.com/in/rutvigohel-datascience/
https://github.com/PassionateDataScientist