# Movie Recommendation System Documentation

May 5, 2024

## Contents

# 1 Introduction

This program implements a movie recommendation system using the TMDB 5000 Movie Dataset. It utilizes movie descriptions, genres, keywords, cast, and crew information to recommend similar movies to the user.

# 2 Libraries Used

## 2.1 numpy

NumPy is a fundamental package for scientific computing with Python. It provides support for arrays, matrices, and mathematical functions, making it essential for numerical computations in data analysis and machine learning tasks.

## 2.2 pandas

Pandas is a powerful Python library for data manipulation and analysis. It provides data structures like DataFrame and Series, along with functions to efficiently handle structured data. Pandas is widely used for tasks such as data cleaning, exploration, and transformation.

## 2.3 ast

The ast module in Python is used for abstract syntax tree manipulation. In this program, it's utilized for converting stringified lists into actual lists. This is particularly useful for parsing and evaluating Python code represented as strings.

## 2.4 nltk

NLTK, or the Natural Language Toolkit, is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, such as WordNet. NLTK is commonly used for tasks like tokenization, stemming, tagging, parsing, and semantic reasoning.

## 2.5 sklearn

Scikit-learn is a versatile Python library for machine learning. It provides simple and efficient tools for data mining and data analysis, built on top of NumPy, SciPy, and matplotlib. Scikit-learn includes various algorithms for classification, regression, clustering, dimensionality reduction, and model selection.

# 3 Data Loading

The program loads two datasets:

- `tmdb_5000_movies.csv`: Contains information about movies.
- `tmdb_5000_credits.csv`: Contains credits information including cast and crew.

# 4 Data Preprocessing

- **Merging Datasets**: Merges the two datasets on the 'title' column. This step combines movie information with credits information.
- **Handling Missing Values**: Drops rows with missing values in essential columns. This ensures data integrity and avoids errors during analysis.
- **String-to-List Conversion**: Converts stringified lists in columns like 'genres', 'keywords', 'cast', and 'crew' to actual lists using the `ast.literal_eval()` function. This step prepares the data for further processing.

These preprocessing steps are crucial for cleaning and preparing the data for feature engineering and analysis.

# 5 Feature Engineering

- **Extracting Cast and Crew Information** One aspect of feature engineering involves extracting relevant information from the dataset. In this step, we extract the top 3 cast members and the director from the 'cast' and 'crew' columns respectively. This information is crucial as it helps in identifying key contributors to the movie's production and can influence the recommendation process.

## 5.1 Tokenization and Stemming

Tokenization and stemming are crucial steps in the feature engineering process of the movie recommendation system. Here's a detailed explanation of each:

### 5.1.1 Tokenization

Tokenization involves breaking down the text data into individual words or tokens. In the context of the movie recommendation system, we tokenize various textual features such as movie overview, genres, keywords, cast, and crew.

For example, consider the movie overview "A group of teenagers discover a mysterious book that leads them to a hidden treasure." After tokenization, this overview would be split into individual tokens: ["A", "group", "of", "teenagers", "discover", "a", "mysterious", "book", "that", "leads", "them", "to", "a", "hidden", "treasure"].

Tokenization standardizes the text data and enables further analysis such as sentiment analysis, keyword extraction, and similarity computation.

#### 5.1.2 Stemming

Stemming is the process of reducing words to their root form by removing suffixes and prefixes. This normalization technique helps in reducing the dimensionality of the text data and ensures consistency in word representations.

For example, consider the words "running", "runs", and "ran". After stemming, these words would be reduced to their root form "run". Similarly, "discover" and "discovered" would both be stemmed to "discover".

Stemming is particularly useful in natural language processing tasks where word variations need to be treated as the same entity. It simplifies text analysis and improves the efficiency of algorithms such as text classification and clustering.

### 5.2 Combining Features

Combining features involves aggregating different individual features into a single feature vector. In the movie recommendation system, we combine textual features such as movie overview, genres, keywords, cast, and crew into a unified feature vector called 'tags'.

Each component of the 'tags' feature vector represents a specific aspect of the movie. For example, the tokenized and stemmed movie overview captures the essence of the plot, while the genres feature indicates the genre(s) of the movie. Similarly, the keywords, cast, and crew components provide additional contextual information about the movie.

By combining these features into a single feature vector, we create a comprehensive representation of each movie. This allows us to compute similarity scores between movies more effectively and generate accurate recommendations based on user preferences.

## 6 Vectorization

Vectorization is a critical step in the movie recommendation system, facilitating the conversion of textual data into numerical representations for analysis and computation. Here's a detailed explanation of each aspect:

### 6.1 Text Vectorization

Text Vectorization is the process of converting textual data into numerical vectors. In our movie recommendation system, we utilize the CountVectorizer, a feature extraction technique available in the scikit-learn library.

CountVectorizer converts a collection of text documents into a matrix of token counts, where each row represents a document and each column represents a unique token (word). The value in each cell of the matrix represents the frequency of the corresponding token in the document.

For example, consider a corpus containing three movie overviews: 1. "A group of teenagers discover a mysterious book." 2. "A detective investigates a series of murders in a small town." 3. "An astronaut embarks on a mission to explore a distant planet."

After applying CountVectorizer, the textual data is transformed into a numerical matrix where each row represents a movie and each column represents a unique word from the corpus. The cell values indicate the frequency of each word in the respective movie overview.

Text vectorization enables us to perform mathematical operations and analysis on textual data, such as computing similarity between movies and building machine learning models for recommendation.

### 6.2 Cosine Similarity

Cosine Similarity is a metric used to measure the similarity between two vectors in a multidimensional space. In the context of the movie recommendation system, cosine similarity is applied to the numerical vectors representing the movies to determine their similarity.

Cosine similarity calculates the cosine of the angle between two vectors, indicating how closely aligned they are in the vector space. A cosine similarity score of 1 indicates that the vectors are perfectly aligned (i.e., identical), while a score of 0 indicates that the vectors are orthogonal (i.e., completely dissimilar).

By computing cosine similarity between the numerical vectors representing movie features (e.g., movie overviews, genres, keywords), we can quantify the similarity between movies. Movies with higher cosine similarity scores are considered more similar, and thus, suitable candidates for recommendation to users with similar preferences.

Cosine similarity provides a robust measure of similarity that is independent of the magnitude of the vectors, making it well-suited for comparing textual data and generating accurate movie recommendations based on content similarity.

## 7 Recommendation Functions

Recommendation functions play a crucial role in the movie recommendation system, enabling users to discover movies based on various criteria. Here's a detailed explanation of each recommendation function:

## 7.1 Recommend Based on Similarity

The Recommend Based on Similarity function leverages cosine similarity scores computed between movies to recommend similar movies to the user. It takes a target movie as input and identifies other movies with high cosine similarity scores, indicating similarity in content.

For example, if a user expresses interest in the movie "Inception", the system would recommend other movies with similar themes, plots, or genres, such as "The Matrix" or "Interstellar". This function helps users discover new movies that align with their preferences and interests.

## 7.2 Recommend Based on Cast

The Recommend Based on Cast function recommends movies with the same cast members as a target movie. It identifies movies featuring actors or actresses who have appeared in the cast of the target movie and suggests them to the user.

For instance, if a user enjoys movies starring Leonardo DiCaprio and is interested in "Titanic", the system would recommend other movies featuring Leonardo DiCaprio, such as "The Wolf of Wall Street" or "Catch Me If You Can". This function allows users to explore movies based on their favorite actors or actresses.

## 7.3 Recommend Based on Director

The Recommend Based on Director function suggests movies directed by the same director as a target movie. It identifies movies helmed by the director of the target movie and presents them as recommendations to the user.

For example, if a user admires the work of Christopher Nolan and enjoys "The Dark Knight", the system would recommend other movies directed by Christopher Nolan, such as "Inception" or "Interstellar". This function enables users to explore movies based on their appreciation for specific directors' styles and visions.

### 7.3.1 Recommend Based on Genres

The Recommend Based on Genres function recommends movies based on similar genres to a target movie. It analyzes the genres associated with the target movie and identifies other movies sharing the same or similar genre categories.

For instance, if a user enjoys action-adventure movies and is interested in "Indiana Jones and the Raiders of the Lost Ark", the system would recommend other action-adventure movies, such as "The Mummy" or "National Treasure". This function allows users to discover movies within their preferred genre categories.

Each recommendation function offers users a unique way to explore and discover movies tailored to their preferences, whether based on content similarity, cast, director, or genres.

# 8 Streamlit Movie Recommendation System

The Streamlit movie recommendation system is a web-based application developed using the Streamlit framework, allowing users to discover movie recommendations based on various criteria. Below is a detailed explanation of the key components and functionalities of the Streamlit application:

## 8.1 User Interface Setup

The user interface (UI) of the application is configured using Streamlit commands. The `st.set_page_config()` function is used to configure the page title, icon, and layout settings. Additionally, the application header is set using the `st.header()` command, displaying the title "MOVIE RECOMMENDATION SYSTEM" with a rainbow-colored divider.

## 8.2 Footer Customization

A custom footer is added to the application UI using HTML and CSS styling. The footer displays the message "Developed by Team No-Itihad" in white text on a background color of d95a00 (orange), providing credit to the development team.

## 8.3 Functions for Recommendation

Several functions are defined within the application to facilitate movie recommendations based on different criteria:

- `fetch_poster(movie_id)`: Retrieves the URL of the movie poster for a given movie ID using the TMDB API.
- `recommend(movie)`: Recommends similar movies based on the selected movie using cosine similarity scores. Retrieves movie titles and posters for the recommended movies.
- `recommend_Dir(movie)`: Recommends movies directed by the same director as the selected movie. Retrieves movie titles and posters for the recommended movies.
- `recommend_with_cast(movie)`: Recommends movies featuring the same cast as the selected movie. Retrieves movie titles and posters for the recommended movies.
- `recommend_genres(movie)`: Recommends movies based on similar genres to the selected movie. Retrieves movie titles and posters for the recommended movies.

### 8.4 User Interaction

The Streamlit application allows users to interact with the recommendation system by selecting a movie from a dropdown menu and clicking the "Show Recommendation" button. Upon clicking the button, the application displays recommended movies based on similarity, genres, director, and cast, along with their posters and titles in a grid layout.

### 8.5 Integration with Pickle Files

The application loads precomputed dataframes containing movie information and similarity matrices from pickle files. These files are generated during the preprocessing and modeling stages, enabling quick access to the required data for recommendation tasks.

## 9 File Saving

Saves the processed dataframes and similarity matrix to pickle files for future use.

### 9.1 File Saving

The File Saving process is an essential component of the movie recommendation system, enabling the preservation of processed dataframes and the similarity matrix for future use. Here's a detailed explanation of this process:

Once the data preprocessing, feature engineering, and vectorization steps are completed, the processed dataframes containing relevant movie information and the similarity matrix computed using cosine similarity are saved to pickle files. Pickle is a Python serialization format that allows objects to be serialized and deserialized efficiently, preserving their structure and state.

Saving the processed dataframes and similarity matrix to pickle files ensures that the preprocessed data and computed similarity scores are preserved in a compact and efficient format. This enables easy retrieval and reuse of the processed data and similarity information for future recommendation tasks, without the need to recompute them from the raw data.

By saving the processed dataframes, the system maintains a record of essential movie information such as titles, tags, and feature vectors. Similarly, the similarity matrix captures the pairwise similarity scores between movies, providing valuable insights into their content relationships.

The use of pickle files for storage offers several advantages, including fast read and write operations, platform independence, and support for various Python data types. Additionally, pickle files are binary files, ensuring data integrity and security.

Overall, saving the processed dataframes and similarity matrix to pickle files enhances the efficiency and scalability of the movie recommendation system, fa-

cilitating seamless integration with other applications and enabling quick access to valuable data for recommendation tasks.

# 10 Conclusion

The movie recommendation system implemented in this program offers a comprehensive and personalized approach to movie discovery, leveraging a diverse range of movie features such as overview, genres, cast, and crew. Through advanced data preprocessing, feature engineering, and recommendation algorithms, users are provided with tailored movie recommendations that align with their preferences and interests. Here's a detailed overview of the system's capabilities:

## 10.1 Comprehensive Recommendation System

The movie recommendation system analyzes multiple facets of movies, including their plot summaries, genre classifications, and the individuals involved in their production. By considering a wide array of features, the system ensures that recommendations are holistic and reflective of the diverse aspects that contribute to a movie's appeal.

## 10.2 Personalized Recommendations

Users can receive personalized recommendations based on their individual preferences and viewing history. Whether users are seeking movies similar to their favorite films, featuring their preferred actors or directors, or within specific genre categories, the recommendation system tailors its suggestions to meet their unique requirements.

## 10.3 Multi-Faceted Recommendation Criteria

The system offers recommendations based on various criteria, including content similarity, cast members, directors, and genres. This multi-faceted approach enables users to explore movies from different perspectives and discover hidden gems that match their tastes and preferences.

## 10.4 Efficient and Scalable Architecture

Through efficient data preprocessing, feature extraction, and vectorization techniques, the recommendation system ensures optimal performance and scalability. By saving processed dataframes and similarity matrices to pickle files, the system facilitates quick access to valuable information for recommendation tasks, enhancing overall efficiency.

### 10.5 User-Centric Experience

At its core, the movie recommendation system is designed to enhance the user experience by providing relevant, high-quality movie recommendations. By leveraging advanced machine learning algorithms and data processing techniques, the system aims to delight users with personalized and engaging movie suggestions, fostering a deeper appreciation for cinema and enriching their viewing experiences.

### 10.6 Streamlit

The movie recommendation system, coupled with the Streamlit framework, represents a sophisticated blend of data science, machine learning, and user-centric design principles. By harnessing the power of data analytics and artificial intelligence, the system empowers users to discover and enjoy movies that resonate with their tastes and preferences, ultimately enhancing their overall entertainment experience.

Streamlit, with its intuitive interface and powerful backend functionality, plays a crucial role in enhancing the user experience. Its user-friendly design allows seamless interaction with the recommendation system, enabling users to explore movies based on various criteria and make informed viewing decisions effortlessly.

In conclusion, the integration of Streamlit into the movie recommendation system offers a comprehensive solution for movie discovery and recommendation. By leveraging Streamlit's capabilities alongside advanced machine learning algorithms, the system caters to the diverse preferences of users and enriches their movie-watching experiences.

In conclusion, the movie recommendation system represents a sophisticated blend of data science, machine learning, and user-centric design principles. By harnessing the power of data analytics and artificial intelligence, the system empowers users to discover and enjoy movies that resonate with their tastes and preferences, ultimately enhancing their overall entertainment experience.

## 11 Team Segregation

The project team is divided into three teams and managed by one Project Manager:

### 11.1 AI Model Training (4 members)

- Fayaz Noor (22P-9012)
- Kashif Khan (22P-9005)
- Aashir Shehzad (22P-9268)
- Saad Khan (22P-9007)

### 11.2 Web Page Development (3 members)

- Muhammad Hussain (22P-9270)
- Sami ur Rehman (22P-9006)
- Alamzeb (22P-9266)

### 11.3 Documentation (2 members)

- Muhammad Afnan (22P-9256)
- Muhammad Huzefa (22P-9267)

### 11.4 Project Manager (1 member)

- Ibrahim Aslam (22P-9275)

## 12 References


1. Movie Recommender System Project. [Online Video]. Available: https://www.youtube.com/watch?v=1x
   5639$s$
2. Dataset: TMDB Movie Metadata. Available: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
3. Movie Recommender System Video. [Online Video]. Available: https://www.youtube.com/watch?v=HOMXxr