Movie Finder: Human-Machine Dialogue project

Sara Callaioli
207484
sara.callaioli@studenti.unitn.it

1 Introduction

1.1 Model

Movie finder is an Alexa skill that allows the user to ask for information about movies and people (cast or crew of a movie). The dialogue is based on a question-answering model with a mixed-dialogue purpose implemented using Rasa.

1.2 Domain

The questions that Movie Finder can handle are:

- Which movie provider you can find a certain movie
- The actors starring in a movie
- Who directed or wrote a movie
- What a movie is about
- How a film is rated
- Information about a person

Responses to these questions always include the movie title or the person's name to give the user feedback on what the agent has understood.

1.3 User Group

The target users of Movie Finder are people that want rapid and straightforward information about a movie or people. The idea is to speed up the delivery of this type of information compared to searching the internet the question.

2 Conversation Design

Thanks to the mixed-dialogue model, the dialogues that the system can handle are relatively long. Despite this, the user can always ask questions randomly in the conversation, and the agent will respond to the last question. The agent will also provide help if the user doesn't know what to ask to the skill listing the types of questions they can ask.

2.1 Initiatives

The agent takes the initiative after every question of the user. For example, when the users ask something about a movie, the agent continues the conversation proposing other information about the same movie.

2.2 Acknowledgments

To show that the bot has understood the question, it will start the response with some short utterances such as "Ok", "Let me check", "Mmm...". This method is used especially if the answer needs more time to be elaborated, such as the plot computation (a summarization) or the calculation of the rates (with several calls to different API).

2.3 Ground-taking

The agent always includes the entity from the question in the response. This implementation gives users confirmation that the subject of their request has been understood and elaborated. On top of that, the response always resumes the question. So in that way, the user has confirmation the topic is also correct.

2.4 Fall-back Responses

If the user asks questions not within Movie Finder's purview, the agent will say, "Sorry, but I can't answer your request because I only deal with questions about movies. So instead, ask me something like, Where can I find Eternals?"

2.5 Error Recovery

If something went wrong with a request to the API or, in general, with computation, the agent would

respond with "Ops! This is embarrassing... Something went wrong. Please ask me your question again."

3 Data Description and Analysis

3.1 NLU Data

To generate the NLU data, I implemented a generator that writes the nlu.yml file joining the static part of the file (where there are no entities) and the dynamic part. The static part is manually generated with the most commonly used phrases for greetings, goodbyes, affirmation, denying, thanking, request of help, the bot status, and some out-of-scope. The dynamic part focuses on the intents with an entity in their phrases which are:

- movie provider
- movie actors
- movie plot
- movie reviews
- movie director
- movie writer

general info about a person In this part, the generator merges each request of a single intent with the entity needed. For example, the intent $\text{movie}_{p}lothasfourtrainingphrases$:

- "What is the plot of movie"
- "Tell me something about movie"
- "Tell me the plot of movie"
- "What is movie about?"
- "Tell me what the movie is about."

In this case, movie is replaced with a string retrieved by an API call.

Each phrase of each intent is repeated around 20 times with different movies or people as an entity. This technique allows me to have an extensive training set of requests for each intent with a lot of variety to perform the model's training.

As can be understood from what has been said, the entities are 2: "movie" and "person". To have more completeness in the entity "people" were considered both actors, actresses, and the cast.

Because a mixed-dialogue is the goal of this agent, two slots are considered:

- $last_{q}uestion$: to keep track of the last question proposed to the user while asking him if he wants it to be performed or not.
- $movie$: to remember the movie the user and the agent are talking about and bring it to the next question implicitly.

3.2 Domain data

The data are collected from 3 different API:

- TMDB[1]: The Movie Database (TMDB) API, where the majority of the data are extracted from and the main API
- JustWatch[2]: an unofficial JustWatch library that performs API calls to the site. It is used exclusively to retrieve information about movie providers. Although TMDB also has movie providers, JustWatch allows you to have more movie prices. This information is not used at the moment, but it may be useful in future works.
- IMDB[3]: Internet Movie Database (IMDb) API is mainly used to retrieve the rates from other movie rating sites.

In addition to being used to satisfy user requests, the APIs create the training set by filling the entities. For example, in the case of films, a call is made to the top 50 pages of the Most Popular Movies on TMDB to collect a good amount of movie names. In the case of people, the call is also made to Most Popular Movies, but only to the first page. This is sufficient because for every movie on each page, both the cast and crew people are considered.

4 Conversation Model

The model is implemented with the usage of Rasa, an opensource machine learning framework to automate text and voicebased conversation.

4.1 Rasa Intents

The intents implemented in Movie Finder are both of general use and movie-specific. The general intents are:

- greet

[1] https://developers.themoviedb.org/3/getting-started/introduction

[2] https://github.com/dawoudt/JustWatchAPI

[3] https://imdb-api.com

- goodbye
- affirm
- deny
- bot_status
- bot_challenge
- out_of_scope
- help
- thanks

While the intents specific on movies and people are:

- movie_provider: the user wants to know on which provider he can find a particular movie
- movie_director: the user wants to know who is the director of a movie
- movie_writer: the user wants to know who is the writer of a movie
- movie_plot: the user wants to know who is the plot of a movie
- movie_reviews: the user wants to know how a movie is rated
- all_stars_movie: the user wants to know the actor and actresses in a movie
- person_info: the user wants to know some general information about a person

4.2 Rasa Actions

The actions implemented in Movie Finders are divided into general utter responses and custom actions.

The general responses are:

- utter_greet
- utter_goodbye
- utter_movie
- utter_out_of_scope
- utter_help

The custom actions are:

- action_affirm: handle if user positively respond to an agent question, triggering the last question (and action) that was asked
- action_deny: handle if user negatively respond to an agent question, resetting the slots
- action_movie_provider: give the user a list of movies providers of a specific movie if available. First, it delivers the providers with subscription, and then the ones you can rent from
- action_movie_director: return the director (or a list of directors) of a specific movie
- action_movie_writer: return the writers (or a list of writers) of a specific movie
- action_movie_plot: return a summary of the movie plot
- action_movie_reviews: provides the movie rating on TMDB and, if available, the average of the ratings on other ratings movie sites.
- action_all_stars_movie: return a list with the most famous actor and actress in a movie.
- action_person_info: tell some information about a person, such as their roles in movies and what they are known for.

All those actions have an error handling part implemented where the agent tells the users that no results are available for the movie. In the response, the bot always includes the entity value it caught to let the user know what the bot understood.

Instead, if there were technical problems while creating the response, it returns a generic error message and asks the user to repeat the question.

To make the model mixed-dialogue the actions, at the end of the response, suggest the following question about the same movie. The actions are connected as in Figure 1.

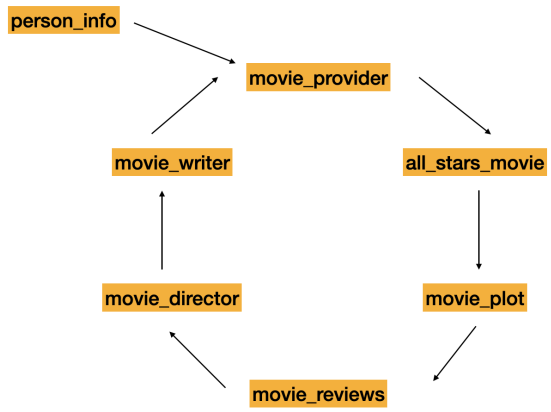

Figure 1: Action schema

4.3 Vocal model

The Rasa model is connected to Alexa, using it as its voice channel. To do that, I created a skill named *sara movie finder* on Alexa Developer Console and added in my model a connector file. In this file, there is a welcome message for when the Alexa skill is started saying, "Hello! Welcome to Movie Finder skill. If you want to know what I can do, ask me, 'What can you do?' ".

4.4 Pipeline configuration

The pipeline configuration has the following main components:

- WhitespaceTokenizer, which produce tokens using whitespaces as a separator
- RegexFeaturize, which creates a vector representation of user message using regular expressions
- CountVectorsFeaturizer, which creates bag-of-words representation of user messages, intents, and responses.
- DIETClassifier, which is a multi-task architecture for intent classification and entity recognition.
- ResponseSelector, which is used to build a response retrieval model to directly predict a bot response from a set of candidate responses
- FallbackClassifier, which classifies a message with the intent nlu_fallback if the NLU intent classification scores are ambiguous

5 Evaluation

An evaluation of the model has been done by using the Rasa testing option and a human evaluation.

The Rasa testing is done on a series of test_stories with different phrases from the one used to train the models and other entities. The results are successful both for intents and entities, as shown in Tables 1 and 2. Moreover, they demonstrate that the model has a proper generalization. Nevertheless, the results probably need deeper testing with even more different examples of stories to stress out the model.

Table 1: DIET Classifier

precision	1.0
recall	1.0
f1-score	1.0

Table 2: Intent Classifier

precision	0.9995
recall	0.9994
f1-score	0.9994

The human evaluation is done by asking people to test the skill on Alexa and then fill out a questionnaire. In this form, the tester is asked to respond to several questions, such as:

- Was the chatbot engaging?
- Did the chatbot understand the scope of your questions?
- Did the chatbot understand which subject you were referring to?
- Was the chatbot able to remember the movie name during the dialogue?
- Did it happen that the chatbot wasn't able to find results for your question?
- If it happened, how did the chatbot manage those cases?
- How did the chatbot manage out-of-scope questions?

In general, the quality of the conversation seems good, and the user is satisfied with the conversation (around 3.5 of engagement of the conversation).

The most significant results are that the agent recognizes the intent almost every time, but it has difficulties with entity recognition. This is probably because English is not the mother language of the people it interacts with, so the pronunciation may be incorrect.

6 Conclusion

In conclusion, the model can successfully generalize the data, and the user finds it engaging. However, it would be interesting to add diversity in the testing stories, as said before.

Possible future works are:

- Ask the user their favorite movies providers or which movie providers they are subscribed to so that we can suggest them first
- Ask the follow-up questions randomly by checking a list of already-done-questions
- Add more information the user can demand about a movie (e.g. similar movies, suggestions based on the genre, etc..)