

**Applications of Machine Learning**

2021 MOD006567 TRI2 F01CAM

Component 3: Report

Movie Recommender system  
Student ID- 1940984

Table of Contents

1. Objectives-------------------------------------------------------------------------------------- 3
2. Brief Solution----------------------------------------------------------------------- 3
3. Technique--------------------------------------------------------------------------- 3
4. Motivation------------------------------------------------------------------------------------- 3
5. Introduction----------------------------------------------------------------------------------- 4
6. Content Based--------------------------------------------------------------------- 4
7. Collaborative filtering------------------------------------------------------------ 4
8. Hybrid-------------------------------------------------------------------------------- 4
9. Content Based Recommendation-------------------------------------------------------- 4
10. Dataset------------------------------------------------------------------------------- 4
11. Pre-Processing Data------------------------------------------------------------------------- 6
12. New Data frame------------------------------------------------------------------------------ 10
13. Vectorization---------------------------------------------------------------------------------- 11
14. Applying Stemming---------------------------------------------------------------- 13
15. Calculating distance between Vectors---------------------------------------- 13
16. Using Cosine Function----------------------------------------------------------------------- 13
17. Conclusion-------------------------------------------------------------------------------------- 14
18. References------------------------------------------------------------------------------------- 15

**Objectives**

The focus of the project is to create a website where the user will give a name of the movie and in return, the model will have to give 5 similar movies which are highly recommended based on the similarity of the content.

For checking the similarity, we are creating tags for all the movies. As the data we have chosen is pure text-based data it is not that easy to find the similarity score between the movies. If the data was in numbers, we could have easily found using mathematical equations for similarity check.

* What we can do for the pure text-based data?

We can check for similar words in the tags between the movies and check for similarity. But it is not an accurate way of checking similarity between them.

**Brief Solution**

So, what we will be doing in this project is we will be vectorizing the data, which means we will be converting the text data into vector, and if we talk in brief, every movie will become a vector on a multidimensional space and when the user gives a name of the movie then we recommend the closest vectors location in the space will be considered and 5 of the closest will be extracted and recommended to the user.

**Technique**

Here we are using ‘bag of words’ which is one of the famous vectorization techniques, where we will combine all the tags which we will be generating for the movies to get one big string which has each and every tag for any movie available in the dataset. We will be calculating the frequency of each word in the large string so that we can extract the most commonly used 5000 words which are frequently used.

Our aim is to select as a smaller number of words as possible to get the accuracy.

The time we perform vectorization we will have to do one important thing that is we should not consider the ‘stop words’. Stop words are nothing but the words which are used for sentence formation and they don’t posses any contribution in the meaning of the sentence.

There are other techniques as well which can be used for vectorizing the data like tfidf, word2vec.

**Motivation**

1. I personally like watching movies and when I want to watch something I am always confused what to watch, so with the help of this project I can enter a name of the movie which I have seen and want to watch similar type of movie for example,

I have seen ‘Avengers’ and I really liked the concept of the movie and I want to watch similar content.

1. My brother watches gaming videos on streaming platforms and when the video is about to finish there use to be recommendations for similar type of videos, it is because of the tags which were created by the publisher which he use to enjoy because every time rather than searching for a next video to play the platform was already recommending him similar content to watch.

**Introduction**

What is a recommender System?

Recommender System is nothing but a tool which helps user to find similar content and overcome the information overload. With the help of building a interest model based on the users It will predict the interest of the user.

With the help of the knowledge on a product the system will recommend the user with the help of the user’s preferences. Now a days in almost all the sectors we can see the usage of recommender system, for example e-commerce, Spotify, YouTube, Facebook, Instagram, Netflix and many more.

If classified broadly there are 3 types of recommender systems.

1. **Content based**

Content based recommender system is nothing but the recommendation is done based on the similar content.   
Content-based recommender system is continuation and development of collaborative filtering, where we don’t need any user’s information or evaluation.

1. **Collaborative filtering**

It recommends based on the users interest, for example there are 2 user’s. A and B and based on there behaviour we can see that they have a similarity score of 0.9. And if a movie M1 is seen by user A and the user liked the movie. Then as user B is similar to that of A, the movie M1 is recommended to user B as well.

1. **Hybrid**

The combination of both Content and Collaborative based is know as Hybrid.

**Content Based Recommendation**

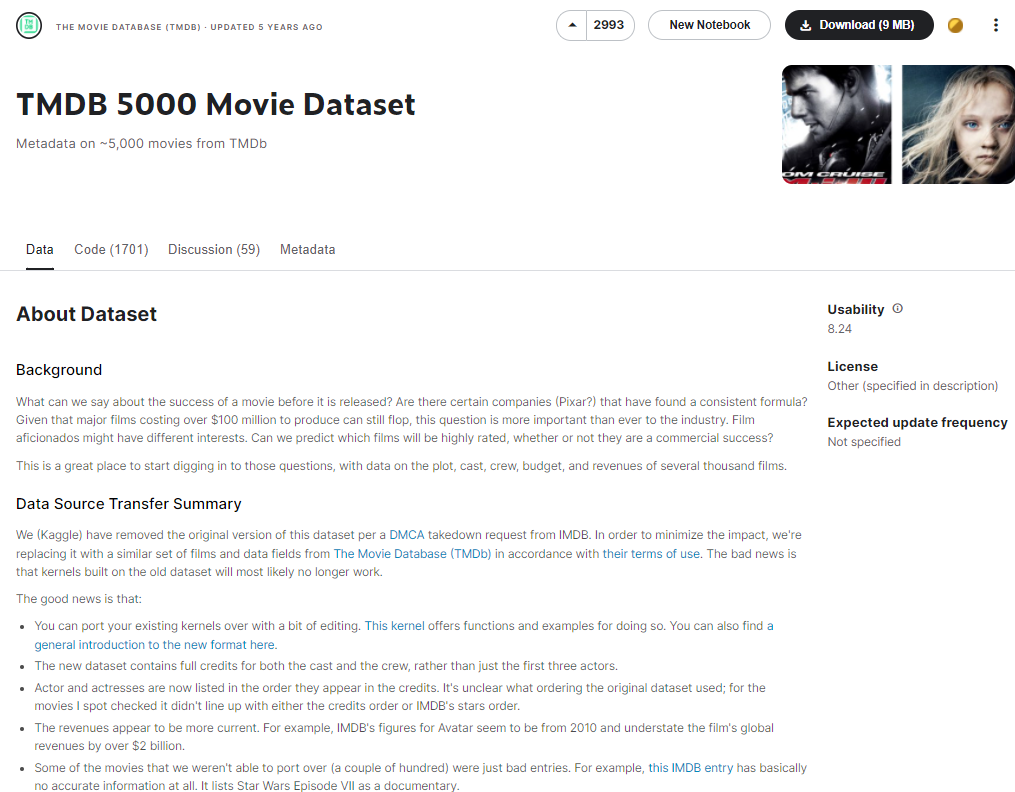
* Why Content Based Recommender System?

Content-based recommendation is commonly described as ‘recommending items that are similar to those the user liked in past’

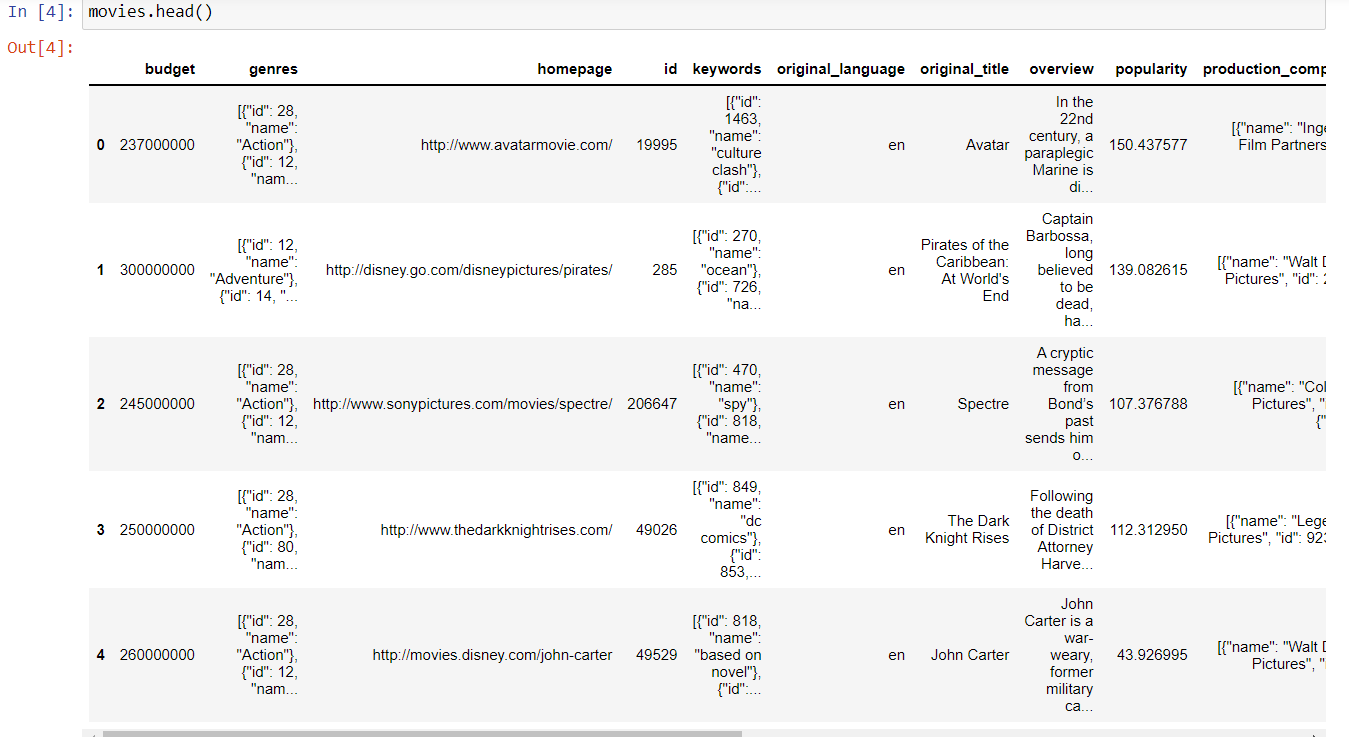
Here we are creating a recommender system for a movies dataset where we will be using Content based Recommender system. In the current era of overloaded information, it is really difficult for any user to find information that they are interested in. And if we talk about movies, we can create tags based on the description of the movies, genres, title, cast and crew, with which we can check the similarities between them and recommend accordingly.

**Dataset**

First, we need to find a dataset which we can use for this project. TMDB provides 5000 movies dataset called the TMDB 5000 movie dataset on Kaggle which we will be using because I found the features in the dataset are appropriate for finding the similarities between the movies.

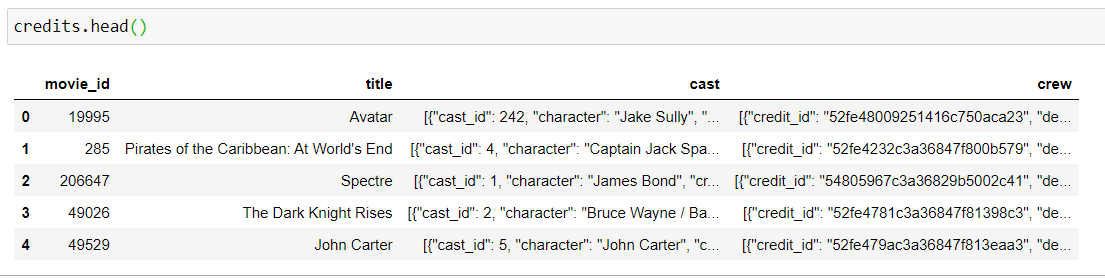


If we talk about the dataset there are 2 files available where we can find all the details of the movies. In the movies dataset we can find all the features like budget of the movie, genres, homepage of the movie, movie id on the TMDB website, keywords where it describes some keywords for the movie, original language, title of the movie, overview where we get the brief scenario of the movie, popularity, production companies and countries, release date, revenue generated by the movie, runtime of the movie, spoken languages, status and taglines. Overview is considered as one of the most important features specially for content-based recommender system.



In the credits dataset there are details about the cast and crew of the movie which are also important for content-based filtering.

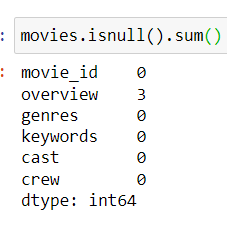
Based on the requirement of the project this dataset is a good starting point.



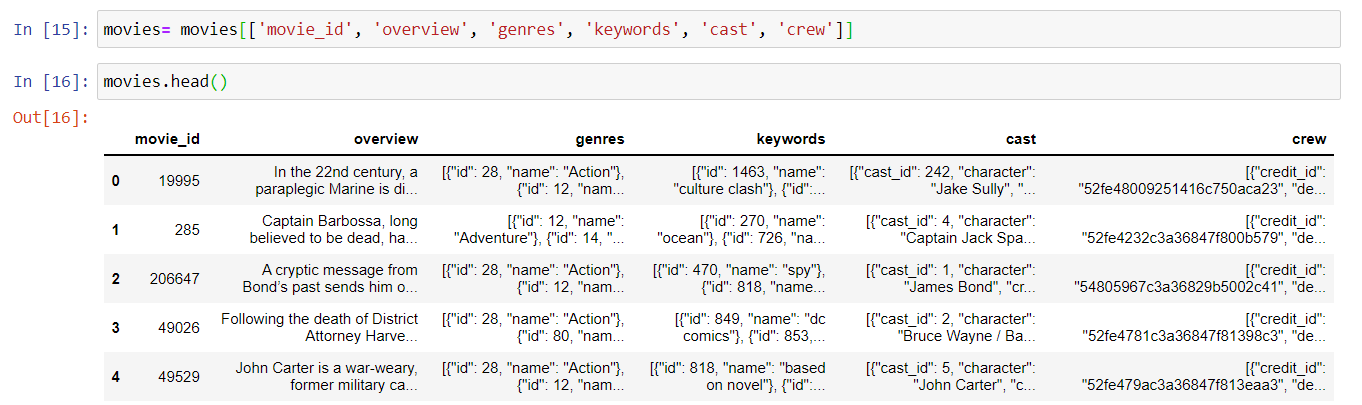
For building a recommender system from scratch, there are many problems we face. And the data from Kaggle which we are using is raw data and has many flaws in it. A lot of data pre-processing is required for getting the right features which we will be using to create tags which will help in checking for the similarities.

**Pre-Processing Data**

While data pre-processing, we first check if there are any missing data in the data frame. If any missing values are there, we have to drop them because when we deal with creating tags for movies, we need the data to be complete.



We also have to select the features that can be used for creating tags for the movies, like for example. The movie id, overview of the movie, title of the movie, genres, cast, crew and keywords.

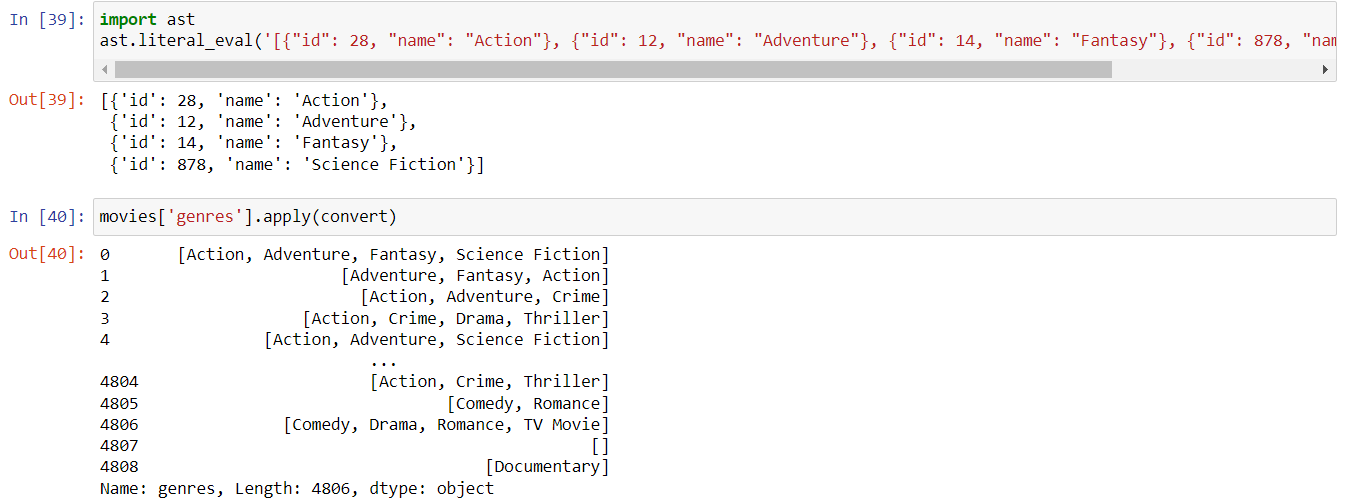


The columns when imported into a data frame, gives all the details of the movies. When importing the data there are 2 different files named movies and credits where we get the details of the movies and in credits, we get the details of the cast and crew of the movie. If we see the format of the data for a particular column for example the genres it has a list of dictionaries where not all the data in it are not useful so we have to perform some pre-processing on the data to extract only the features required from the list.

Like for example, from genre we don’t need all the data, we need only few features which gives the genres so we create a helper function and giving it a name where we run a loop on the list to get one dictionary and from that dictionary we need the name value and needs to be appended in the List and return the List.

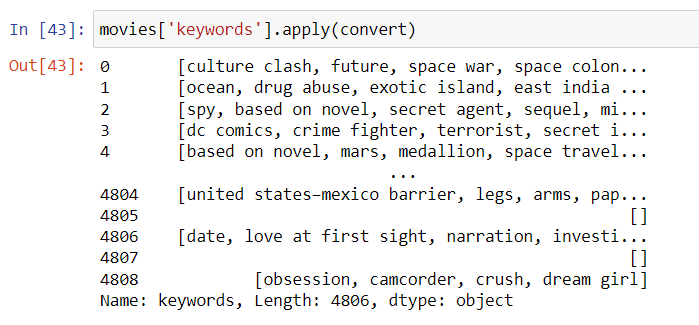
1. Genres

The data in the genres column has string data in it so when we call the List function there will be error, so for that we have to convert the string of list in the genres column into list, here we can use a module from python ‘ast’. And in that there is function named ‘literal\_eval’ which we will be using to get the list of strings from which we can retrieve only the genres name.



After that we will be applying the same to retrieve the genres and storing them of all the movies in the database by using the helper function which we created.

We will have to do the same pre-processing on the keywords column as well because the data in it is also in the form of list of strings. So we can use the same helper function to create list and extract only the name function form it.

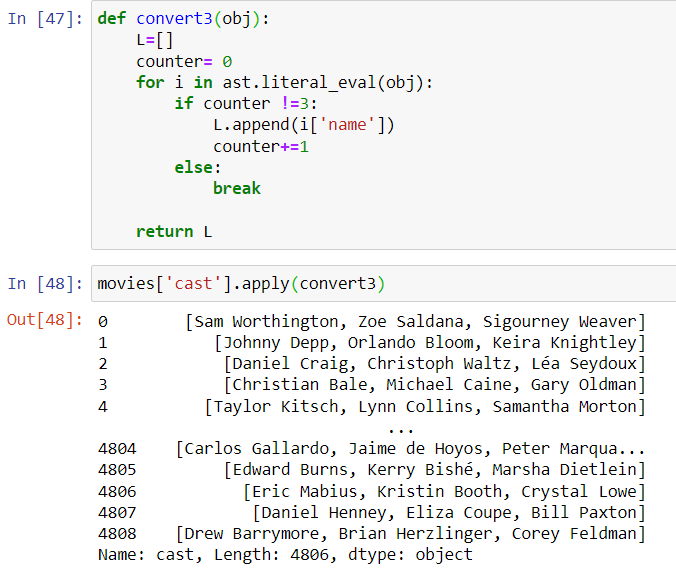


1. Cast

Now for applying data pre-processing on the cast column, we have huge data in the column where the data is in similar format that is list of strings, for now we will consider only the first 3 dictionaries where we find the top 3 actors from the movie. And extract the lead role actors which are also considered as tags for the movies.



So using the same helper function and applying an if condition where it limits to 3 because we need only the top three lead roles from the cast, creating tags. By applying this helper function we get the lead cast for the movies in the cast column.

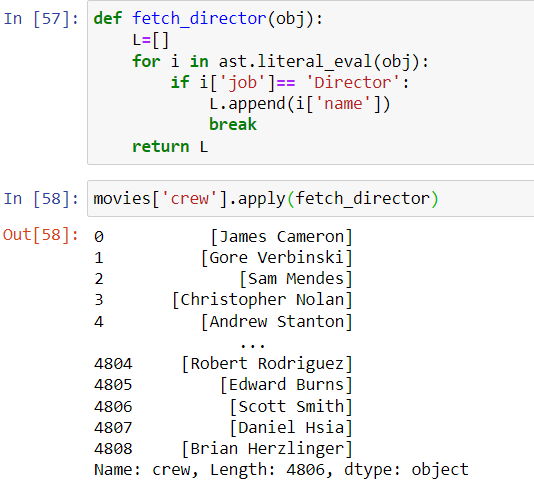


1. Crew

Similarly, we see the crew column. The data in it is also huge with many list of strings where we need not use all the dictionaries, we will only consider director which makes sense to all in tags for the movie. Here also we will use the similar helper function where we will find the job title as ‘director’ and extract the names of the director for the column of crew.



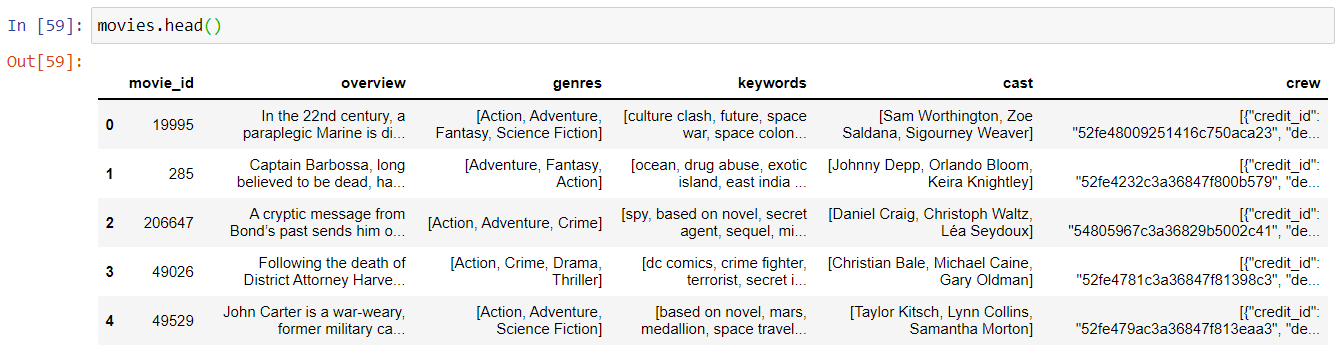
Now we can see that the overview column can be seen as string, so we need to change it to list as the basic concept of using method ‘bag of words’ is that the data has to be in list so that we can easily concatenate the columns. For concatenation we need a lambda function, so we can split the string.



There is a problem with the columns even after pre-processing the data.

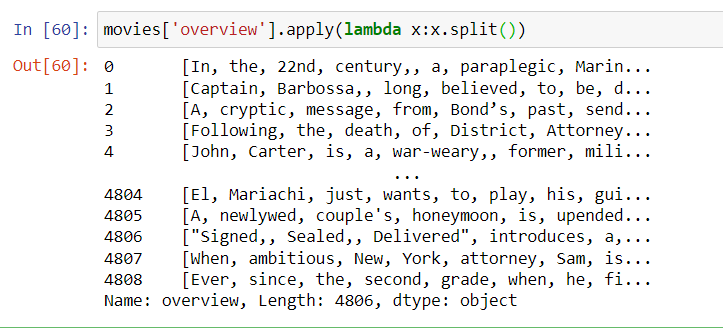
1. Removing spaces

We will have to transform the data such a way that there has to be no spaces between the words so that the tags can be created without any confusion. For that we will have to use the list comprehension function with the help of lambda where we will be replacing the spaces with nothing.

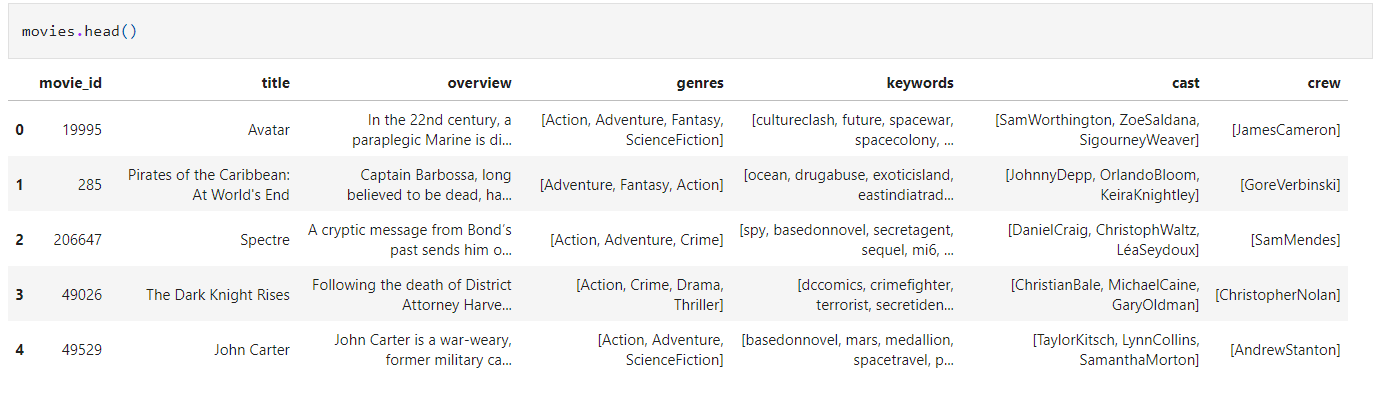


We will be applying the function to the first column ‘genres’ and storing it in the genre column.

Similarly, with all the other columns ‘keywords’, ‘cast’, and ‘crew’ we will be applying the same function to remove the spaces.



After removing the spaces, we get the actual format of the data what we need for creating tags by concatenating all the columns into one column and we will name the column as tags.

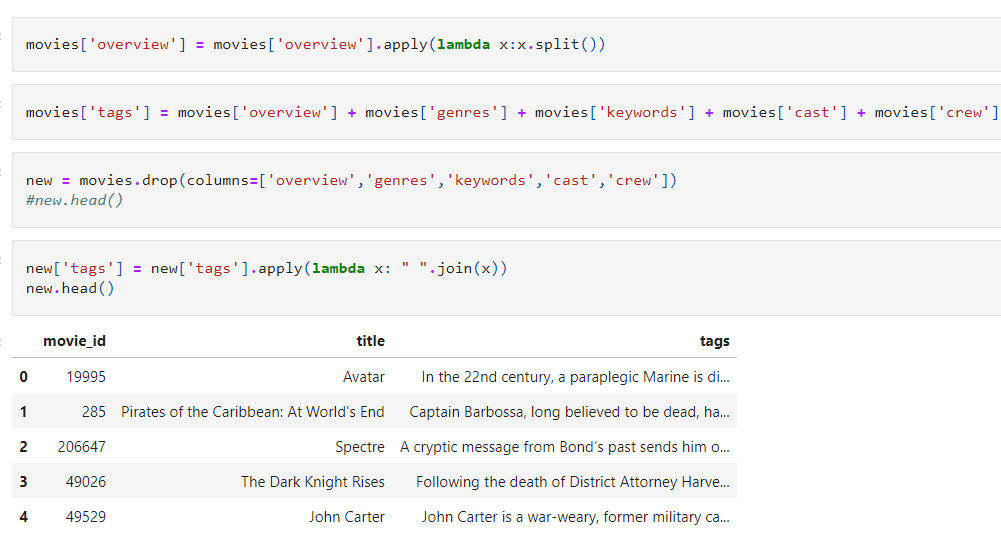


Now after concatenating the data into a new column that is tags column, we see that the data is all in list, now we will convert the data from list to string by creating a new data frame and using the lambda function.

**New Data Frame**

We will be only having 3 main columns which are ‘movie\_id’, ‘title’, and ‘tags’ where the movie id is the id provided to the movies by TMDB, we are considering it in the table because when we deploy the model that is after website building, we can use these ‘movie\_id’ to call back the posters for the movie to highlight the project. Title is the main title of the movie and tags are nothing but the description of the movie which are helpful while finding the similarities between the movies and recommend similar movies.

But before that the best suggestion for tags is to be in lower case letters, so we will have to convert the data into lower case.



**Vectorization**

For the tags we will have to convert the data into vectors format using the vectorization class available on scikit learn.

For example, in a multi-dimensional array, vector is nothing but a co-ordinate. So, what we will be doing is converting all the movies in the form of vectors and plotting them on a multidimensional space and when a movie name is passed then the closest vector available in the space to that movie vector are returned back to the user recommending the similarity of the movies.

For that we will be creating a new string by concatenating all the data available in the tags column, which results in getting a huge string data and now from that we will be extracting the most frequently used 5000 words by calculating the frequency of each word in the huge string.

Now when we get the most frequently used 5000 words we can build a matrix where we can see how many times the set of 5000 words are used in the tags of 5000 movies. Which will result us to in a vector form of the movies in 5000-dimensional space.



The most important thing while getting 5000 words we shall ignore the ‘stop words’ from the huge data frame. Stop words are nothing but the words which are used for sentence formation and they don’t possess any contribution in the meaning of the sentence.

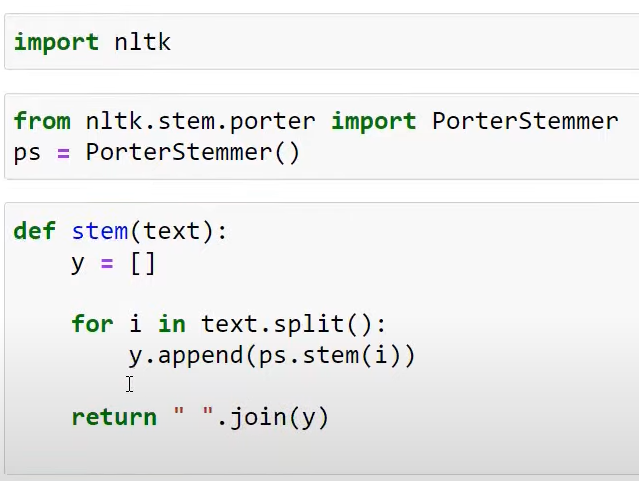
We will be using class which are already available in the python libraries. We will be extracting class from scikit-learn called ‘count vectorizer’ which have many hyper parameters we can use where we are going to use ‘max\_features’ to set .5000 words limit and ‘stop\_words’ will be set to English.

We will be converting the matrix into a explicitly NumPy array because many of the values will be 0, and if we talk about the count vectorizer, by default it will return in sparse matrix which we will have to convert it into NumPy array.

**Applying Stemming**

Now when we see the features that is the 5000 words, we can see same words in different tenses like for example, we can see that there are 2 different words for ‘act’ and ‘acting’ which basically have to same, so here we apply stemming on the string.

For applying stemming on the string we will have to import it from ‘nltk’ library where we will be importing ‘PorterStemmer’ which is one of the most frequently used NLP technique. We will have to process stemming on the whole tags column to remove the repetition of words.



**Calculating the distance between the vectors**

Now when we have vectors for all the movies in a 5000-dimensional space and now we can calculate the distance between the vectors, so that less the distance more the similarity and more the distance less the similarity between the vectors.

While calculating the distance between the vectors, we will not be calculating the ‘Euclidean distance’, rather we will be calculating the ‘cosine’ distance between the vectors. ‘Cosine’ distance is calculated by finding the angel between the vectors, that is less the angle that means the similarity is more.

‘Euclidean distance’ is not a good way to measure the distance between the vectors when we are working on a multi-dimensional vector.

**Using Cosine Similarity**

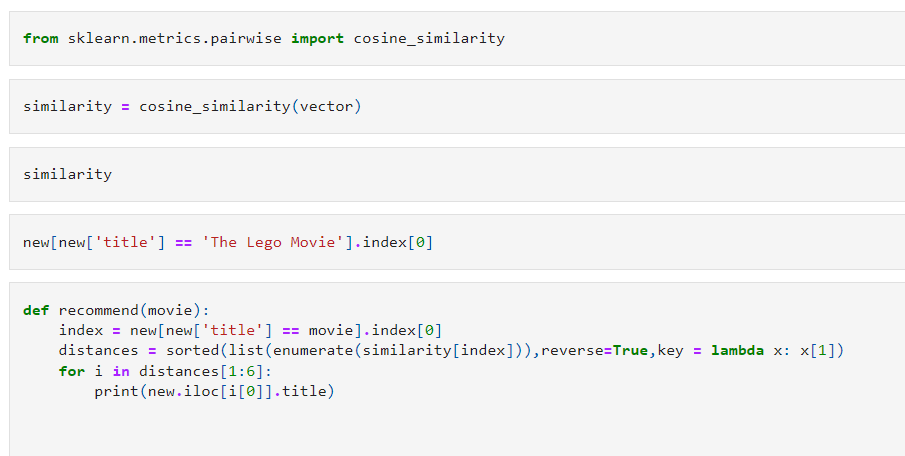
We will be using Cosine similarity function on the vectors of the movies by calculating the distance for each movie with the rest of the movies to get the similarity array. We can see that the diagonal of the matrix will always be 1 because the distance between the vector itself will always be shown 1.

**Creating a Function**

Now we will be creating a function where the user can give a movie name and in return the model will have to recommend the user 5 movies with similar contents.

Here, first when the movie is given we will have to find the index of the movie from the data then with the help of the index we will enter the similarity matrix and we will extract vector using the index.

To make it simpler, we will create and save the index into list of tuples where we will create the index position with distances in the similarity matrix so that when we apply sorting on the similarity matrix to get the 5 best score and make sure while sorting the data we have to sort on the basis of similarity which is the 2nd column in the list.



We will now have to create a function where we will have to give a name of the movie which will then check for the index and store the index which will then check the distances with the help of similarity function which will revert back the 5 similar movies recommended to the user. But here it will revert back to us with the index of the movie which is not what we need.

So now we have to fetch the name of the movie with the help of index, for that we will create a for loop in the above function itself so that when it reverts back with the index it has to fetch the title of the movie.

**CONCLUSION**

To conclude, a recommender system powered by content-based filtering performed using the cosine similarity algorithm can make better recommendations for users by suggesting them movies that have similar key features. Just let a user enter a movie title and the system will find a movie which has most similar features. As we will be sorting the scores in descending order we can retrieve the 5 highest similarity scored movies and return to user.

**References**

1. Aggarwal, C.C., 2016. Content-based recommender systems. In *Recommender systems* (pp. 139-166). Springer, Cham.
2. Musto, C., Semeraro, G., Gemmis, M.D. and Lops, P., 2016, March. Learning word embeddings from wikipedia for content-based recommender systems. In *European conference on information retrieval* (pp. 729-734). Springer, Cham
3. 3. Volkovs, M., Yu, G.W. and Poutanen, T., 2017. Content-based neighbor models for cold start in recommender systems. In *Proceedings of the Recommender Systems Challenge 2017* (pp. 1-6)
4. Blanco-Fernandez, Y., Pazos-Arias, J.J., Gil-Solla, A., Ramos-Cabrer, M. and Lopez-Nores, M., 2008. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics*, *54*(2), pp.727-735
5. Chen, H.W., Wu, Y.L., Hor, M.K. and Tang, C.Y., 2017, July. Fully content-based movie recommender system with feature extraction using neural network. In *2017 International conference on machine learning and cybernetics (ICMLC)* (Vol. 2, pp. 504-509). IEEE