Content-based Movie Recommendation System

Abstract

At the time of putting this together it is 2020, the corona virus is running wild, we are all confined to the four corners of our homes(and screens) and boredom is the order of the day. As a result of this, being avid movie watchers we will build a movie recommender system that will cater to our taste of movies and provide us with personalised entertainment during this period.

- The system should allow a user to give an example of the movie the user enjoys (This is to create a user profile)
- Using the content based algorithm the system should return suggestions that the user might like.

Introduction

Recommendation engines are everywhere today and have been in the background having profound impacts on our lives . They are behind some of our favourite apps such as netflix ,youtube, and even amazon. Essentially, they are the silent salesmen we don't know or see . But what are they ? What is their relevance to this project ? and How do they work ? .

In very plain language they are systems that give users suggestions on movies they might like or content they might like based on their behaviour or the behaviour of other similar users . This behavioural data comes in numerous forms and can either be implicit or explicit some examples of these are,movies purchased , ratings giving to content or items , sentiments of user towards a given item , scrolling activity or even content you viewed.The basic question asked when getting data for such systems is , how does one behave when they like or dislike some content or movie.

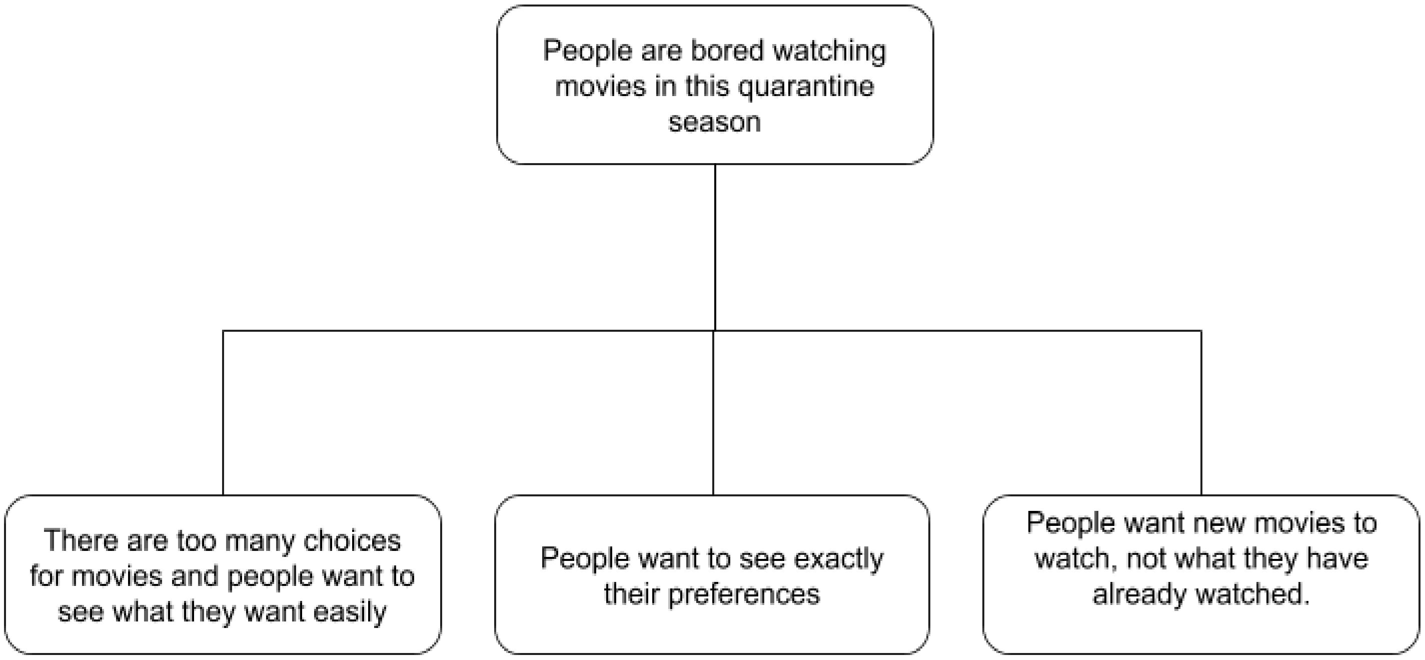
Generally speaking as humans we generally know what we like but we might not always know why we like what we like and hence in a situation where there is an abundance of choice , choice paralysis can easily set in. Once that happens we might just once again go into a cycle of boredom and end up unhappy. As a result of this the relevance of this project is to eliminate choice paralysis and provide enjoyable content to the user.

There are two main ways to figure out what someone likes. The first is by them openly telling you what they like or secondly by observing the likes of their closest friends. Based on this analogy there are two types of recommendation systems . The content based recommendation system or the collaborative recommendations system .

The content based system creates a **user profile**(a description of the users likes or dislikes) based on the users activities and compares it to a **movie profile**(important metadata about a movie). For example, let's say a user named Arun gave a 5 star rating to an action movie featuring Khan. We can infer from this that Arun likes action movies and the actor Khan and would hence suggest action movies or movies where khan is involved . This is an overly simplified example however it captures the general idea of a content based system.

The collaborative system is built around the idea of similarities in user profiles. A system like this seeks to deduce the way user A(who has a similar user profile to user B) will rate an item based on the way user B rated the item.

Problem Breakdown


People are bored watching movies in this quarantine season
There are too many choices for movies and people want to see what they want easily
People want to see exactly their preferences
People want new movies to watch, not what they have already watched.

Our project will focus primarily on solving the problems
Before jumping into data structures or the algorithms that will be used in this project it is important to ask the most important question.
What type of recommendation system will be used and why?The system will be a content based recommendation system due to the need for a highly personalized experience. This option was also selected based on the skill level of the programmers .

System performance criterias

- <u>Individual user preferences:</u>
  The system should be able to identify a user’s preferences based on the users favourite movie.

- <u>Suggest Preferred movies:</u> The system should be able to provide a list of movies on the user’s preference.

- <u>Suggest New Movies</u>: The system should be able to provide movies which have not yet been seen by the user and are similar to the users preference.

If the above structures are incorporated, the user should be able to:

1. Receive accurate movie suggestions.
2. Have movies catered to their personal taste.
3. Receive suggestions that may not necessarily be popular movies but should be interesting.

Problem Interpretation

Data objects within our project**:** *Movie , User profile*

<u>Attributes</u>

Movies: movie *Id, title, overview*

The movie object will be used to store the features of the movie mentioned

User: *Name, movie id (Users favourite movie id)*

This object will store the features of each user within the system.

<u>Relationship Attribute</u>

Each user should be able to store which movie most interests them.

<u>Method</u>

add_user(): Creates a new instance of a user

remove_user (): Remove user details from

movie(): This method retrieves movie information from the db(CSV file)

recommend(): Gives the recommended movie after the movie_id of the user has been inputted.

<u>Other Functions needed.</u>

- Creation of term frequency-inverse document frequency(tfidf) matrix based on the "overview" feature of movies.
- Generation of cosine similarities based on tfidf matrices
- Sorting according to cosine similarity values.

Constraints **:**

1. Operations done by the objects should to be at most O(n)
2. Accessing from the database should be fast hence O(1)
3. Adding a user profile should be fast hence O(1)
4. Deleting a user profile should be fast hence O(1)
5. The storage should be limited: Space complexity should be O(m+u) where m is the movies and u the users.

Data structure requirements and considerations

- A data structure that allows us to uniquely store movies or the use profile it should permit insertion, quick access

- The data structure used must be efficient in storing the data.

- A data type that makes sorting relatively easy since we will be sorting the movies by similarity to the user preference

- A data type that can easily be analysed in the form of matrices(since we will try to find cosine similarity using tfid matrix)

Considerations

Stability of data: The Data is extremely stable given that the main information about movies doesn't change.

Order ,Duplicates: Order is not of much importance since this system doesn't use any other features for movies apart from the movie description and id(basically the movies are not stored in any order).

Simplicity of implementation: Giving our current status of novice programmers and time constraints we will need to easily implement the data structure building around the idea of

Quality and format of data set: The data we will be processing will need to have a consistent format and should contain minimal errors and gaps.

Solution Generation and implementation

Based on all the above considerations we came up with the solution below

Solution walkthrough
The user gets access to the system(it is open to anyone authentication not required)
(since it is personal no login is required)

The user profile will then be created , this will happen by taking all the features from the movie the user selects as their favourite .

Using the pandas library and sklearn kit the “overview” feature of the object movie will be analysed , the basic idea behind this is that we need to convert string into numerical values to analyse them algorithmically. We do this by creating the term frequency-inverse document frequency(tfidf) matrix. What this means is that we are able to find out relevant words within each string. It does this by weighting words based on frequency meaning the less times a word appears the more relevant it is.(Usually , pronouns, conjunctions and other non relevant words are not looked at)’

A tfid matrix is created and finally using cosine similarity we compare the tf idf matrix to itself.

Finally we compare the users preference to all the movies in the DB and sort according the movie with the highest similarity score . and generate at least 5 similar movies.

Solution implementation

<u>Operations on csv file</u>

Data is gotten from a csv file this file will contain all the movie metadata needed to go about the operations .

The movies that are used in the system(recommended movies) are stored as dictionaries after they are retrieved from the database. This operation would have a complexity of O(m) where m would be the number of movies. O(m) will come about because we will need to move through the entire DB.

Also in implementing the user profile we used a dictionary , The following reasons are why a dictionary was selected as the data structure of choice for both the storage of the user profile and movies .

<u>Choosing a data structure</u>

During the brainstorming process, Sets,Tuples and dictionaries were considered.
Sets were immediately discarded due to the fact that storing key value pairs were going to be a bit challenging.
Tuples were also not selected because it is not mutable making updating the user profile problematic

Dictionaries fit in perfectly with O(1) access speeds and the use of key value pairs .
In this case we created a user dictionary and movie dictionaries based on a key-value pair name ,movie-id and id, movie,title and overview.

Also a the movies were initial drawn from the database as lists and then the recommended movies were converted to dictionaries to ease operations

Summary of DS used per method

Methods and func	**Data structure**	**Time complexity**	**Space complexity**
add_user()	Dictionary {user_id: (username,movie_id)}	Worst case O(1)	O(1)
remove_user()	Dictionary {user_id: (username,movie_id)}	Worst case:O(1)	O(1)
movie()	list[(id, name, overview)]	Worst Case: O(m)	O(m) where m is the number of movie objects added to the program from the csv file
recommend()	Dictionary { id : (name, overview) }	Worst case:O(1)	O(n) where n is number of recommended movies

Conclusion

<u>Does the app solve the problem ?</u>
Well from the functionality point of view all the objectives for which the app was created were answered. Using a large DB of movies, good similar movies were easily found and displayed to the user .

<u>Is the interface friendly to use?</u>
While the solution does offer practical answers it lacks some level of polish to the UI or UX and
It is an area that can be improved upon in later variations of this project.
We assumed that the user profile had already been recorded.

<u>FInal thoughts</u>.
A content based system is excellent at catering to a users unique taste however, It does not take into account quality input other users can give. Also the user profile contains just one movie preference. This may not be enough to actually fully describe a users personal taste and should be increased in future versions of this program.

Finally, the program allowed us to not only delve into the creation of a solution but the creation of the most efficient solution that would cost the least resources and find the best balance in performance and other costs . It challenged the way we saw writing algorithms and will serve as a framework for our future projects.

By David Norman Amatey Masoperh and Wendy Essuman