

**CMPSC 445: Applied Machine Learning in Data Science**

**(Spring 2022)**

**Movie Recommendation**

Dharmik Patel, Love Patel, Stephen Brown, Shiv Patel

Computer Science

Vinayak Elangovan

Submitted On: May 1, 2023

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| 1 | ABSTRACT & INTRODUCTION | 5 |
| 2 | PROBLEM STATEMENT | 5 |
| 3 | BACKGROUND & RELATED WORK | 6 |
| 4 | DESIGN | 8 |
| 5 | RESULTS | 10 |
| 6 | CONCLUSION | 14 |
| 7 | REFERENCE | 14 |

**Abstract**

The project utilizes the TMDB5000 data set to generate meaningful and interesting insights and then create a movie recommendation model based on key features. Some of the key features that will be used to predict movies which the user may be interested in are director name, writer name, run time, genre, and plot (key features). This will be done mainly using the similarity matrix method and algorithms learned in class to compare the performance of each method. This program is content based, so if a user watches one movie, similar movies are recommended. The input for building a content-based recommender system is movie attributes.

**Introduction**

Welcome to our group project on Movie Recommendation System, where we will explore the intricacies of this system and the technology behind it. As a movie enthusiast, one often finds oneself in search of similar movies to watch, based on actors or directors. This is where the movie recommendation system comes into play. Our project focuses on a front-end web-based design that utilizes a similarity matrix to suggest movies to users based on their input. With a pre-existing data file of over 500 movies, our system recommends movies based on similarities between the user's input and the content of the recommended movies. The Movie Recommendation System comprises multiple aspects, including Movies & TV Shows Content-based recommendation, which suggests movies and TV shows based on the user's content preferences and watch history. Additionally, our system caters to User Preferences, enabling users to input their movie choices, and recommends items based on explicit preferences and previous interactions. Our system utilizes a similarity matrix, which is generated through Data Collection, Feature Extraction, and Recommendation Generation techniques. The backbone of our project is the Similarity Matrix, a powerful tool that enables us to mark similarities between the user's input and the content of the recommended items. Our system is designed to read movie information from a CSV file, sourced from the TMDB5000 data set available online. Overall, our group project aims to provide an in-depth understanding of the Movie Recommendation System and its various components. Through this project, we hope to showcase the capabilities of the Similarity Matrix and its potential applications in other domains.

**Problem Statement**

In today's digital era, the entertainment industry is continuously growing, with thousands of movies and TV shows being released every year. As a result, users often find themselves overwhelmed with options and unable to decide what to watch next. Additionally, users may have specific preferences in terms of actors, directors, genres, or even production studios, making it challenging to find movies that meet their criteria. Our solution will incorporate machine learning algorithms that analyze user data, such as viewing history, genre preferences, and rating history, to provide relevant and personalized recommendations. Our proposed solution comprises a Web Front End Application that will allow users to input their movie preferences and receive personalized movie recommendations instantly. The application will incorporate a similarity matrix that matches user input with movie data to generate recommendations that align with the user's preferences. Through this project, we aim to showcase the capabilities of machine learning in addressing real-world problems, specifically in the entertainment industry. By providing users with relevant and personalized movie recommendations, we hope to enhance their viewing experience and improve customer satisfaction.

**Background**

Recommendation systems have become a vital application area for personalized product recommendations in machine learning, and content-based recommendation is a popular method that suggests items to users based on their past interactions with similar items. The method typically involves data collection, feature extraction, user profile creation, and recommendation generation. Our movie recommendation system is based on the content-based recommendation method and uses machine learning and natural language processing techniques to gather data from various sources like the TMDB5000 dataset and the OMDBAPI. The system then extracts key features of each item and matches them with user input to generate a ranked list of recommendations based on the similarity between the user input and the item features calculated using a similarity matrix. Although user profiles are typically used in this method, our system is hosted locally and doesn't require a user to sign up, making it more convenient for users.

The similarity matrix plays a crucial role in our content-based recommendation system, which relies on identifying similarities between items to generate personalized recommendations for users. Mathematically, the similarity matrix is a square matrix where each element represents the similarity score between two items. The diagonal elements of the matrix represent the similarity score of each item with itself, which is always equal to 1. The similarity score is calculated by comparing the features or attributes of each item, such as actors, directors, genres, or ratings. Various mathematical techniques, such as cosine similarity, Euclidean distance, or Jaccard similarity, can be used to calculate the similarity score. Once the similarity matrix is generated, it can be used to find similar items to a user's input and generate personalized recommendations based on their preferences. For example, our movie recommendation system can identify other movies similar to the user's input by using the similarity matrix and recommend them to the user. Overall, the similarity matrix is a powerful tool in machine learning that enables us to identify similarities between items and can be widely used in recommendation systems to provide personalized recommendations to users.

The Similarity Matrix for our system uses multiple types of vectorizer. The first vectorizer is called the Tfidf-Vectorizer, which stands for Term Frequency – Inverse Document Frequency. This technique is used to score the importance of every word in a document amongst a collection of documents. The idea is that if the word *w* is frequent in document *a* but not frequent in all the other documents in collection *D*, then word *w* is important in the document *a.* This tool produces a vector whose values represent the importance of the words in each document*.* The second vectorizer is the Count-Vectorizer, which is a tool that produces a vector whose values are based on the frequency of each word in all the documents. The main idea is to map the most and least frequent words. The final vectorizer that makes up the Similarity Matrix is the Hash-Vectorizer, which is a vectorizer that applies a hashing function to the word frequency in each document. It's a variation of the count vectorizer. It also removes stop words.

**Related Work**

[Using Cosine Similarity to Build a Movie Recommendation System | by Mahnoor Javed | Towards Data Science](https://towardsdatascience.com/using-cosine-similarity-to-build-a-movie-recommendation-system-ae7f20842599)

Writer Mahnoor Javed describes a movie recommendation system built using cosine similarity. First, she explains what cosine similarity is and how it can be used to measure the similarity between two items. The article then shows how to use cosine similarity to build a movie recommendation system by creating a matrix of movie ratings and using it to calculate cosine similarity scores between movies. Finally, Javed demonstrates how to use these scores to make movie recommendations for a user. The article provides code examples in Python using the Pandas and Scikit-learn libraries.

[Content Based Movie Recommendation System | by Karan Kaul | Web Mining [IS688, Spring 2021] | Medium](https://medium.com/web-mining-is688-spring-2021/content-based-movie-recommendation-system-72f122641eab)

Karan Kaul discusses the implementation of a content-based movie recommendation system. The system uses a dataset of movie plots to extract features from each movie and then uses these features to recommend similar movies to a user. Kaul explains how the system preprocesses and tokenizes the plot data, and how it extracts features using the TF-IDF vectorization technique. He also explains how the system calculates the cosine similarity between movies based on their feature vectors, and how it uses this similarity score to recommend similar movies to a user. Finally, Kaul provides Python code examples using the scikit-learn and Pandas libraries to implement the recommendation system.

[Movie Recommendation Engine with NLP - Analytics Vidhya](https://www.analyticsvidhya.com/blog/2022/01/movie-recommendation-engine-with-nlp/)

Sukanya Bag of Analytics Vidhya discusses the implementation of a movie recommendation engine using Natural Language Processing (NLP). The author explains how to preprocess movie plot data and extract features using NLP techniques such as tokenization, stemming, and lemmatization. Bag also demonstrates how to use the Latent Dirichlet Allocation (LDA) algorithm to identify topics in the movie plot data and assign weights to them. These weights are used to calculate the similarity between movies and recommend similar movies to a user. Finally, the author provides Python code examples using the scikit-learn and Gensim libraries to implement the recommendation engine. The article emphasizes the importance of tuning hyperparameters and evaluating the performance of the recommendation engine using metrics such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG).

[Movies Recommendation System Using Cosine Similarity | by Ameer Hamza | Medium](https://ameerhamza1.medium.com/movies-recommendation-system-using-cosine-similarity-4850d9067511)

Within the article, Ameer Hamza |describes how to build a movie recommendation system using cosine similarity. The author explains the concept of cosine similarity and how it can be used to measure the similarity between two movies. The system uses a dataset of movie ratings and calculates cosine similarity scores between the movies based on their ratings. The author explains how to preprocess the data, normalize the ratings, and calculate the cosine similarity scores using Python code. The system then recommends similar movies to a user based on their ratings and cosine similarity scores. The article also discusses some limitations of the system, such as the cold start problem and the need for a large amount of user data to make accurate recommendations. Finally, the author suggests some possible improvements to the system, such as incorporating content-based filtering or collaborative filtering techniques.

[Create a Personalized Movie Recommendation Engine using Content-based Filtering in Python (relataly.com)](https://www.relataly.com/content-based-movie-recommender-using-python/4294/)

Wriiten by Florian Follonier, this article explains how to build a content-based movie recommendation system using Python. The system uses a dataset of movie metadata, including the title, genres, cast, and crew, to recommend similar movies to a user based on their preferences. The author explains how to preprocess the data, extract features from the metadata using the CountVectorizer technique, and calculate the similarity between movies based on their feature vectors using cosine similarity. The author also demonstrates how to use the system to make movie recommendations for a user based on their input. Finally, the article discusses some limitations of the system, such as the need for a large amount of metadata and the lack of personalization, and suggests some possible improvements, such as incorporating collaborative filtering or hybrid recommendation techniques.

**Design**

Data Analysis

* Csv file



Data is presented in CSV format, consists of information for each movie such as title and plot description.

Design Objectives

* Filter unnecessary data from the datasets and consolidate it into 1 file.
* Calculate the similarity matrix that shows which movies are similar
* Term Frequency Inverse Document Frequency Vectors
* Count Vector
* Hashing Vector
* Recommendation
* Web Front-End

Design Specifications

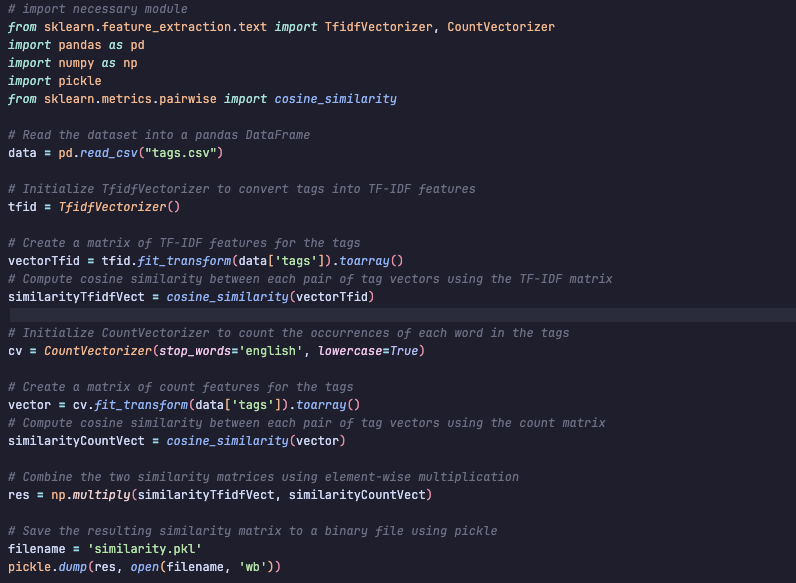
* UI which allows users to search for movies with suggestions displaying automatically. Movie posters are also shown to give users an image as well as text representing the movie.

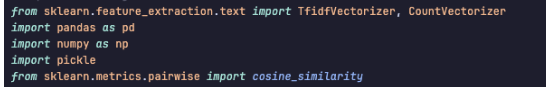
Design Challenges

* Implementing the similarity matrix into a web-based application with data from project files as well as data such as images returned from API calls, solved by using streamlit framework to create application

Design approach (have a diagram to explain the design)

**Results**





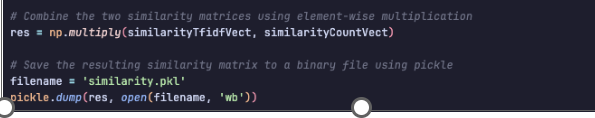
**TfidfVectorizer** and **CountVectorizer** are classes used for converting text documents into numerical feature vectors. **pandas** (**pd**) are a library for data manipulation and analysis. **numpy** (**np**) is a library for numerical computing, providing support for large, multi-dimensional arrays and matrices. **pickle** is a module used for object serialization and deserialization, allowing the model to be saved and loaded. **cosine\_similarity** is a function that calculates the cosine similarity between two vectors.



The Dataset is reading into a pandas DataFrame and the vect is converting into **TFIDE** vect.



Here, the **VecttTfid** is creating a matrix for the tags and **SimilarityTfidvect** is computing the similarity between each pair of tags.

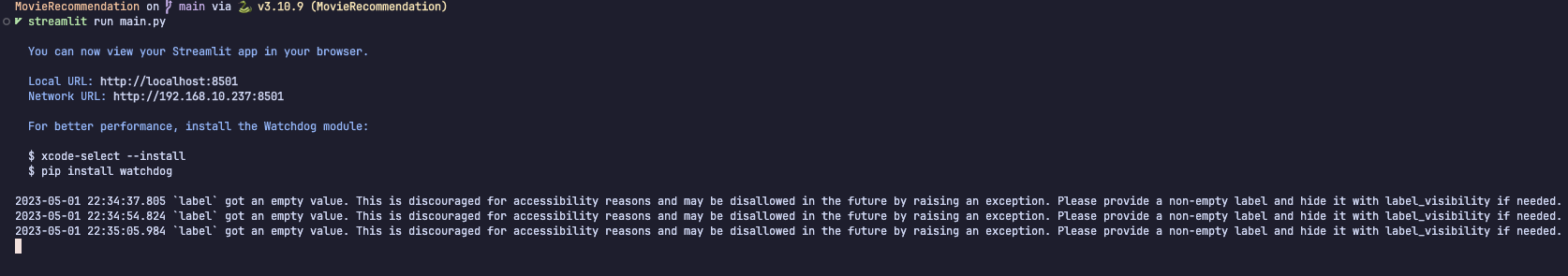


The two vectors(**SimiliartyTfidvect** and **similaritycountvect**) are combing using element multiplication.

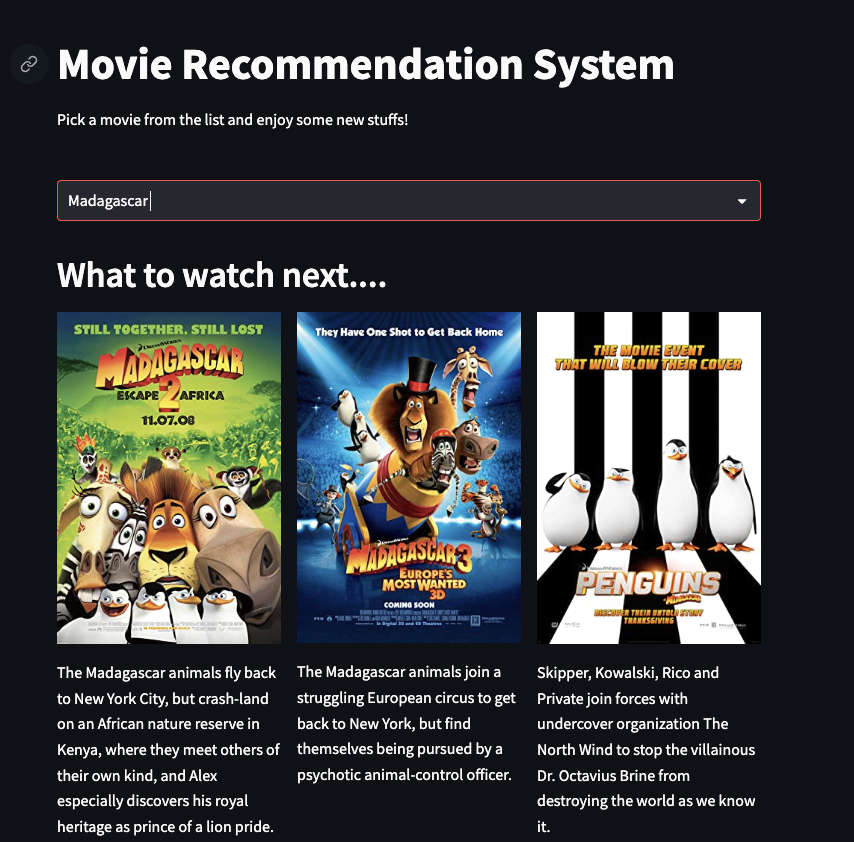


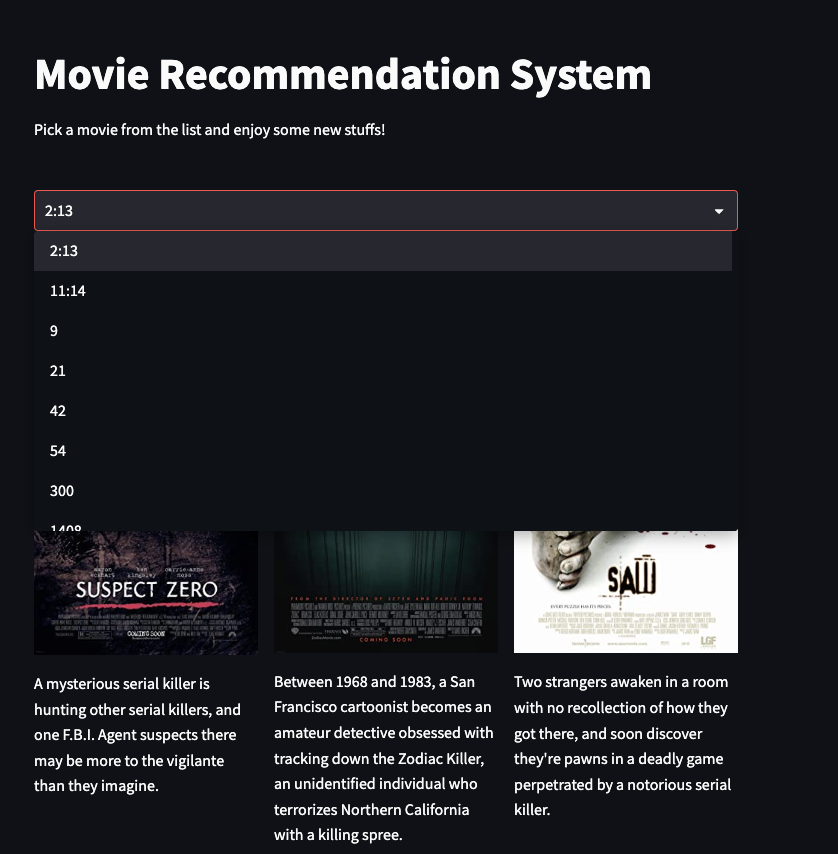


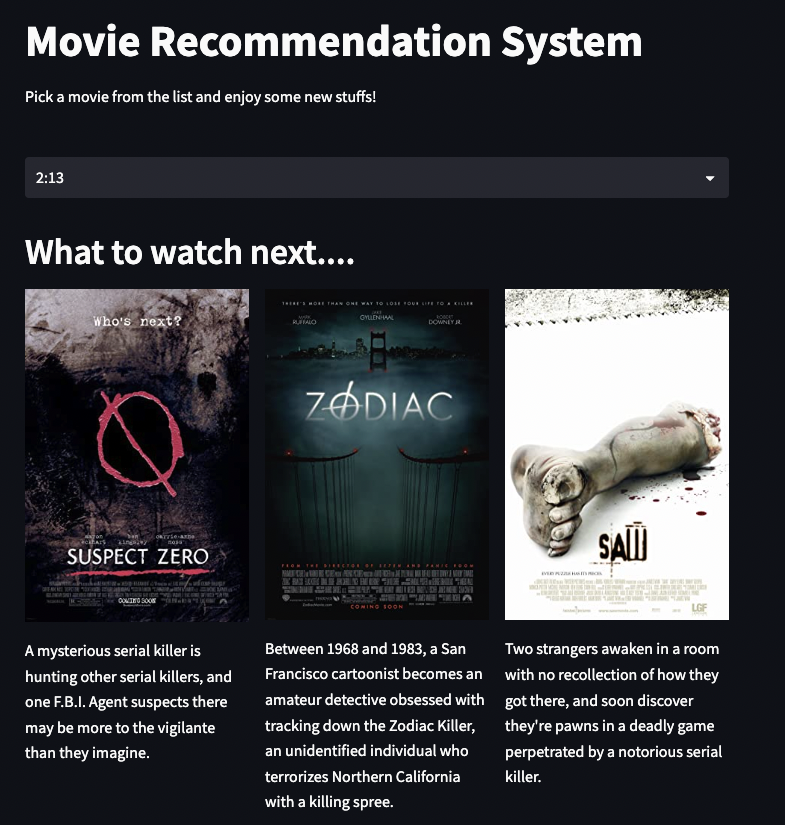
The tags.csv file



Output when running the streamlit service







**Contributions**

•Shiv – TFIDF-Vectorizer and Recommendation

•Love – Count Vectorizer and Data Cleaning

•Stephen – Hash Vectorizer and API Integration

•Dharmik – Similarity Matrix and Web Application

**Meeting Times**

Initial Meeting (March 20-21,2023) (Met through Microsoft Teams):

Discussed the different sections in order to complete the report. This included the goal, the abstract, and assigning the roles. The roles are:

Stephen: Creating the recommendation function that ranks existing items by their similarity to a selected item: when the user picks a movie, this function will propose N movies like A.

Dharmik: Creating similarity matrix using the 1st vectorization method

Love: Creating similarity matrix using the 2nd vectorization method. Creating the web front end to display the recommended movies.

Shiv: Creating similarity matrix using the 3nd vectorization method. Cleaning up the 2 csv files by removing unnecessary data, and combing relevent data into 1 file.

We also discussed when to meet next, and what to do between now and then. Dharmik is going to start working on creating the similarity matrix, I am going to start creating the front end, and try to get the final look before the next meeting. Shiv was given the csv file and he had to start working on cleaning up the data. The csv included the movie data when searching for the movie watched, the csv would also be used to recommend new movies. The next meeting will be on April 1, 2023. This will allow the team to work on their part of the project. No one is expected to finish their part, but try to research and get started on it.

April 6th (through zoom)

Recommendation function updates:

Ranking system assigns values to items according to their similarity to a chosen item

Similarities include genre and content from movie plot descriptions

April 25th (through zoom)

Started working on the powerpoint slides. The code is finished, currently working on clean up and commenting.

May 1st (through zoom)

We finished up the power point slides and the report. We also split up the slides based on the contributions. Then we did a mock presentation.

**Conclusion**

In conclusion, through creating this project, I was able to successfully take the concepts of machine learning together and it taught me how to use different algorithms into Movie Recommendation project.

**Reference**

(TMDb), T. M. D. (2017, September 28). *TMDB 5000 movie dataset*. Kaggle. Retrieved May 2, 2023, from <https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?resource=download&select=tmdb_5000_movies.csv>

*Streamlit docs*. Streamlit documentation. (n.d.). Retrieved May 2, 2023, from <https://docs.streamlit.io/>

Vidiyala, R. (2020, October 21). *How to build a movie recommendation system?* Medium. Retrieved May 2, 2023, from <https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109>