Design and Implementation of a Chatbot for Netflix Movie Recommendation

Bruno Bucalon Serra
Department of Computer Science
University of London

CONTENTS

LIST OF FIGURES

Design and Implementation of a Chatbot for Netflix Movie Recommendation

Abstract—**This report describes the development of a ChaBot for movie recommendation. The project was built on Python programming language on a Jupyter notebook, with pandas, json and nltk as main libraries.**

1. Introduction

The chatbot is designed to provide movie and television show recommendations based on user-defined criteria. The system operates over a structured dataset about netflix movies, and generates recommendations by filtering attributes such as type of title (television show or movie), duration, release year, and country the title was made in. The goal of this chatbot is to assist users in discovering relevant content efficiently through natural language.

2. Chatbot Application and Domain of Operation

The domain of the chatbot is based on content recommendation, focusing on movies and television shows from Netflix. The primary usage is to identify relevant titles based on explicit preferences from user expressed in natural language. Where according to this user inputs, queries are performed in a dataset to filter up to 5 titles and print to user according to:

1) Type of title (television show or movie).
2) Duration.
3) After or before a specific year.
4) Country the title was made in.

For the project, target audience are people from all genders and age that wants to watch a movie or television show and want up to five title recommendations

3. System Overview: Architecture, Data, Organization, and Recommendation Logic

This section presents the system architecture, data collection process, the main data transformations applied to the dataset, the functions implemented and orchestrated within the system, and the recommendation logic.

3.1 Architecture

The program is divided into three components, each responsible for a specific part of the chatbot workflow:

- content_recommender_chatbot.ipynb, with all programming code to run the chatbot;
- intents.json, a set of possible user intents and associated patterns
- netflix_titles.json, with the data used for recommendation.

During execution, the chatbot loads the dataset and intent definitions, processes user input, applies the recommendation logic over the dataset, and returns the filtered results to the user.

3.2 Data and Organization

The data was collected from Kaggle website about Netflix movies and television shows.

As the first steps of data collection and organization, several libraries were imported to support data handling, text processing, and interaction control. The `pandas` library was used for data manipulation, while `json` to handle structured data formats and `re` library for text pattern matching. Natural language processing tasks were supported by the `nltk` library, including tokenization through `word_tokenize`, lemmatization using `WordNetLemmatizer`, removal of non-informative terms with `stopwords`. Finally, the `time` library was used for basic execution control and response timing.

Then, a set of functions were created in order to import the dataset and apply filters that will be used in the main loop (`select_movie`), as well as the application of natural language processing, stop words removal, tokenization, and lemmatization. (`tokenize_text`, `stopwords_removal`, `lemmatize_text`).

Finally, a `chat_bot` function was implemented, in which the system is structured according to predefined user states. Several variables (such as `content_type`, `duration_selected`, and others) are used to control the chatbot flow, along with a dictionary that stores all user responses throughout the interaction. Once all required responses have been collected and the dictionary is complete, the `select_movie` function is invoked. This function applies the corresponding filters to the dataset using the values stored in the dictionary and outputs the top five recommendations according to the user's preferences [Fig. 1.].

4. Test Cases and Chatbot Behaviour

In this section, three test cases will be showed in order to verify if a specific mechanism is properly being executed.

4.1 Test Case 1: End-to-End Recommendation Workflow

The objective of this test case is to verify whether the chatbot completes the full interaction flow and produces coherent recommendations based on the user inputs. In this scenario, the user specifies the content type, duration, release period, and country. The chatbot is expected to guide the conversation through all required steps and, at the end, return a set of movie recommendations that are correctly according to user responses.

After the interaction, the chatbot successfully calls the recommendation function and outputs a list of titles that match the selected criteria.

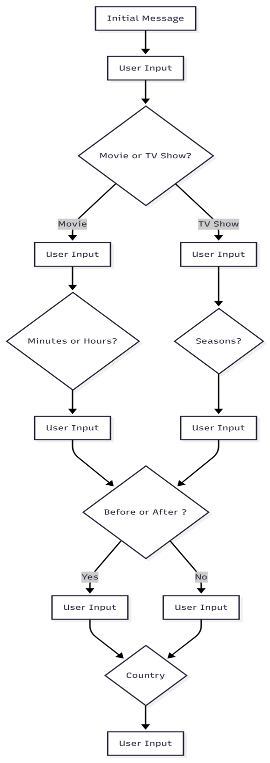

Fig. 1. User flow in chatbot.

4.2 *Test Case 2: Duration Handling According to Content Type*

When the user selects *movie*, the chatbot is expected to interpret the duration in terms of *hours* or *minutes*. In contrast, when the user selects *TV show*, the duration should always be interpreted and stored as *seasons*, regardless of the numeric value provided.

For movies, inputs such as *"2 hours"* or *"120 minutes"* result in duration values being stored as *"2 hours"* or *"120 minutes"*. For TV shows, an input such as *"3"* results in the duration being stored as *"3 seasons"*.

Observed state (dictionary snapshot – movie):

```
{
  'content_type': 'movie',
  'duration': '2 hours',
  'year': None,
  'after_before_year': None,
  'country': None
}
```

4.3 *Test Case 3: Reset Functionality and State Clearance*

The objective of this test case is to ensure that the chatbot correctly resets its internal state when the user requests a restart.

When the user types *"reset"* or *"restart"*, all previously stored preferences are cleared, and the conversation returns to its initial state.

Observed state after reset (dictionary snapshot):

```
{
  'content_type': None,
  'duration': None,
  'year': None,
  'after_before_year': None,
  'country': None
}
```

5. Advanced Techniques

In this final section, three advanced techniques will be presented in the subsections below: stateful rule-based dialogue, validations and fallback, and intentional delay using time library.

5.1 *Stateful Rule-Based Dialogue*

The chatbot implements a stateful rule-based dialogue management approach using explicit state variables to control multi-turn conversational flow. Intent recognition and entity extraction are performed through regular expressions, improving deterministic parsing of content type, duration, and temporal constraints.

5.2 *Validations and Fallback Mechanisms*

The system incorporates input validation and fallback mechanisms to ensure robustness against inconsistent or unrecognized responses. Additionally, artificial response latency is introduced via time.sleep() prior to recommendation delivery.

5.3 *Time Delay*

According to experimentation research, overly fast system responses may negatively affect perceived reliability and suggest computation failure; intentional delays were added using the `time` library can improve user trust [1].

The overall architecture follows a sequential, deterministic processing pipeline, characteristic of classical rule-based conversational systems.

6. Conclusion and Reflection

Due to prior experience with Python, most of the weekly activities were straightforward to implement. However, from week 5 onward, when regular expressions were introduced, the level of difficulty increased. Understanding the semantics of each expression and applying them correctly proved challenging. Consequently, this topic represented the most significant difficulty of the course. I overcame this challenge by conducting an in-depth study of the different types of regular expressions, supported by additional Python reference books.

Overall, my feedback is positive, as I was able to implement a different and improved chatbot while also integrating Pandas for dataset manipulation and filtering.

References

[1] Ron Kohavi, Diane Tang, and Ya Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, Cambridge, UK, 2020.