

Rasa Movie Chatbot
A Task-Oriented Chatbot built with Rasa



Student: Flora Koutrotsiou - 7115182400016

Course: M932 Course : Special Topics in Language Technology:

Multimodal and Dialogue Systems and Voice Assistants

Motivation & Domain

- The chatbot operates in the movie domain, which is a topic that most users are already familiar with and feel comfortable interacting with.
- In everyday situations, users often look for movie recommendations, want to know which movies are currently popular, or ask for information about a specific title.
- This domain makes it possible to combine simulated functionality with real-world data sources, allowing the chatbot to demonstrate different types of interaction.

System Overview

- The chatbot is implemented using the Rasa framework and follows a task-oriented dialogue approach.
- User inputs are processed through intent classification and entity recognition in order to identify the user's goal.
- Depending on the detected intent, the chatbot either executes a simulated action or retrieves information from external or local data sources.
- Custom actions are used to handle movie recommendations, API requests, and dataset queries, ensuring that each interaction scenario is handled appropriately.

Data Sources

- The chatbot uses a combination of simulated data and real-world data sources in order to support different interaction scenarios.
- For the genre-based recommendations, predefined movie lists are used, allowing the chatbot to simulate task execution without relying on external services.
- Trending movies are retrieved dynamically through the TMDB API, which provides up-to-date information about popular movies.
- For movie details such as plot and duration, the chatbot queries a local CSV file that contains a manually created subset of the Netflix titles dataset.

Scenario 1: Genre-based Movie Recommendations

- In this interaction scenario, the user requests movie recommendations by specifying a genre, such as comedy, action, or drama, thriller, horror, or sci-fi.

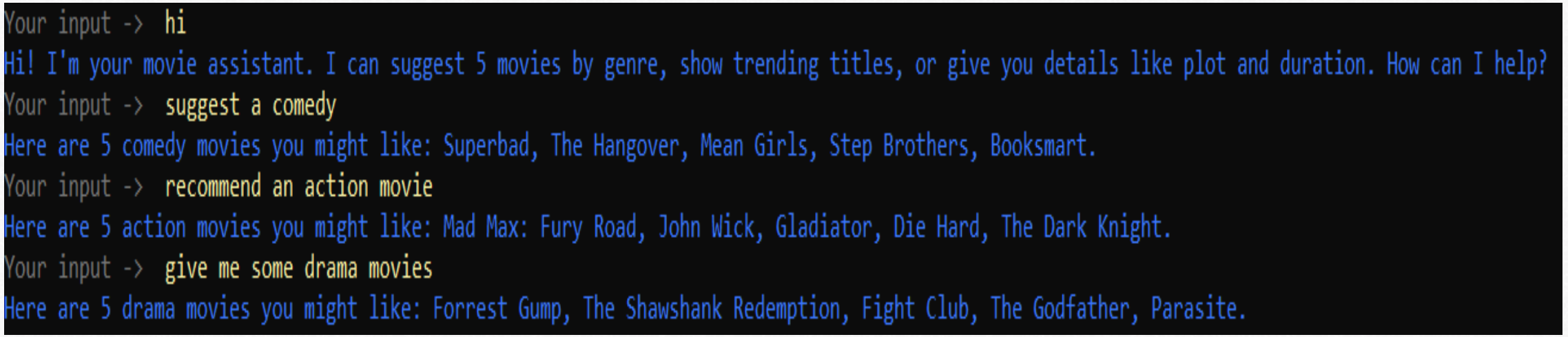
- The chatbot extracts the genre from the user's input and responds with a fixed list of five predefined movies that correspond to the selected genre.

- The movie lists are hard-coded in the custom action and do not depend on any external API or database.

- This scenario represents a simulated task execution, as it allows the chatbot to demonstrate recommendation logic and dialogue flow without accessing real-world data.

- The purpose of this scenario is to validate the basic interaction structure and intent handling of the chatbot.

Scenario 1: Genre-based Movie Recommendations (Example Dialogue)



- The screenshot presents a continuous dialogue where the user requests movie recommendations for different genres within the same session.
- The chatbot correctly adapts its response each time, returning genre-specific movie lists for comedy, action and drama.
- This interaction highlights the chatbot's ability to handle consecutive genre changes without requiring the dialogue to restart.
- The example demonstrates stable dialogue flow and consistent intent handling across multiple user turns.

Scenario 1 – Robust Handling of Unsupported Genres

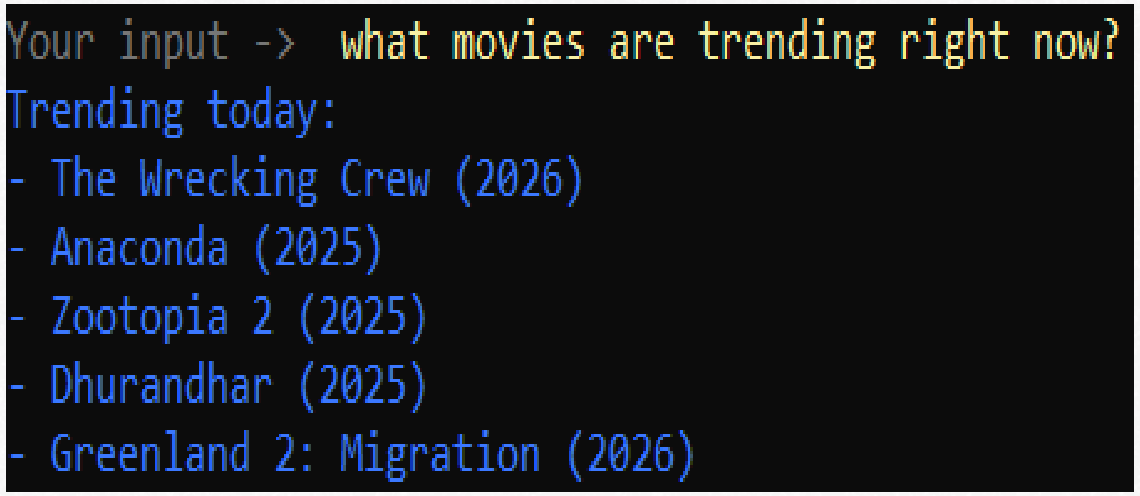
```
Your input ->  suggest a romance movie
Sure! Which genre are you interested in? (e.g., comedy, action, drama, sci-fi)
Your input ->  sci-fi
Here are 5 sci-fi movies you might like: Inception, Interstellar, The Matrix, Blade Runner 2049, Arrival.
```

- In the dialogue shown above, the user initially requests a movie recommendation for a genre that is not supported by the predefined lists.
- Instead of returning an error message or an irrelevant recommendation, the chatbot prompts the user to select a valid genre from the supported set.
- After the user provides a supported genre (sci-fi), the chatbot successfully completes the request and returns the corresponding movie recommendations.
- This example demonstrates a clarification-based strategy that allows the dialogue to recover gracefully from unsupported input and continue without restarting the interaction.

Scenario 2 – Trending Movies Right Now (TMDB API)

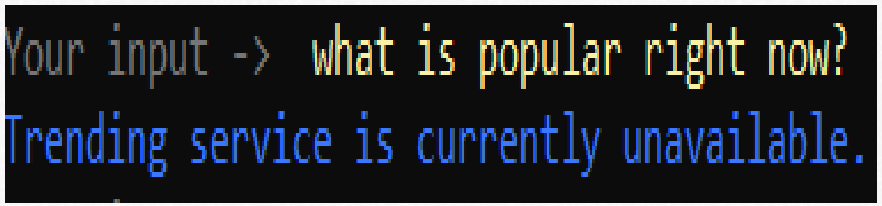
- In this interaction scenario, the chatbot provides information about movies that are currently popular.
- When the user asks for trending or popular movies, the chatbot sends a request to the TMDB (The Movie Database) API.
- The API returns real-time data about trending movies, including their titles and release years.
- The chatbot processes the API response and presents a short list of trending movies to the user.
- This scenario demonstrates real-world data integration and shows how the chatbot can dynamically retrieve up-to-date information instead of relying on predefined data.

Scenario 2 – Trending Movies Right Now (Example Dialogue)



- In the dialogue shown in this screenshot , the user directly asks for movies that are currently trending.
- The chatbot responds by fetching real-time data from the TMDB API and returns a list of the most popular movies at the moment of the dialogue, along with their release years.
- The response reflects the dynamic nature of the data, as the listed movies depend on the current trends provided by the external API.
- This example highlights how the chatbot can provide up-to-date information that is not predefined or static, in contrast to the mock-based recommendation scenario.

Scenario 2 – Robustness (TMDB API)



- In this example, the user requests trending movies while the external TMDB API is not available.
- Since the required API key is missing, the chatbot is unable to retrieve real-time data from the external service.
- Instead of failing silently or returning incorrect information, the chatbot responds with a clear message indicating that the service is currently unavailable.
- This behaviour demonstrates robust handling of real-world data dependencies and ensures that the user is informed when external resources cannot be accessed.

Scenario 3 – Movie Information from a CSV Dataset

- In this interaction scenario, the user asks for information about a specific movie, such as its plot or duration.
- The chatbot retrieves this information from a local CSV dataset rather than an external API.
- The dataset (netflix_titles.csv) is a manually created subset of the original Netflix titles dataset and contains structured movie metadata, including title, release year, duration, and description.
- When a user submits a query, the chatbot attempts to identify the movie title either from the extracted movie_title entity or directly from the raw user input.
- To make the search more robust, movie titles are normalized and a fuzzy matching mechanism is applied, allowing the chatbot to handle minor typos or variations in the user's input.
- The chatbot also takes into account the type of information requested (e.g. plot vs duration) and adjusts its response accordingly, returning the most relevant details for each query.

Scenario 3: Movie Information from CSV (Example Dialogue)

```
Your input ->  tell me about Inception
The plot of 'Inception': A thief who steals corporate secrets through the use of dream-sharing technology.
Did that information help you?
Your input ->  Yes
Great! I'm glad I could help you.
Your input ->  How long is The Irishman?
The duration of 'The Irishman' is 209 min.
Did that information help you?
Your input ->  Yes
Great! I'm glad I could help you.
Your input ->  What is the story of Matilda?
The plot of 'Matilda': A girl gifted with psychic powers uses them to deal with her crude parents.
Did that information help you?
Your input ->  Yes
Great! I'm glad I could help you.
```

- The dialogue above shows a multi-turn interaction where the user asks for different types of information about specific movies stored in a local CSV dataset.
- The chatbot correctly distinguishes between requests for a movie's plot (e.g. Inception, Matilda) and requests for duration (e.g. The Irishman), adapting its response each time.
- Each query is answered by retrieving the relevant fields from the dataset, demonstrating structured access to movie metadata rather than simple keyword-based replies.
- The interaction also illustrates a complete dialogue flow, including confirmation questions and follow-up user responses, which contributes to a more natural conversational experience.

Scenario 3 – Robust Handling of Typographical Errors and Missing Titles

```
Your input ->  tell me about Romaa
The plot of 'Roma': A year in the life of a middle-class family's maid in Mexico City in the early 1970s.
Did that information help you?
Your input ->  What is the plot of Titanic?
I couldn't find 'the plot of titani' in the dataset.
Did that information help you?
```

- In the first query shown above, the user requests information about a movie title that contains a minor typographical error (Romaa instead of Roma).
- Despite the spelling mistake, the chatbot successfully identifies the intended movie and retrieves the correct plot information from the local CSV dataset.
- In the second query, the user asks about a movie title that does not exist in the dataset (Titanic). In this case, the chatbot responds with a clear message indicating that no matching entry was found.
- This example demonstrates robust behaviour when working with structured local data, handling both noisy input and missing entries without returning incorrect information.

Conclusions and Future Improvements

- This project presented a task-oriented dialogue system developed using Rasa, focusing on the movie recommendation and information domain.
- Through three distinct interaction scenarios, the chatbot demonstrated different types of functionality, including simulated task execution, real-world API integration, and structured data querying from a local dataset.
- Robustness was addressed within each scenario, showing how the system handles unsupported input, external service unavailability, and noisy or missing data.
- Overall, the chatbot provides a functional and extensible prototype that could be further expanded with additional genres, richer datasets, and more advanced dialogue management techniques.

References and Resources

- GitHub repository: https://github.com/floraktr/rasa_movie_chatbot
- TMDB API: https://www.themoviedb.org/settings/api
- Netflix Titles Dataset: https://github.com/allenkong221/netflix-titles-dataset
- (A manually created subset of the dataset was used in this project.)