National University of Computer and Emerging Sciences
Islamabad Campus

Project

Movie Summary Analyzer

AI2002

Artificial Intelligence

Submitted by: Haziq Hussain, Hussain Abdullah
Roll number: 22i-1685, 22i-1639
Date: 6 May 2025

1. Introduction

This report provides a detailed analysis of the MovieGUI application, a Python-based graphical user interface (GUI) designed to process movie summaries. The application leverages natural language processing (NLP) and machine learning to perform tasks such as genre prediction and multilingual audio generation. Built using tkinter, the application integrates libraries like pandas, scikit-learn, transformers, and gTTS to offer a robust user experience. The report is structured to explain the code's functionality, architecture, and implementation details, catering to both technical and non-technical audiences.

2. System Overview

The MovieGUI class creates a desktop application that allows users to:

- Input a movie summary via a text box.
- Predict the movie's genres based on similarity to a preprocessed dataset.
- Convert the summary into audio in Arabic, Urdu, or Japanese.
- Clear the input or exit the application.

The application relies on a dataset (cleaned_movie_data.csv) containing preprocessed movie summaries and genres. It uses TF-IDF vectorization for genre prediction and MarianMT models for translation.

3. Code Structure and Dependencies

a. Dependencies

The application imports the following Python libraries:

- tkinter and appointments for GUI creation.
- gTTS and playsound for text-to-speech and audio playback.
- pandas and numpy for data handling.
- scikit-learn (TfidfVectorizer, cosine_similarity) for NLP tasks.
- transformers (MarianMTModel, MarianTokenizer) for translation.

• os and time for file management and timestamp generation.

b. Main Class: MovieGUI

The MovieGUI class is initialized with a tkinter root window and encapsulates all functionality. Its constructor (__init__) sets up the GUI, loads translation models, and prepares the dataset.

4. Detailed Code Explanation

a. Initialization (__init__)

The constructor performs the following tasks:

1. Configures the main window with a title ("Movie Summary Analyzer") and size (600 x 500 pixels).

2. Defines a dictionary of supported languages (Arabic, Urdu, Japanese) with their respective translation model names and gTTS language codes.

3. Loads MarianMT translation models and tokenizers for each language, displaying an error and exiting if loading fails.

4. Loads the movie dataset from cleaned_movie_data.csv, handling missing summaries and parsing genres into lists.

5. Initializes a TfidfVectorizer with 500 features and English stop words, then transforms the dataset summaries into TF-IDF vectors.

6. Creates GUI elements, including:

- A text box for summary input.
- Buttons for actions (audio conversion, genre prediction, exit).
- A dropdown menu for language selection.
- Execute and Clear buttons.
- A read-only text area for results.

b. Key Methods

1. preprocess_summary

• Converts text to lowercase and removes non-alphabetic characters, matching the preprocessing in preprocess_split.py.

• Returns a cleaned string for consistent processing.

2. translate_text

• Translates input text to the target language using the appropriate MarianMT model and tokenizer.

• Handles errors by displaying a message and returning an empty string.

3. create_audio

• Translates the input summary to the selected language.

• Creates a language-specific directory (e.g., audio_ar) and generates a unique audio file name using a timestamp.

• Uses gTTS to convert translated text to audio and saves it as an MP3 file.

• Returns the file path or None on failure.

4. play_audio

• Plays the audio file using playsound and deletes it afterward to save space.

• Displays an error message if playback fails.

5. predict_genres

• Preprocesses the input summary and transforms it into a TF-IDF vector.

• Computes cosine similarity between the input vector and dataset vectors.

• Identifies the most similar summary and retrieves its genres.

• Returns the genres or a default message if none are found.

6. update_output

- Updates the read-only output text area with the provided message.

7. clear_summary

- Clears the input text box and updates the output area to confirm.

8. execute_action

- Handles button actions: exit, audio conversion, or genre prediction.
- Validates the input summary, displaying a warning if empty.
- For audio conversion, it creates and plays the audio file in the selected language.
- For genre prediction, it displays the predicted genres.

5. Implementation Details

a. Dataset Handling

The dataset is expected to have two columns: cleaned_summary and genres. Missing summaries are replaced with empty strings, and genres are split into lists for processing. The TfidfVectorizer is configured to ignore common English words and limit features to 500 for efficiency.

b. Translation Pipeline

The MarianMT models from Hugging Face are used for translation. Each language has a dedicated model (e.g., opus-mt-en-ar for Arabic). The translate_text method ensures truncation at 512 tokens to prevent memory issues.

c. Audio Generation

The gTTS library generates audio in the target language, with output saved in language specific directories. Files are named using timestamps to avoid conflicts and deleted after playback.

d. Genre Prediction

Genre prediction uses cosine similarity between TF-IDF vectors. The summary with the highest similarity score determines the predicted genres, leveraging the dataset's preprocessed summaries.

6. Error Handling

The application includes robust error handling:

- Displays error messages for missing dataset files, failed model loading, translation errors, and audio issues.
- Validates user input to prevent processing empty summaries.
- Ensures cleanup of temporary audio files.

7. Error Handling

- Requires cleaned_movie_data.csv, which must be generated separately.
- Translation models may struggle with complex or lengthy summaries due to the 512-token limit.
- Genre prediction accuracy depends on the dataset's quality and diversity.
- Audio playback may fail on systems without proper sound drivers.

8. Conclusion

The MovieGUI application is a well-structured tool for analyzing movie summaries. It combines NLP, machine learning, and GUI programming to deliver a user-friendly experience. Future improvements could include support for additional languages, enhanced genre prediction models, and integration with online movie databases.