WiDS ‘22 - ‘23 Final Documentation

25 - Movie Recommendation System
Mentor- Yadav Amar Singh

Team Member Name	Roll Number	Email-Id
KIRAN PRAKASH EC	22M1514	22m1514@iitb.ac.in

Introduction to Problem Statement

Aimed to develop an advanced movie recommendation system using a combination of natural language processing (NLP) techniques, machine learning algorithms, and web scraping. The primary goal was to create a content-based recommendation system that provides personalized movie recommendations to users based on their preferences and sentiment analysis of movie reviews.

Existing Resources

Retrieve movie metadata from Wikipedia and The Movie Database (TMDB) websites using the tmdbv3api library.

Proposed Solution

Content-based Recommendation system:

This technique makes recommendations based on the characteristics of the movies themselves, such as their overview, genre, director, tagline etc.

Methodology & Progress (Mention the work done week-wise)

Week 1: Utilized the tmdbv3api library to collect movie metadata from Wikipedia and TMDB. Applied NLP techniques including lemmatization and regex to preprocess the movie data. The focus was on learning the basics of movie recommendation systems, gaining an overview of the project and its objectives, and becoming proficient in the use of Python and its libraries. I also spend time pre-viewing the data that will be used in the project, such as movie metadata and user ratings, in order to better understand the characteristics of the data and how it can be used to make recommendations. This week served as a foundation for the rest of the project.Focused on pre-processing, feature engineering and utilizing relevant Python libraries. I also delved into the basics of machine learning models and natural language processing (NLP) techniques, which will be used to build the movie recommendation system. Pre-processing was carried out to clean, organize and transform the data, while feature engineering was used to create new features from the existing data that will be useful for making recommendations. I also got familiarized with basic machine learning models and NLP techniques, which will be used to build the movie recommendation system.

Week 2: Focused on understanding the different types of recommendation systems that are available and how they can be applied to the movie recommendation system. I specifically studied content-based filtering, which is a technique that makes recommendations based on the characteristics of the items. I also looked into various techniques such as collaborative filtering, hybrid systems and knowledge based filtering, which can be used to make personalized recommendations for movies to individual users. I started coding in my 4th week. I have acquired four different datasets from various movie databases. In the **Text preparation** step, I have done merging of these different datasets, which enabled the use of multiple data sources to make more accurate recommendations. In the **pre-processing** step, I have performed several activities to ensure that the data is clean, consistent, and ready for use in the movie recommendation system. The team first dropped any unwanted columns that were not relevant to the project. Next, any duplicate rows were removed to avoid bias and inconsistencies in the data. The index of the dataset was then reset to ensure that it is properly aligned. To improve consistency and readability, any punctuations ,stopwords were removed, and all the text values were converted to lowercase. Any missing values in the data were then filled with the appropriate placeholder such as 'NA' to ensure that the data is complete. In the **Feature extraction** step, I have combined the columns of genres, overview, tagline and original language to make a new column named description. Later I have dropped the mentioned 4 columns. Then, I have used the **Tfldf vectorizer** to convert the text data into numerical feature vectors. This technique is widely used in natural language processing and information retrieval as it can effectively represent the importance of a particular word or phrase within a document, as well as across a set of documents. The tf-idf vectorizer assigns a weight to each word based on its frequency within a document and its rarity across all documents, resulting in a feature vector that captures the relevance of each word to the overall text. Then I implemented the **content based recommender system**. I have used the **linear_kernel** function to compute the **cosine similarity** between all pairs of documents represented in the tf-idf matrix. Cosine similarity is a measure of similarity between two

non-zero vectors of an inner product space that measures the cosine of the angle between them. In the context of text data, it compares the similarity of two documents or sets of documents by measuring the angle between their tf-idf vectors. The linear_kernel function computes the cosine similarity scores between all pairs of documents. This cosine similarity score helps to find similar documents based on their content , based on which movies are recommended. We select the top ten cosine similarity scores and recommend the respective movies.

Week 3: Sentiment Analysis:
Performed web scraping on IMDb to collect movie reviews.
Implemented sentiment analysis using both Naive Bayes and LSTM models.
Achieved sentiment analysis accuracies of 98.77% (Naive Bayes) and 98.96% (LSTM).

Web Application Development:

Utilized Django to build a user-friendly web interface for the recommendation system.
Enabled users to input preferences and receive personalized movie suggestions.

Week 4: Dockerization of the Content-Based Recommendation System

Dockerization is the process of packaging an application along with its dependencies, libraries, and runtime environment into a standardized container. This container can be easily transported across different environments and platforms, ensuring consistent behavior regardless of the underlying infrastructure. In the context of the Movie Recommendation System (MRS) project, Dockerization was employed to streamline the deployment and enhance the scalability of the content-based recommendation system.
Docker Installation: Install Docker on the host machine where the application will be deployed. Docker provides a consistent runtime environment for containers.

Dockerfile Creation: Write a Dockerfile, which is a text file containing instructions to build the Docker image. This file specifies the base image, dependencies, environment setup, and commands to run the application.

Application Containerization: Use the Dockerfile to build a Docker image. This image contains the application code, dependencies, and runtime environment, effectively creating a self-contained package.

Image Registry (Optional): Push the Docker image to a container registry like Docker Hub or a private registry. This step facilitates sharing the image with team members and deploying it on multiple machines.

Container Deployment: Deploy the Docker container on target machines or cloud platforms. This can be done using Docker commands or container orchestration tools like Kubernetes.

Environment Configuration: Use environment variables or configuration files to customize the application's behavior based on the deployment environment.

Container Management: Monitor and manage the running containers using Docker commands or orchestration tools. Scaling, updating, and managing containers become easier with Docker's toolset.

Results

Link to my Github repository:
MRS: https://github.com/KiranEC11/movie-recommendation_Analytics-club-Project/tree/main
Docker: https://github.com/KiranEC11/web-interface-mrs

Learning Value

During my project, I gained a comprehensive understanding of the fundamentals of natural language processing (NLP) and applied these concepts to model a recommender system. I learned about the NLP pipeline, including data acquisition, text preprocessing, feature extraction, modeling and Deployment. Also I got familiarized with various recommender systems such as content based filtering, collaborative based filtering etc.

Tech-stack Used

Programming language: Python
Python libraries:

- Numpy
- Pandas
- Matplot library
- NLTK - Natural Language Tool Kit
- Sk learn - scikit learn
- Tfldf vectorizer
- Linear_kernel function
- Cosine_similarity
- Django
- Docker
- LSTM
- Naive Bayes

Coding and execution in Google Colaboratory

Suggestions for others

For those who want to model and design a 'movie recommendation system', they should have a comprehensive understanding about python libraries, NLP pipeline (Data acquisition, Text preprocessing, Feature extraction, modeling and Deployment) and different types of recommendation systems.

Contribution by each Team Member

Mine is a one member team.

References and Citations

- "Content-Based Recommendation Systems" by J.O. Perlich and D.P. Hill.
- "Content-Based Book Recommending Using Learning for Text Categorization" by C.J. van Rijsbergen.
- "A Hybrid Recommender System Based on Content-Based Filtering and Collaborative Filtering" by X. Su and H. Liu.
- "Content-Based Filtering Recommendation Algorithms: A Comprehensive Study and Performance Analysis" by S. Kim and H. Park.
- "Combining Collaborative Filtering with Content-Based Filtering: A Hybrid Recommender System" by X. Ning and G. Karypis.

- "Content-Based Recommendation Systems: A Literature Review and Classification" by M. Adomavicius and A. Tuzhilin.