MOVIE RECOMMENDATION USING SPEECH RECOGNITION

By

1.	Ricky Setiawan	- 001202200068
2.	Marchelleo Suhandi	- 001202200027
3.	Reinhard Cavin Saroinsong	- 001202200082
4.	Ryan Ruland	- 001202200065
5.	Yustina Yunita	- 001202200024

A Report Submitted
in Partial Fulfillment of the Requirements
for the Artificial Intelligence Class

Cikarang, Bekasi, Indonesia
April, 2024

TABLE OF CONTENTS

THEME

Speech Recognition for Movie Recommendation System using speech recognition to give accessibility for those who prefer to use verbal interaction over typing. This means people can easily tell the system what kind of movies they like without typing. Then, the movie recommendation system generates personalized movie recommendations that match their preferences. It makes finding new movies fun and easy for everyone, especially those who prefer talking over typing.

PROBLEM & OBJECTIVES

1. Problem

   Many people struggle with choosing a movie that suits their preferences because there's a lot of movies available, and sometimes it's challenging to find the movies that we are interested in or that are suitable to our preferences. This problem may cause a waste of time scrolling through all the list of movies, and you can end up spending more time searching for a movie than actually watching one.

2. Objectives

   Our objective is to create a movie recommendation system by using speech recognition to make the movie selection process easier and more personalized for users. Specifically, we aim to achieve the following:

   1. Personalized Recommendations : Using the content-based algorithm, our project aims to provide personalized movie suggestions based on user input by using genre, cast, directors, and some keywords from the dataset to recommend relevant movies.

   2. Speech Recognition Integration : By using speech recognition, we aim to make users able to use talk instead of typing to reduce time wasted and make it easier for users.

   3. Enhance User Experience : Focus on enhancing the overall user experience by reducing the time spent searching for movies and increasing the accuracy of recommendations.

METHOD

Speech and content-based filtering approaches are all incorporated into the creation of the movie recommendation system. The methods used to accomplish the goals of integrating voice recognition, making tailored suggestions, and improving user experience is described in this section.

Libraries:

1. Importing Libraries: We import some necessary libraries for our program, such as
    - Pandas: it's used to handle and manipulate tabular data structures (DataFrames).

    - Numpy: is used for numerical operations in our code as handling vote counts and averages, and for converting data types.

    - Scikit-learn (TfidfVectorizer, CountVectorizer, and linear_kernel):
        - TfidfVectorizer: is used to convert a collection of raw documents into a matrix of TF-IDF features.
        - CountVectorizer: is used to convert a collection of text documents into a matrix of token counts.
        - linear_kernel: computes the linear kernel between pairs of vectors.
        - cosine_similarity: cosine_similarity calculates the cosine similarity between pairs of vectors.
        - In our movie recommendation system, these modules are used for text processing tasks like feature extraction and similarity calculation.

    - NLTK: is used for stemming words and accessing WordNet for lemmatization.

    - spaCy: is used for NLP functionalities like tokenization, part-of-speech tagging, and named entity recognition.

    - Speech_recognition: is used for capturing audio input from the user microphone and converting it to text.

Method & Algorithm:

1. Data Collection and Preprocessing:
    - Movie metadata, including titles, genres, cast members, directors, release dates, and keywords, was obtained from a publicly available dataset.
    - Data preprocessing involved handling missing values, converting data types, and extracting relevant features such as genres, cast, directors, and keywords from the dataset.

- The dataset was cleaned and transformed into a structured format suitable for recommendation system implementation.
- The NLTK and spaCy libraries are used for the text processing tasks (tokenization, lemmatization, and stemming).

2. Content-Based Filtering:
    - Content-based filtering recommends items similar to those a user has liked in the past.
    - Content-based filtering was employed to recommend movies based on the attributes and features of the movies themselves.
    - The algorithm for our Content-Based Filtering is following these steps:
        - Extract relevant features from the movie metadata, such as genres, keywords, cast, director, and release year.
        - Calculate a similarity score between movies based on these features using cosine similarity.
        - When a user requests recommendations, the system identifies movies similar to the ones specified by the user and recommends them.

3. Cosine Similarity:
    - Each movie is represented as a vector in a high-dimensional space, where each dimension corresponds to a feature (e.g., genres, keywords, cast, director).
    - The cosine similarity between two movie vectors measures how similar they are in terms of their features.
    - A cosine similarity of 1 indicates that the two movies are perfectly similar, while a cosine similarity of 0 indicates no similarity, and a cosine similarity of -1 indicates perfect dissimilarity (opposite directions).

4. Integration of Speech Recognition:
    - Speech recognition technology was integrated to enable users to interact with the recommendation system using spoken language.
    - The SpeechRecognition library in Python was utilized to convert spoken words into text.
    - A function was implemented to capture user input via speech and process it for recommendation generation.
    - Users were prompted to speak their movie preferences, and the system converted the speech to text for further processing.

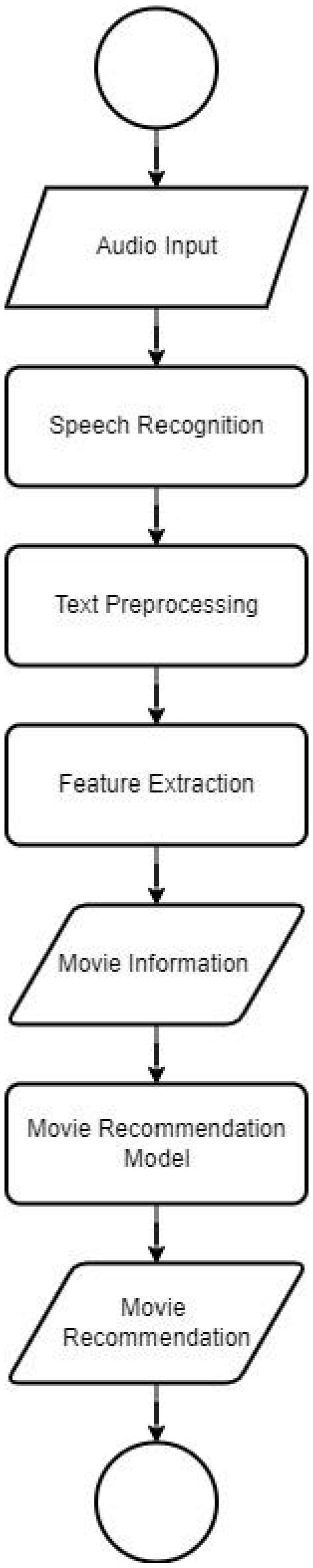
5. Recommendation Generation:
    - Upon receiving user input, the system generated movie recommendations based on collaborative filtering and content-based filtering algorithms.
    - For collaborative filtering, similar movies were recommended based on user preferences and behavior.
    - For content-based filtering, movies with similar attributes and features were recommended based on the extracted metadata.
    - The recommendations were personalized to match the user's preferences and aligned with the objectives of the project.

6. Enhancing User Experience:
    - The primary focus throughout the development process was on enhancing the overall user experience.
    - By integrating speech recognition, users were provided with a convenient and intuitive interface for interacting with the system.
    - The recommendation algorithms were designed to reduce the time spent searching for movies and increase the accuracy of recommendations, thereby improving user satisfaction.

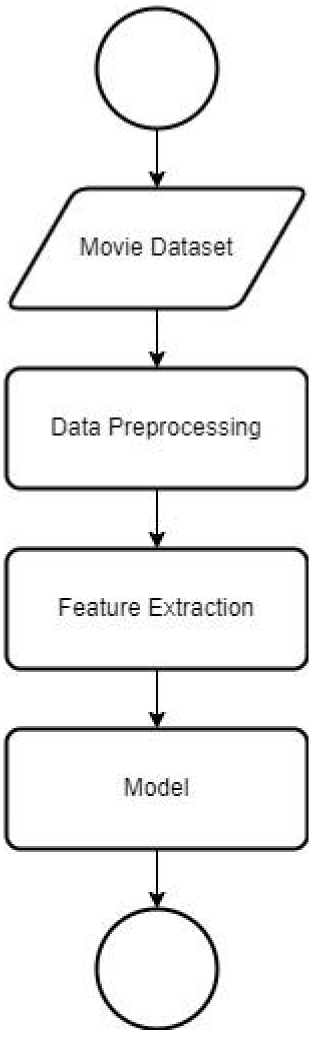
7. Testing and Evaluation:
    - The recommendation system underwent rigorous testing to ensure functionality and accuracy.
    - User feedback and testing results were used to fine-tune the system and address any issues or limitations.
    - Performance metrics, such as recommendation accuracy and user satisfaction, were evaluated to assess the effectiveness of the system.

FLOWCHART

Speech Flowchart



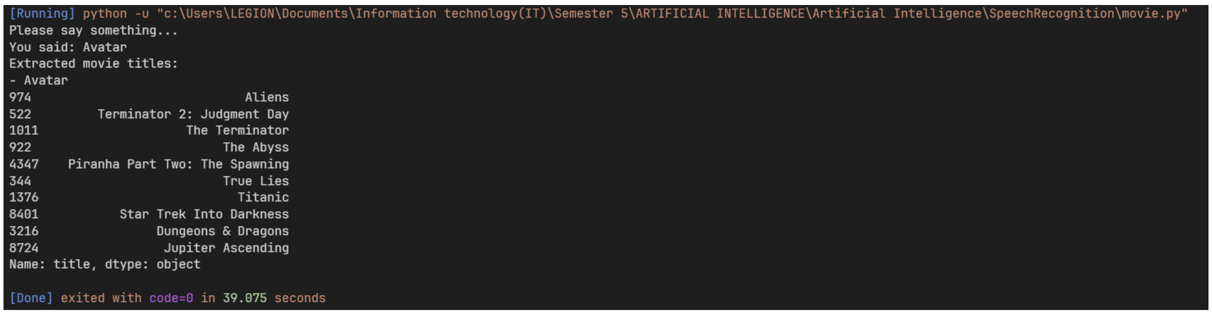
Model Flowchart



EXPECTED RESULTS

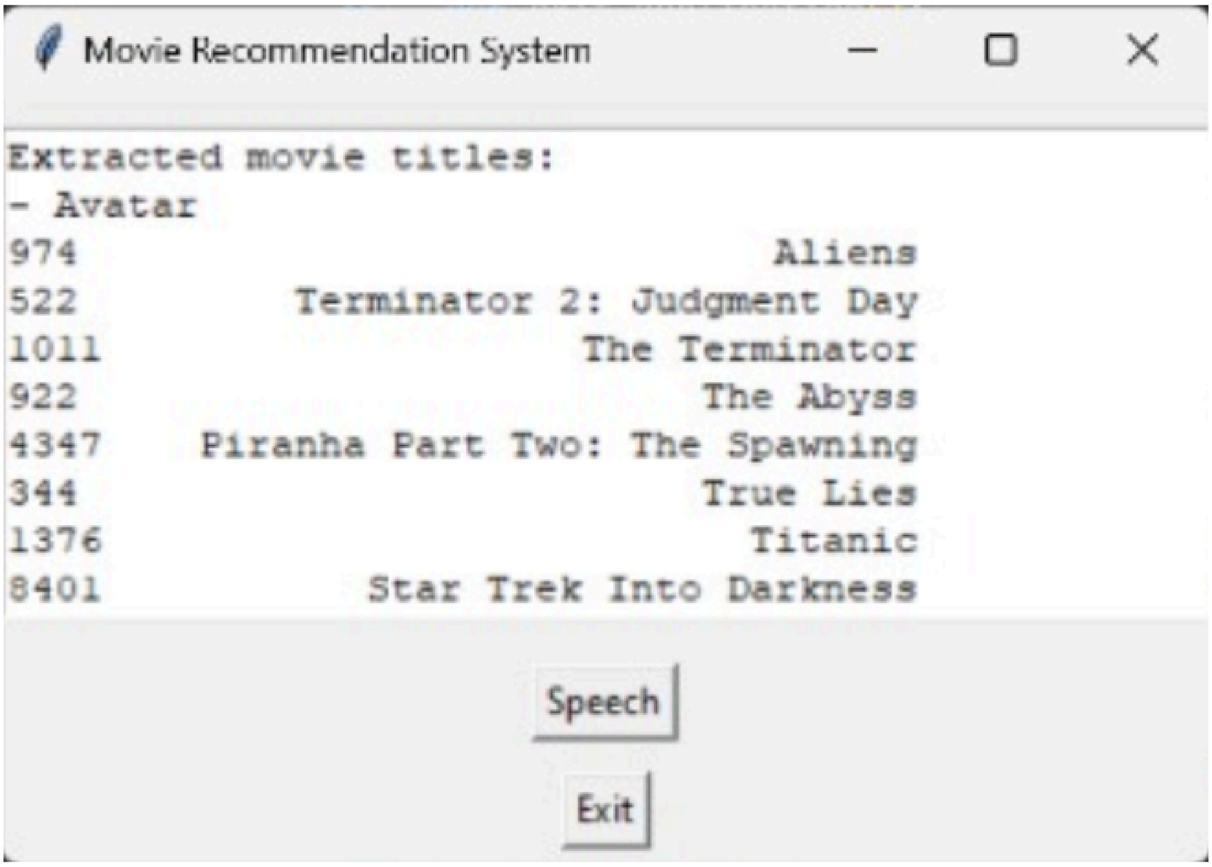
- By using speech input, the user is expected to provide a list of movies that they would like to see recommended by the script.
- Based on similarity in movie elements like cast, crew, keywords, and genres, recommendations are made.
- The recommendations ought to take into account the user's indicated preferences as well as the similarity scores determined by the content-based filtering algorithm.

In conclusion The script is designed to deliver movie recommendations based on user speech input. By leveraging content-based filtering techniques, it analyzes movie features such as cast, crew, keywords, and genres to identify similarities between movies. The expected outcome is a personalized list of movie recommendations tailored to the user's preferences, inferred from the provided speech input. However, the effectiveness of the recommendations may depend on several factors, including the quality and quantity of available movie data, the accuracy of text processing and feature extraction techniques, and the robustness of the content-based filtering algorithm



```
[Running] python -u "c:\Users\LEGION\Documents\Information technology(IT)\Semester 5\ARTIFICIAL INTELLIGENCE\Artificial Intelligence\SpeechRecognition\movie.py"
Please say something...
You said: Avatar
Extracted movie titles:
- Avatar
974                               Aliens
522         Terminator 2: Judgment Day
1011                    The Terminator
922                          The Abyss
4347    Piranha Part Two: The Spawning
344                          True Lies
1376                           Titanic
8401           Star Trek Into Darkness
3216                Dungeons & Dragons
8724                 Jupiter Ascending
Name: title, dtype: object

[Done] exited with code=0 in 39.075 seconds
```



Movie Recommendation System

```
Extracted movie titles:
- Avatar
974                               Aliens
522         Terminator 2: Judgment Day
1011                    The Terminator
922                          The Abyss
4347    Piranha Part Two: The Spawning
344                          True Lies
1376                           Titanic
8401           Star Trek Into Darkness
```

REFERENCE

[1] R. Sikiru, "Movie Recommender System," Kaggle, [Online]. Available: https://www.kaggle.com/code/rashidatsikiru/movie-recommender-system.

[2] R. Vidiyala, "Movie Recommendation System," Towards Data Science, 2021. [Online]. Available: https://medium.com/web-mining-is688-spring-2021/movie-recommendation-system-bb46ba0f6f86.

[3] Tyasemin, "Speech Recognition and Recommender Systems," GitHub, [Online]. Available: https://github.com/tyasemin/Speech-Recognition-and-Recommender-Systems.

[4] Jiyeon1997, "Movie Recommender Python," GitHub, [Online]. Available: https://github.com/jiyeon1997/movie-recommender-python/blob/master/README.md.

[5] "Movie Recommendation Engine with NLP," Analytics Vidhya, 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/01/movie-recommendation-engine-with-nlp/.

LOG HOURS

Please mention the work done by the team members and the hour spent.

No	Student Name	Task Done and Log Hour Spent
1.	Ricky Setiawan	Libraries, Method & Algorithm used (2 Hours)
2.	Marchelleo Suhandi	Method (1 Hours)
3.	Reinhard Cavin Saroinsong	Code (5 Hours) and Theme Section (30 Minutes)
4.	Ryan Ruland	Flowchart, Code, Expected Result, Reference (3 Hours)
5.	Yustina Yunita	Problem & Objectives (1 Hours)