Features and Implementation Documentation for Voice Interaction with YouTube Music

1. Introduction

This system was designed to offer intuitive voice interactions, allowing the user to manage songs, playlists and playback settings. The structure is based on well-defined Intents, Slots and Entities, ensuring a clear understanding of the user's intentions. Throughout this document there will be a detailed description of the functionalities, with specific explanations for each intent.

2. Functionalities and Implemented Intents

Below is a detailed description of each intent, including examples and explanations on how it was implemented.

Whenever something is said and the confidence is at or below 45%, we receive feedback from the application that it did not understand the request. If confidence is above 45% and below 80%, the application asks for confirmation to perform the action. If it is above 80%, the action is executed immediately.

Feedback is returned to the user whenever an action is to be executed or if execution fails.

2.1. Intent: goodbye

- **Function:** End interaction with the assistant.
- **Examples**:
  - *“Vou sair, obrigado pela ajuda.”*
  - *“Fechar o programa agora.”*
- **Explanation:**
  - A simple structure was chosen to capture common farewell expressions.
  - This intent does not require slots or entities.

2.2. Intent: search_music

- **Function:** Search and play specific songs.
- **Examples**:
  - *“Põe a tocar a música Shape of You do cantor Ed Sheeran.”*
  - *“Procura por Despacito do cantor Luis Fonsi.”*
- **Explanation:**
  - The song and artist entities were included for precise searching.
  - We used lookup tables with examples of artists and songs to enhance the precise recognition of entities.
  - The grammar was designed to recognize commands like “toca” and “procura por”.

2.3. Intent: play_playlist

- **Function**: Start a specific playlist.
- **Examples**:
  - *“Quero uma playlist de “Pop”.”*
  - *“Toca a playlist de "Rock".”*
- **Explanation**:
  - The playlist entity was used to identify the name of the playlist.
  - We used lookup tables with examples of playlists to enhance the precise recognition of playlist entity.

2.4. Intent: control_music

- **Function:** Pause or resume the music.
- **Examples:**
  - *“Pausa a música.”*
  - *“Quero continuar a ouvir a música.”*
  - *“Preciso que interrompas a música.”*
  - *“Volta à música que estava.”*
- **Explanation**:
  - Entity action with values pause or resume were used to distinguish these two actions.

2.5. Intent: change_track

- **Function:** Change the track being played.
- **Examples:**
  - *Próxima faixa.*
  - *Estou farto desta música.*
  - *Não gosto desta música.*
  - *Volta a musica anterior*
  - *Reproduz a mesma música outra vez, por favor.*
  - *Repetir música.*
- **Explanation**:
  - We used the direction entity, such as “next,” “previous,” and “same,” to map different actions

2.6. Intent: adjust_volume

- **Function:** Control playback volume.
- **Examples:**
  - *Aumenta o volume*
  - *Diminui o volume*
  - *A música está muito baixa.*
  - *A música está muito alta.*
  - *Tira/Desativa o som*
  - *Ativa o som*
  - *Põe o som de volta por favor.*
- **Explanation**:
  - Entity action with values increase, decrease, mute or unmute were used to distinguish these four actions.

2.7. Intent: set_mode

- **Function**: Configure playback modes (shuffle, repeat).
- **Examples**:
    - *Ativa o modo aleatório para as músicas.*
    - *Dá para desligar o modo aleatório?*
    - *Podes misturar as músicas?*
    - *Desativar o modo aleatório.*
- **Explanation**:
    - We used the mode entity, such as “shuffle_on,” “shuffle_off,” “repeat_one,” “repeat_all,” and “repeat_off,” to manage playback settings.

2.8. Intent: add_to_favorites

- **Function**: Like/Mark as favorite the current music music.
- **Examples**:
    - *Dá like na música*
    - *Adiciona a música aos favoritos.*
- **Explanation**:
    - No slots required, just the intention to favorite the current song.

2.9. Intent: confirm_action

- **Function:** Confirm or cancel actions.
- **Examples:**
    - Confirmo a opção escolhida.
    - Não, desfaz a escolha.
- **Explanation:**
    - We used the action entity, such as “confirm” and “cancel,” to handle user decisions.

2.10. Intent: which_music_is_playing

- **Function:** Tell the current song playing.
- **Examples**:
    - *Que música está a tocar?*
- **Explanation**:
    - Direct response with song name and artist, when available.

2.11. Intent: add_music_to_queue

- **Function:** Add songs to the queue.
- **Examples:**
    - *Põe a tocar a música Shape of You do cantor Ed Sheeran a seguir.*
- **Explanation:**
    - The song and artist entities were included for precise searching.

2.12. Intent: add_music_to_playlist

- **Function:** Add songs to a specific playlist.
- **Examples:**
  - *“ Adiciona a música Someone Like You da cantora Adele à playlist de Pop.”*
- **Explanation:**
  - The song and artist entities were included for precise searching and playlist entity was included for playlist search.

2.13. Intent: help

- **Function:**
  - Reveal all possible functionalities.
  - Explain all functionalities.
  - Explain one functionality.
- **Examples:**
  - *O que posso fazer?*
  - *Explica-me todas as opções.*
  - *Ajuda com pesquisar uma música*
- **Explanation:**
  - The help_option entity was added to allow users to select the specific functionality they need assistance with.

4. Adaptations for Speech Technologies

To optimize the assistant for speech technologies, we included a diverse range of phrases for each intent, ensuring robust recognition of user commands in various contexts.

Additionally, for certain entities, synonyms were implemented to map different expressions to the same entity. For example, terms like “pausa,” “pausar” and “interromper” were mapped to the “pause” entity, while terms such as “continua” and “retomar” were mapped to the “resume” entity, improving consistency and enhancing the user experience in voice interactions.

5. Use Instructions

To start the assistant, first, train the Rasa model by navigating to the rasaDemo folder and running the command “rasa train”. After this, return to the previous directory and execute the “start.ps1” script in the Miniconda PowerShell to start the Rasa server, the Fusion Engine, the MMI Framework and the Web Page for the voice detection.

Next, initialize the application by navigating to the app folder. You can optionally create a .env file to store your EMAIL and PASSWORD credentials; otherwise, the application will prompt you for these when it starts. Then, create a virtual environment, activate it, and install the dependencies listed in requirements.txt using “pip install -r requirements.txt”. Finally, run the application by executing “python main.py” from within the virtual environment.

6. Conclusion

The system was designed to offer fluid and flexible interaction, adapted to natural commands. Each intent and entity has been carefully configured to minimize ambiguities, promoting an optimized experience for Portuguese-speaking users.

6. References

- Class's PowerPoints
- Videos in the class's PowerPoints
- https://rasa.com/docs/rasa/nlu-training-data/