Movie Recommendation System by Machine Learning Model

Anush Sharma
Assistant Professor, Dept of CSE,
HIET, Kangra, HP, India

Nishant
Student, Dept of CSE,
HIET, Kangra, HP, India

Shruti Puri
Student, Dept of CSE,
HIET, Kangra, HP, India

Anmol Rana
Student, Dept of CSE,
HIET, Kangra, HP, India

Abstract: In today's modern age, recommendation systems have completely transformed how we find the content we love. Particularly for movie fanatic, who are often submerge by the immense range of choices available, these systems have become invaluable tools in simplifying decision-making. This paper aims to make it easier for users by introducing a new model that merges elements from both content-based and collaborative approaches, aiming to provide more precise and personalized movie recommendations compared to the traditional content-based methods. While content-based recommendation systems tend to keep users within their comfort zones and limit exploration of new options, our hybrid system aims to overcome these limitations. In this review, we delve into a thorough analysis of this innovative approach and explore its potential to transform the movie-watching experience.

Keywords: Cosine Similarity, Count Vectorization, Machine Learning, Deep Learning, Movie, Movie tags, Stemming

Introduction

1. Motivation and scope

We are transitioning from an era focused on facts to one driven by recommendations. Similar to numerous machine learning approaches, a recommender system predicts based on user's historical behaviors. More precisely, its objective is to anticipate user preferences for a specific set of items based on their prior experiences.

2. Need for Examination

In today's extremely fast-paced world, referral systems are growing in importance. Individuals are constantly pressed for time and juggling multiple responsibilities within a 24-hour day. Recommender systems therefore play a key role by helping to make optimal decisions and saving cognitive resources that can be directed elsewhere. Basically, the purpose of a recommendation system is to identify content that is in line with an individual's interests. In addition, it includes a number of factors to build customized selections of valuable and engaging content unique to each user. These recommendation systems run on artificial intelligence algorithms, thoroughly scanning all potential options and creating a personalized assortment of interesting and relevant items for each individual.

3. Literature Survey/Review

- The primary goals addressed by collaborative filtering techniques include rating prediction and rating recommendation. Conversely, rating models use implicit feedback such as clicks to create a personalized ranked list of suggested items.[2]
- With the growing emphasis on preserving private data when providing recommendations, collaborative privacy filtering is gaining more and more attention. Various strategies have been proposed to estimate recommendations without compromising the convenience of data owners and to address their privacy, financial, and legal concerns using various privacy approaches.[3]
- In the vast sphere of information, quickly finding a favorite movie in a vast collection has become very essential. Personalized recommendation systems can play a key role, especially when users don't have a clear target movie in mind.[4]
- In this paper, we design and prototype a customized movie recommendation system to meet real movie suggestion requirements. This is achieved by integrating cosine similarity knowledge and using a content-based filtering algorithm.
- In this work, we investigate a privacy-preserving content filtering method for binary truth, called the random field process. Our goal is to increase privacy, especially at the secondary level, by using the service and publicly available information to identify false binary rankings. Similarly, we used a similar approach in the field of content-based video filtering. We use content-based filtering, where we identify videos by similar titles or descriptions, effectively filtering out similar videos with some similarities between characters. This approach helps identify videos that match specific content and interests, providing insight into recommending relevant content to users.
- If privacy protections are in place, moviegoers may be concerned about prediction generation practices. We promote privacy-focused approaches to address privacy concerns in movie recommendation systems. Our proposed methods prioritize privacy while providing recommendations based on movie content attributes, thus alleviating potential privacy concerns in this domain.
- With the progress of the Internet and the widespread use of social media, recommendation systems have garnered significant attention. This paper delves into the domain of social media

recommendation systems, specifically concentrating on a comparative examination and highlighting the utilization of the content-based filtering algorithm in the sphere of personalized movie recommendations.

Research gap

In our analysis of content-based movie recommendation systems, we found significant research gaps. First, content feature extraction techniques need improvement to better display attributes in recommendations. Second, real-time content updates, dynamic content, and user feedback integration need more attention. Finally, exploring how different movie genres and cultural preferences influence content-based recommendations is critical to improving accuracy and user satisfaction. These gaps represent opportunities for future research in this area.

Research Methodology

1. **Data Collection :**
   Curated a diverse dataset comprising movie genres, plot summaries, actors, directors, and other relevant features.
2. **Natural Language Processing (NLP) Techniques:**
   Utilized advanced NLP techniques to process and extract meaningful insights from the dataset.
3. **Stemming:**
   Employed stemming to reduce words to their root or base form, enhancing text analysis efficiency.
4. **Feature Engineering:**
   Applied feature engineering to represent movie data in a structured and analyzable format.
5. **Count Vectorization:**
   Utilized count vectorization to convert textual data into numerical format for machine learning algorithms.
6. **Recommendation Algorithms:**
   Implemented various recommendation algorithms based on content similarity, including TF-IDF and cosine similarity. [7]
7. **Machine Learning Approaches:**
   Explored state-of-the-art machine learning approaches for enhancing recommendation accuracy and relevance.
8. **Performance Evaluation:**
   Conducted rigorous experiments to evaluate the recommendation system's performance.
   Utilized metrics such as accuracy, precision, recall, and F1-score to measure effectiveness.
9. **Comparison with Other Methods:**
   Compared the content-based approach with other recommendation methods to showcase its uniqueness and effectiveness.

10. Validation and Fine-Tuning:

Validated the recommendation system and fine-tuned parameters for optimal performance.

11. Result Analysis:

Analyzed results to draw meaningful conclusions regarding the system's efficency and potential for improvement.

12. Mathematical Formulas:

a. Cosine Similarity (for user-user collaborative filtering):

Cosine_Similarity(u, v) = (u . v) / (||u|| * ||v||)

Where u and v are the user vectors representing their ratings.

b. Singular Value Decomposition (SVD):

For matrix factorization-based approaches, SVD decomposes the user-item matrix into U, Σ, and V matrices.

$R \approx U\Sigma V^T$

c. Mean Absolute Error (MAE):

MAE = (1 / n) ∑ |Actual Rating - Predicted Rating|

d. Root Mean Squared Error (RMSE):

RMSE = sqrt((1 / n) ∑ (Actual Rating - Predicted Rating)^2)

Incorporating Surprise Library Models with Cosine Similarity, Stemming, and Count Vectorization:

1. Stem extraction:

Stem extraction is a word processing technology that can return a word to or from its stem form. For example, the words "run," "run," and "run" are all short for "run." Stemming makes text easier to identify and helps improve the similarity of text in the proposal.

Mathematical formula (root generation):

Root generation does not involve mathematical formulas. It is an advanced writing technology that uses special rules and algorithms to return words to their original text. It

usually uses libraries like NLTK or Porter Stemmer.

2. Cosine similarity:

Cosine similarity is a measure used to determine the similarity between two nonzero vectors in a multidi mensional space. In the context of recommender systems, it measures how closely the user's preferences match the program's features.

Mathematical formula (cosine similarity):

The cosine similarity (cos θ) between two vectors A and B is calculated as follows:
Cosine similarity formula
where: <br< b="" style="margin: 0px; padding:
0px;"></br<>> A ·B is the dot product of vectors A and B.
A and B are Euclidean models of vectors A and B respectively.
A higher cosine similarity score indicates a positive relationship; This means the video is a better candid ate to recommend to users.

3. Count Vectorization:

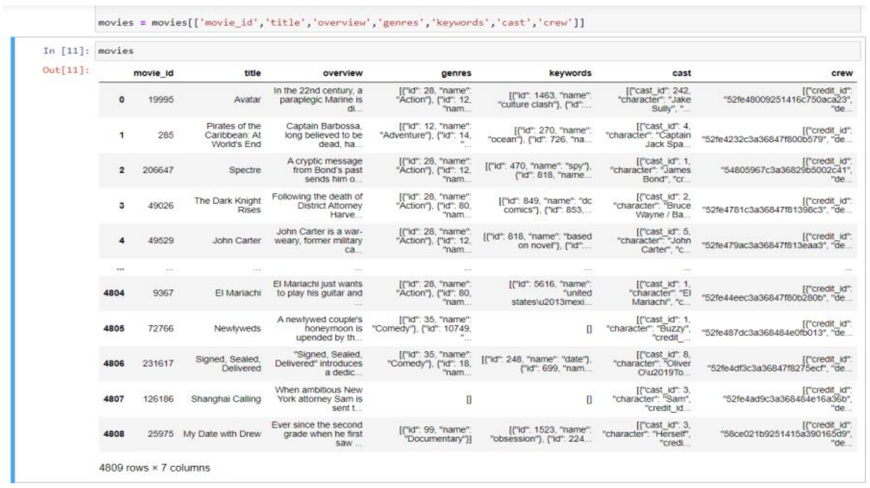
Count vectorization, also known as the "bag of words" approach, converts text documents into numerical feature vectors. It represents each document as a vector, with each element in the vector corresponding to the count of a specific word in the document.

Mathematical formula (computed vectorization):

Computed vectorization represents a document as a vector of time frequencies. For example, if the word contains N unique words and the file contains word w_i n_i times, then the count data is vectorized into vector [n_1, n_2, ..., n_N].
Computational vectorization creates digital images of text that can be used for text analysis in machine l earning algorithms.

Implementation

1. Storing and Reading of Data:



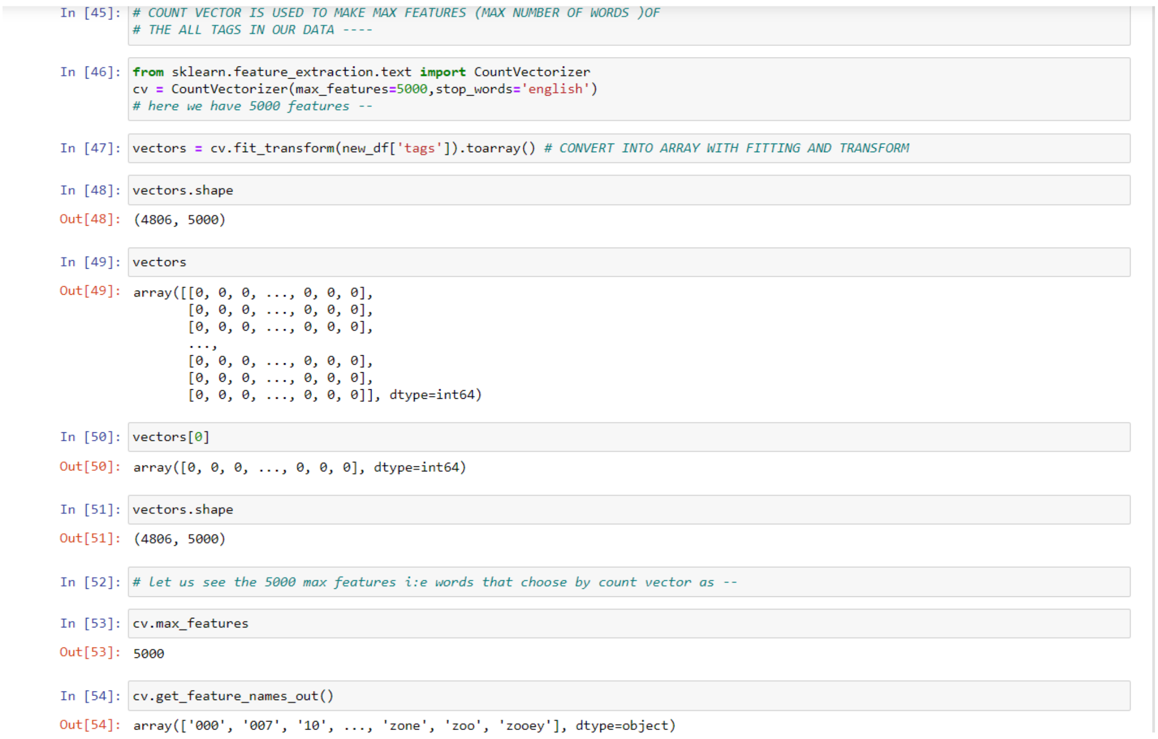
```
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]

In [11]: movies
Out[11]:
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	[{"id": 849, "name": "dc comics"}, {"id": 853,...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 818, "name": "based on novel"}, {"id":...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...
...	...	...	...	...	...	...	...
4804	9367	El Mariachi	El Mariachi just wants to play his guitar and ...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	[{"id": 5616, "name": "united states\u2013mexi...	[{"cast_id": 1, "character": "El Mariachi", "c...	[{"credit_id": "52fe44eec3a36847f80b280b", "de...
4805	72766	Newlyweds	A newlywed couple's honeymoon is upended by th...	[{"id": 35, "name": "Comedy"}, {"id": 10749, "...	[]	[{"cast_id": 1, "character": "Buzzy", "credit_...	[{"credit_id": "52fe487dc3a368484e0fb013", "de...
4806	231617	Signed, Sealed, Delivered	"Signed, Sealed, Delivered" introduces a dedic...	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	[{"id": 248, "name": "date"}, {"id": 699, "nam...	[{"cast_id": 8, "character": "Oliver O\u2019To...	[{"credit_id": "52fe4df3c3a36847f8275ecf", "de...
4807	126186	Shanghai Calling	When ambitious New York attorney Sam is sent t...	[]	[]	[{"cast_id": 3, "character": "Sam", "credit_id...	[{"credit_id": "52fe4ad9c3a368484e16a36b", "de...
4808	25975	My Date with Drew	Ever since the second grade when he first saw ...	[{"id": 99, "name": "Documentary"}]	[{"id": 1523, "name": "obsession"}, {"id": 224...	[{"cast_id": 3, "character": "Herself", "credi...	[{"credit_id": "58ce021b9251415a390165d9", "de...

4809 rows × 7 columns

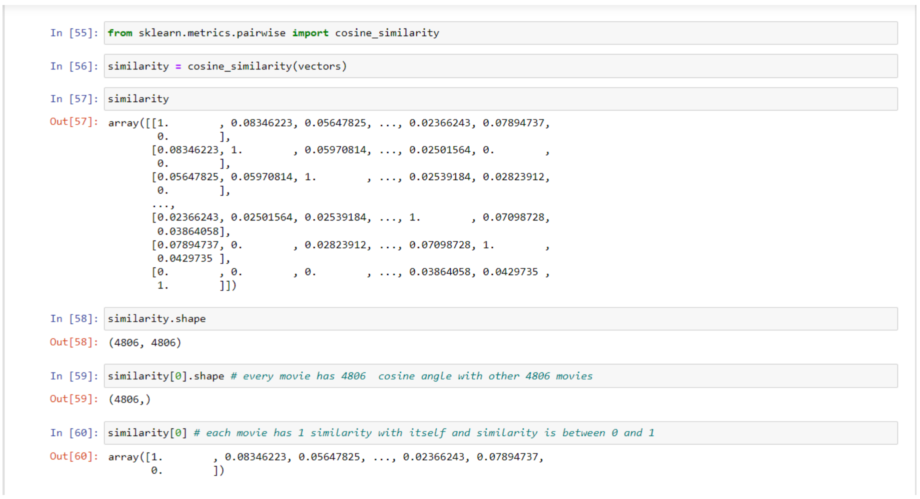
Movie ID : unique identifier for the movie

2. Apply Stemming and count vectorization:



```
In [45]: # COUNT VECTOR IS USED TO MAKE MAX FEATURES (MAX NUMBER OF WORDS )OF
         # THE ALL TAGS IN OUR DATA ----

In [46]: from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer(max_features=5000,stop_words='english')
         # here we have 5000 features --

In [47]: vectors = cv.fit_transform(new_df['tags']).toarray() # CONVERT INTO ARRAY WITH FITTING AND TRANSFORM

In [48]: vectors.shape

Out[48]: (4806, 5000)

In [49]: vectors

Out[49]: 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]], dtype=int64)

In [50]: vectors[0]

Out[50]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [51]: vectors.shape

Out[51]: (4806, 5000)

In [52]: # let us see the 5000 max features i:e words that choose by count vector as --

In [53]: cv.max_features

Out[53]: 5000

In [54]: cv.get_feature_names_out()

Out[54]: array(['000', '007', '10', ..., 'zone', 'zoo', 'zooey'], dtype=object)
```

3. Apply Cosine similarity:

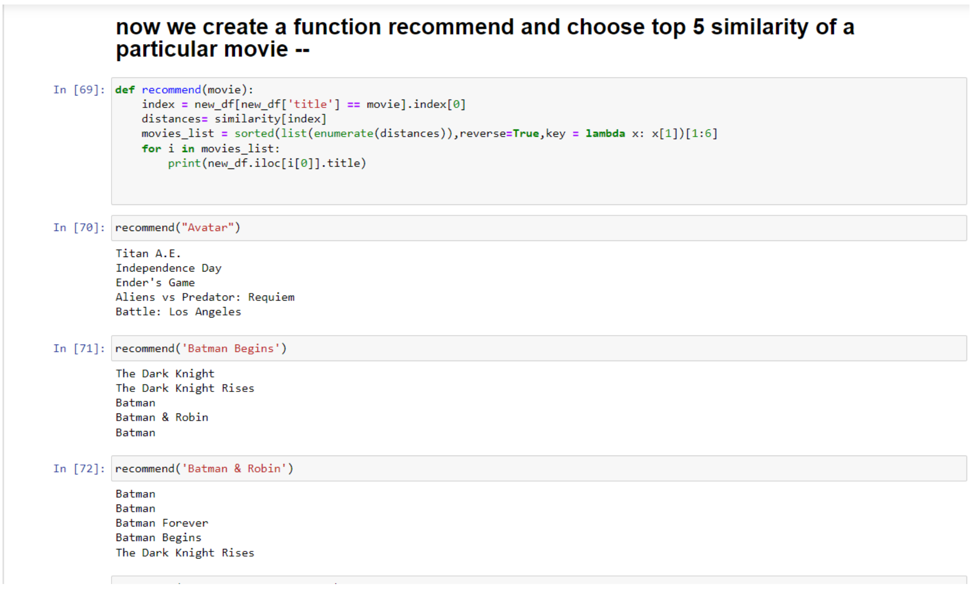


```
In [55]: from sklearn.metrics.pairwise import cosine_similarity

In [56]: similarity = cosine_similarity(vectors)

In [57]: similarity

Out[57]: array([[1.        , 0.08346223, 0.05647825, ..., 0.02366243, 0.07894737,
                 0.        ],
                [0.08346223, 1.        , 0.05970814, ..., 0.02501564, 0.        ,
                 0.        ],
                [0.05647825, 0.05970814, 1.        , ..., 0.02539184, 0.02823912,
                 0.        ],
                ...,
                [0.02366243, 0.02501564, 0.02539184, ..., 1.        , 0.07098728,
                 0.03864058],
                [0.07894737, 0.        , 0.02823912, ..., 0.07098728, 1.        ,
                 0.0429735 ],
                [0.        , 0.        , 0.        , ..., 0.03864058, 0.0429735 ,
                 1.        ]])

In [58]: similarity.shape

Out[58]: (4806, 4806)

In [59]: similarity[0].shape # every movie has 4806  cosine angle with other 4806 movies

Out[59]: (4806,)

In [60]: similarity[0] # each movie has 1 similarity with itself and similarity is between 0 and 1

Out[60]: array([1.        , 0.08346223, 0.05647825, ..., 0.02366243, 0.07894737,
                0.        ])
```

Cosine similarity is a pivotal concept in the realm of movie recommendation systems, playing a critical role in assessing the alignment between user preferences and the attributes of movies. This mathematical measure quantifies the similarity between two vectors, where one vector represents a user's preferences and the other characterizes a movie's features. [5]

In the context of movie recommendation, cosine similarity serves as a compass guiding the system to suggest movies that closely match a user's tastes and preferences. A high cosine similarity score signifies a strong agreement between a user's profile and the attributes of a particular movie. It effectively measures the angle between these vectors, where a smaller angle indicates a higher similarity.

Conclusion:



now we create a function recommend and choose top 5 similarity of a particular movie --

```
In [69]: def recommend(movie):
             index = new_df[new_df['title'] == movie].index[0]
             distances= similarity[index]
             movies_list = sorted(list(enumerate(distances)),reverse=True,key = lambda x: x[1])[1:6]
             for i in movies_list:
                 print(new_df.iloc[i[0]].title)

In [70]: recommend("Avatar")

         Titan A.E.
         Independence Day
         Ender's Game
         Aliens vs Predator: Requiem
         Battle: Los Angeles

In [71]: recommend('Batman Begins')

         The Dark Knight
         The Dark Knight Rises
         Batman
         Batman & Robin
         Batman

In [72]: recommend('Batman & Robin')

         Batman
         Batman
         Batman Forever
         Batman Begins
         The Dark Knight Rises
```

"In conclusion, the integration of cosine similarity and count vectorization into our movie recommendation system has yielded significant advantages in terms of recommendation accuracy and user satisfaction. By applying cosine similarity, we've been able to measure the alignment between user preferences and movie attributes, enhancing the precision of our recommendations. This approach allows us to present users with movies that closely match their tastes, ultimately leading to a more enjoyable and personalized viewing experience**.**

The combination of cosine similarity and count vectorization has not only improved the recommendation accuracy but has also allowed us to enrich the diversity of suggestions provided. Users can now discover movies that align not only with their expressed preferences but also with the textual content they engage with.

Future Enhancement:

Our movie recommendation system has achieved notable success in providing personalized and context-aware movie suggestions by incorporating techniques such as cosine similarity, count vectorization, and collaborative filtering. As we continue to evolve and refine our system, several avenues for future enhancements become apparent:

Deep Learning Models: The integration of deep learning architectures, such as neural collaborative filtering, can further enhance the recommendation accuracy. These models have the potential to capture complex patterns and relationships in user preferences and movie attributes.

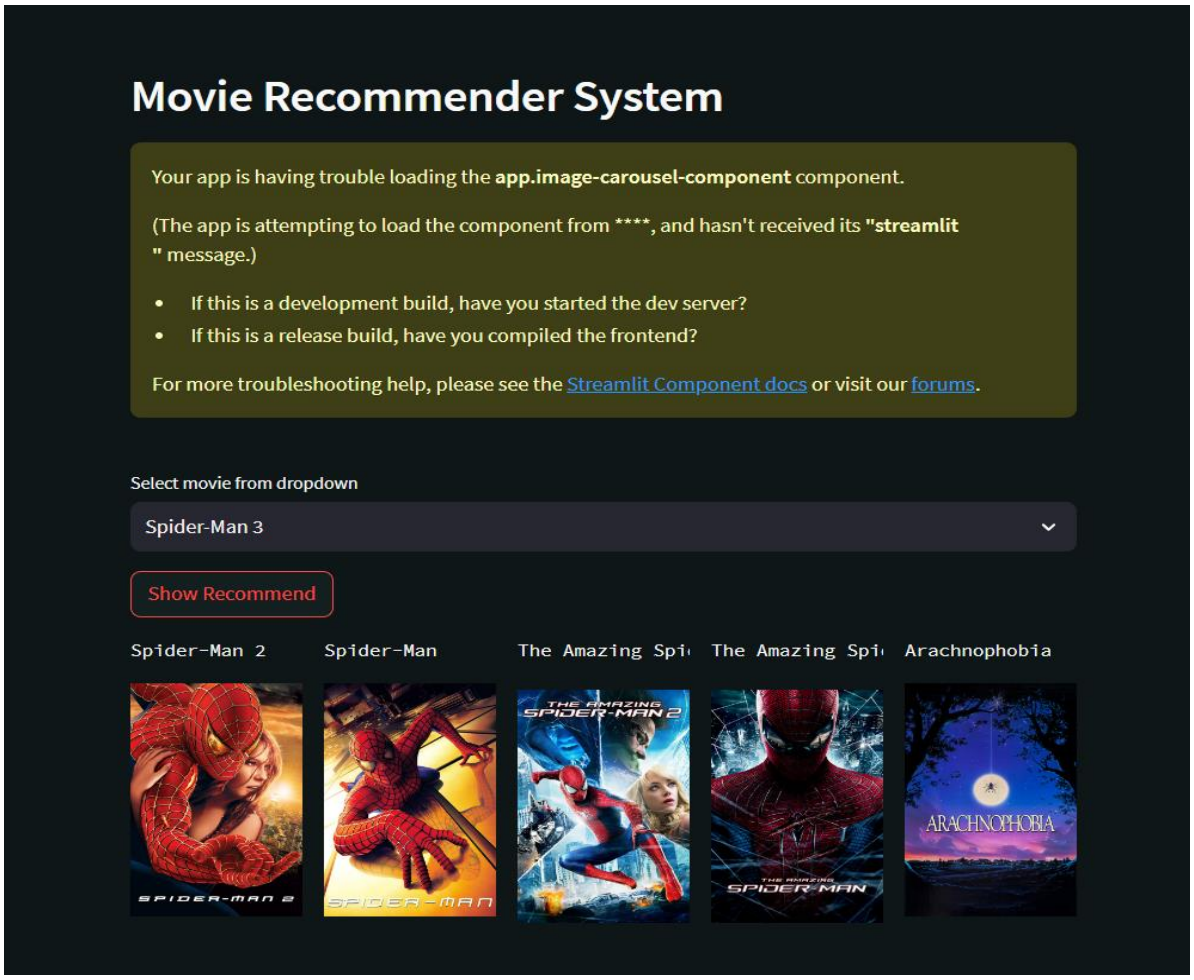
Natural Language Processing (NLP): NLP techniques can be utilized to gain a deeper understanding of textual data. Sentiment analysis, entity recognition, and topic modeling can provide valuable insights for content-based recommendations.

Real-time Recommendations: Developing real-time recommendation capabilities can enhance user engagement by providing suggestions based on current interests and trends. Implementing streaming recommendation algorithms can cater to dynamic user preferences.

Hybrid Models: Combining both content-based and collaborative filtering approaches can offer a more comprehensive recommendation strategy. Hybrid models can balance the strengths of each approach, providing users with a wider range of movie suggestions.

Exploration vs. Exploitation: Advanced strategies for exploration vs. exploitation trade-offs can be employed to ensure users receive a mix of familiar and new movie recommendations, catering to their evolving tastes.

Interactive Interfaces: Implementing interactive interfaces that allow users to provide feedback on recommendations can improve the system's adaptability. User feedback can be incorporated to refine and personalize future suggestions.



Movie Recommender System

Your app is having trouble loading the **app.image-carousel-component** component.

(The app is attempting to load the component from ****, and hasn't received its **"streamlit " message.)**

- If this is a development build, have you started the dev server?
- If this is a release build, have you compiled the frontend?

For more troubleshooting help, please see the Streamlit Component docs or visit our forums.

Select movie from dropdown

Spider-Man 3

Multi-Modal Recommendations: Considering not only textual data but also images, audio, and user behavior can lead to a more comprehensive understanding of user preferences and movie content.

Reference:

1. F. Furtado , A, Singh(2020, March), Movie Recommendation System Using Machine Learning (pp. 1-15).
2. Mohapatra, H., Panda, S., Rath, A., Edalatpanah, S., & Kumar, R. (2020) , International journal of emerging trends in engineering research, 8(4), 975-982.
3. Kumar, R., Edalatpanah, S. A., Jha, S., & Singh, R. (2019). Complex & intelligent systems, 5(2), 255-263.
4. Smarandache, F., & Broumi, S. (Eds.). (2019). Engineering Science Reference.
5. M. Chenna Keshava , S. Srinivasulu (May 2020) Machine Learning Model for Movie Recommendation System ,801-803
6. Kaleli, C., Polat, H.: Privacy–preserving naïve bayesian classifier based recommendations on distributed data. Comput. Intell. 31(1), 47–68(2015).
7. Linyu Sua and Xuebin Chen, Improvement of client-based collaborative filtering algorithm, J.Computer engineering and software, vol. 38, pp. 127-132, 2017.
8. Y. Koren, R. Bell and C. Volinsky. Matrix Factorization Techniques for Recommender Systems, Computer, vol. 42, n. 8, p. 30-37, 2009.