MOT - THE MOVIE CHATBOT

Olga Pagnotta

All the code can be found at the following link https://github.com/chatMot/ChatMot.git.

Introduction

Chatbots: a first sight

Chatbots are Natural Language Processing models that enable computers to decode and even mimic the way humans communicate. They can complete tasks, achieving goals and delivering results.

Dialog systems' technology is almost everywhere these days, from the smart speakers at home to messaging applications in the workplace. Ideal chatbots should understand dialogue with human and answer in an appropriate way: such an architecture could be based on a trained neural network using real dialog and be able to converse with humans to deliver data or information. Essentially it should be trained on a corpus to answer to a specific stimulus. The latest AI chatbots are often referred to as "virtual assistants" or "virtual agents." They can use audio input, such as Apple's Siri, Google Assistant and Amazon Alexa, or interact with the user via SMS text messaging. Either way, the final user is able to ask questions about what he needs in a conversational way, and the chatbot can help refine his search through responses and follow-up questions.[1]

Today's AI chatbots use natural language understanding (NLU) to discern the user's need. Then they use advanced AI tools to determine what the user is trying to accomplish. These technologies rely on machine learning and deep learning to develop an increasingly granular knowledge base of questions and responses that are based on user interactions. This improves their ability to predict user needs accurately and respond correctly over time.
Anyway, it is important to highlight the fact that real chatbots are task oriented, meaning that they are trained in a specific domain to manage a certain activity.

AI chatbots are commonly used in social media messaging apps, standalone messaging platforms, or applications on websites. Some typical use cases include:

- Finding local restaurants and providing directions
- Defining fields within forms and financial applications
- Getting answers to healthcare questions and scheduling appointments
- Receiving general customer service help from a favourite brand
- Setting a reminder to do a task based on time or location
- Displaying real-time weather conditions and relevant clothing recommendations

[1] What is a chatbot?, available at https://www.ibm.com/topics/chatbots.

Generic architecture:

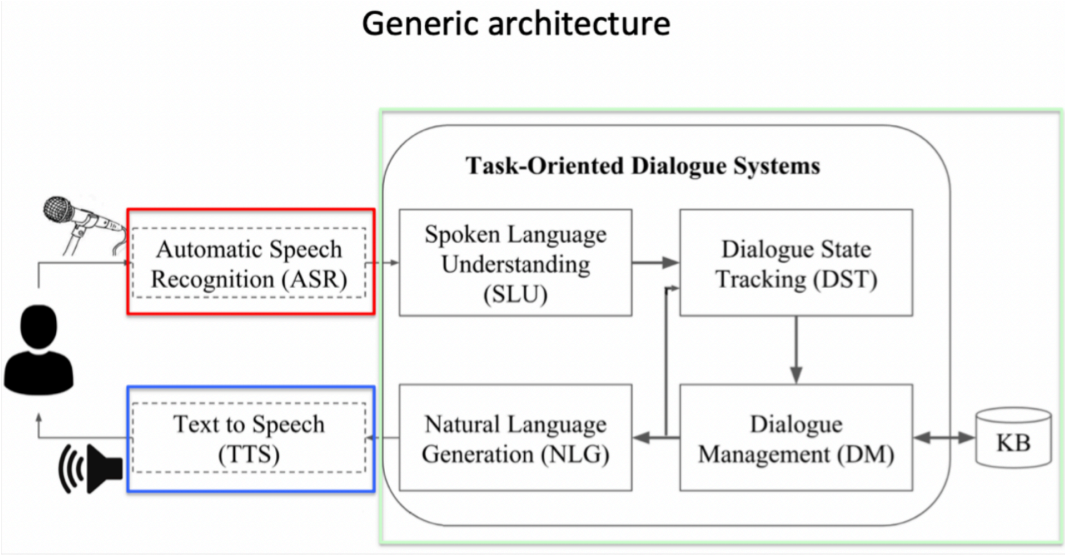


Figure 1 Generic dialog system architecture (Spoken Dialogue Systems set of slides of the NLP course)

First, on one side we find ASR (Automatic speech recognition). The system aims at finding the most likely sentence out of all sentences in the language L given some acoustic input O, which is a sequence of individual symbols or observations. This means that finding the most probable sentence W given some observation sequence O can be computed by taking the product of two probabilities for each sentence and choosing the sentence for which this product is greatest.

$$\bar{W} = \underset{W}{argmax}\, P(W \mid O) = \underset{W}{argmax}\, \frac{P(O \mid W)\, P(W)}{P(O)} = \underset{W}{argmax}\, P(O \mid W)\, P(W)$$ [2]

The main actors include a language model that computes the so-called prior probability P(W), that allows to estimate the most likely orderings of words in a given language; a pronunciation model for each word in that sequence; an acoustic model that computes the observation likelihood P(O|W), that allows to estimate the probability of an input sequence of acoustic observations given each possible words sequence W. When we receive some spoken input, our goal would be to find the most likely sequence of text that maximizes the words probability given a speech-acoustic input.[3]
Nowadays the approach is the end-to-end ASR: they take a sequence of audio inputs and return a sequence of textual outputs and all components of the architecture are trained jointly towards the same goal, while before each part of the ASR model was composed by a neural network, each of which had to be trained individually on specific tasks.
On the other hand, we find TTS (text-to-speech synthesis), for reading output to users. It is the inverse process of the ASR model.[4]

[2] Tamburini F. (2022). Neural Models for the Automatic Processing of Italian, Pàtron.
[3] *ibidem*
[4] *ibidem*

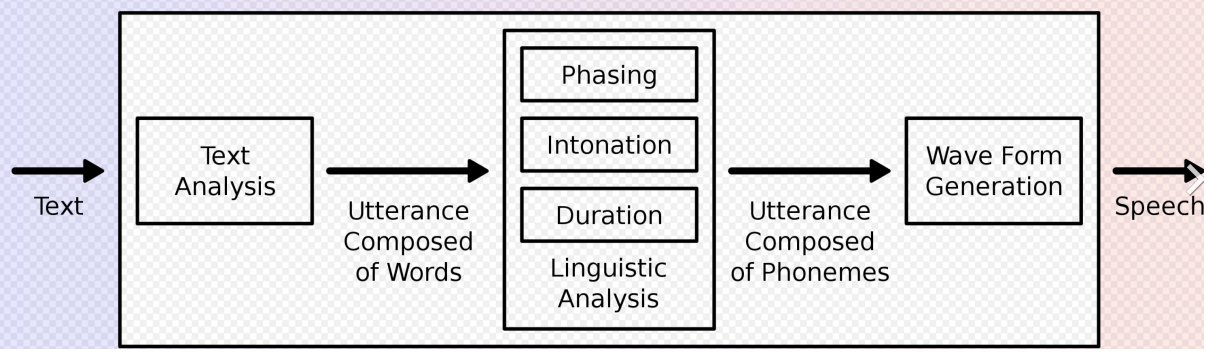


Figure 2 TTS model (Wikipedia)

The second step is intrinsically connected with NLP: Spoken Language Understanding (SLP) and Natural Language Generation (NLG) for synthesizing the written form of the answer.

Lastly, the main actors become the Dialogue State Tracking, to monitor the users' intentional states, and the Dialogue Management, which maintains some state variables, such as the dialog history, the latest unanswered question, etc. It sends instructions to other parts of the Dialog System to process the question, after having its formal representation, and to produce a useful answer to it.

Task oriented Chatbots:

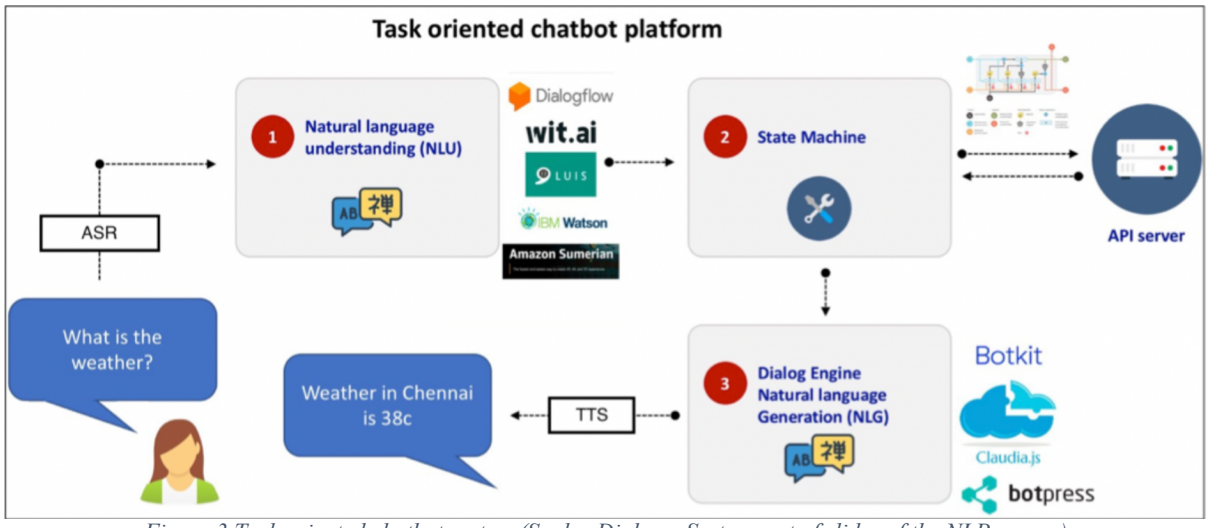


Figure 3 Task oriented chatbot system (SpokenDialogueSystems set of slides of the NLP course)

We have three main steps in a task oriented chatbot platform:

1. NLU. Natural language processing, and thus natural language understanding, can be delegated to some powerful platforms, which perform all the necessary actions to process user's inputs and convert it into structured data (such as intents or entities).
2. State Machine. It keeps the dialogue state, an important step for the creation of a continuous flow in the conversation. For example, it keeps track of the empty slots in the intent frame to be filled in a second moment through a series of question asked by the chatbot to obtain needed information.
3. NLG. Natural Language Generation for producing the sentence which will then be parsed by the TTS to generate the needed answer.

DialogFlow CX

Dialogflow is a natural language understanding platform used to design and integrate a conversational user interface into mobile apps, web applications, devices, bots, interactive voice response systems and related uses.[5]

Google Dialogflow recently introduced Dialogflow CX (Customer Experience), the version we decided to use. The older version of Dialogflow has been renamed to Dialogflow ES (Essentials). Dialogflow CX provides a new way of designing virtual agents, taking a state machine approach to agent design. This gives a clear and explicit control over a conversation, a better end-user experience, and a better development workflow. Some of the new functionalities that CX offer are:

- a visual flow interface, which allows to visualize flows and pages in a graph-like structure[6], to have a clearer view of the conversation;
- a state machine model[7], which allows developers to reuse intents, intuitively define transitions and data conditions, and handle supplemental questions;
- separate flows, allowing the developer to divide the agent into smaller conversation topics.[8]

It offers a wide variety of components to achieve the intended goal, but we will present only a few of them, the fundamental ones.

- *Agent*
  First, to create a dialog system on Dialogflow, we need to create a Dialogflow agent, a virtual agent that handles concurrent conversations with your end-users. It is a natural language understanding module that understands human language and its characteristics. Dialogflow is responsible for the first step in the task oriented chatbot framework, NLU, since it translates end-user text during a conversation to structured data.[9]
- *Flows*
  Flows are used for defining conversation topics in complex dialogs and the associated conversational paths. They provide better conversational control.[10]
- *Pages*
  A Dialogflow CX conversation (session) can be described and visualized as a state machine. The states of a CX session are represented by pages. For each flow, you define many pages, where your combined pages can handle a complete conversation on the topics the flow is designed for.[11]
- *Entity types*

---

[5] Dialogflow on Wikipedia, https://en.wikipedia.org/wiki/Dialogflow.
[6] Dialogflow editions, https://cloud.google.com/dialogflow/docs/editions.
[7] *ibidem*
[8] Dialogflow CX vs ES: A Complete Overview, available at https://chatbotsjournal.com/dialogflow-cx-vs-es-a-complete-overview-33580eca529c.
[9] Dialogflow documentation, https://cloud.google.com/dialogflow/cx/docs/basics.
[10] *ibidem*
[11] *ibidem*

Entity types are used to control how data from end-user input is extracted.

- *Parameters*
  Parameters are used to capture and reference values that have been supplied by the end-user during a session. Each parameter has a name and an entity type.
- *Intents*
  An intent categorizes an end-user's intention for one conversation turn.
- *Fulfillments*
  For an agent's conversational turn, the agent must respond to the end-user with an answer to a question, a query for information, or session termination. Your agent may also need to contact your service to generate dynamic responses or take actions for a turn. Fulfillment is used to accomplish all of this. In our chatbot they mostly take the form of customized payloads, supplied in a JSON format.[12]
- *Interactions*
  For each conversational turn, an interaction takes place. During an interaction, an end-user sends input to Dialogflow, and Dialogflow sends a response. We have used the Dialogflow API. The interaction takes places according to the following steps:
  1. The end-user types or says something, known as *end-user input*.
  2. The user interface receives the input and forwards it to the Dialogflow API in a detect intent request.
  3. The Dialogflow API receives the detect intent request. It matches the input to an intent or form parameter, sets parameters as needed, and updates session state.
  4. Dialogflow creates a detect intent response, using the static response defined in the agent. Dialogflow sends a detect intent response to the user interface.
  5. The user interface receives the detect intent response and forwards the text or audio response to the end-user.
  6. The end-user sees or hears the response.

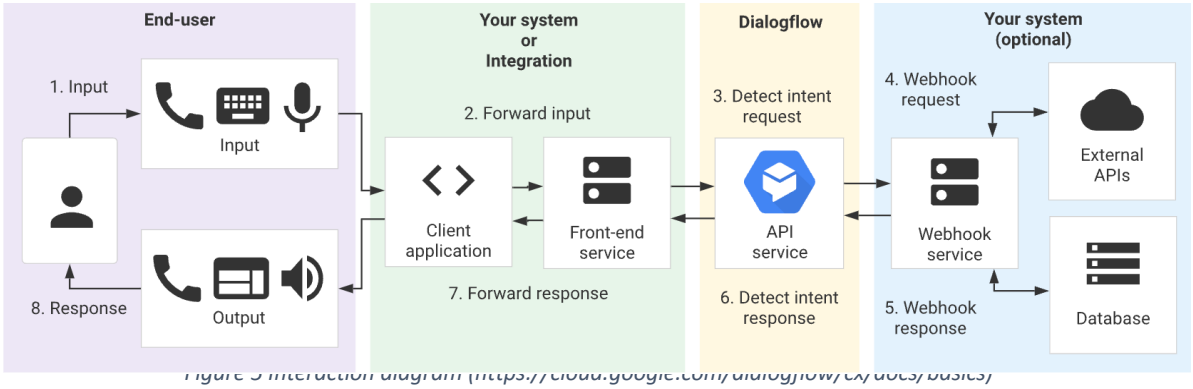


Figure 4 The interaction process in Dialogflow

[12] Dialogflow documentation, https://cloud.google.com/dialogflow/cx/docs/concept/fulfillment#payload.

CHATMOT: purpose and workflow

Have you ever been stuck on the Netflix menu for minutes trying to find the right movie for your chill time? It happened to me many times, therefore we have thought of a possible solution for this problem, and we came up with Mot, the Movie Chatbot.
ChatMot is a personal assistant that simply suggests users a Netflix movie to watch.
Movies' categories and titles, which were taken from an already existing dataset of Netflix America, available at the following link https://www.datacamp.com/workspace/templates/dataset-python-netflix-movie-data.
Training phrases were added in the intents in order to better understand the users' requests.

The Workflow:

ChatMot deals with some movies and film categories so as a first step, I have created a schema of the dialog structure by drawing the different flows. From the *Default Start Flow*, the virtual agent moves to the *Movies flow*, which can lead the user to all the other 3 flows corresponding to the Netflix movies categories chosen for the chatbot.

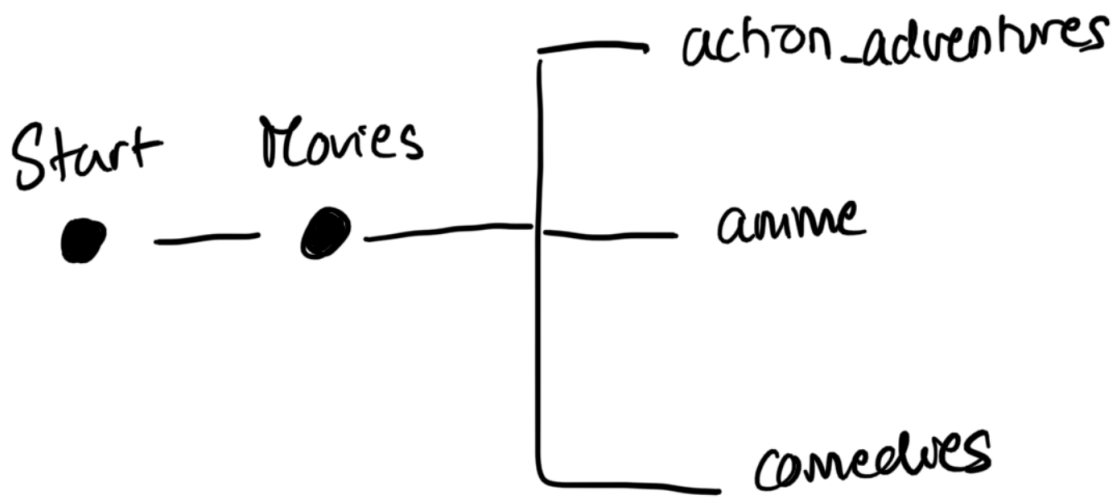


Figure 6 Schema of the Chatbot's flows

Flows: structure

Default start flow

The root-flow for the creation of any chatbot is the *default start flow.* In ChatMot it consists in the following steps, which go from the start, welcoming the user by asking his/her name, to the movies' flow:

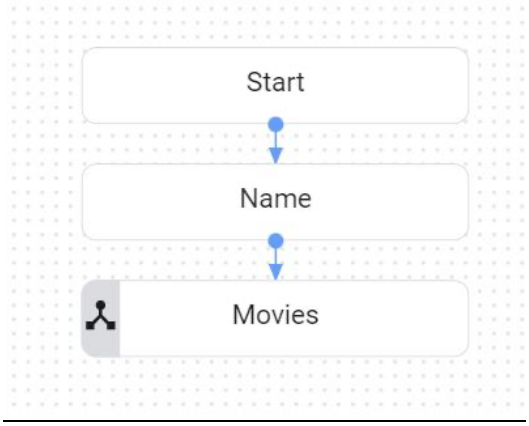


Figure 7 Default start flow

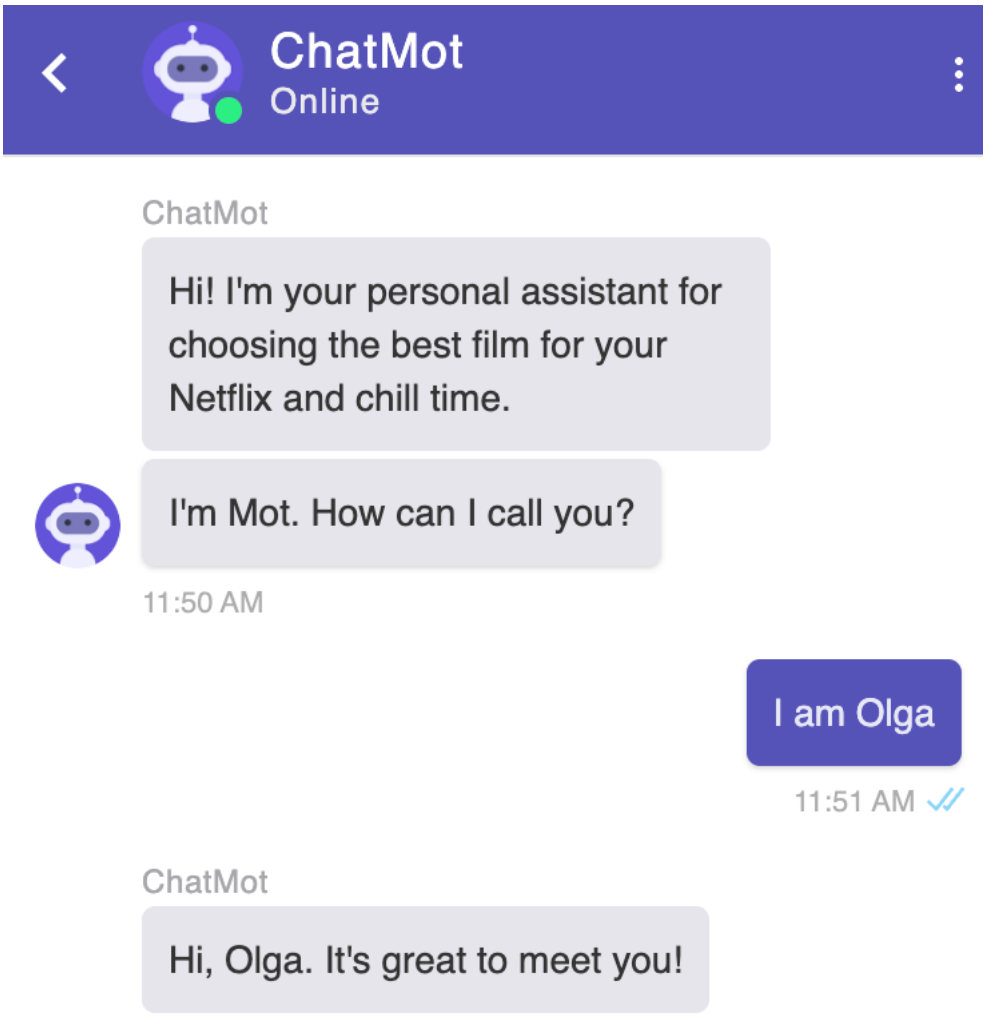


Figure 8 Welcome message from Mot

Movies flow

The second step is the choice of the movies categories by the user, following the below flow:

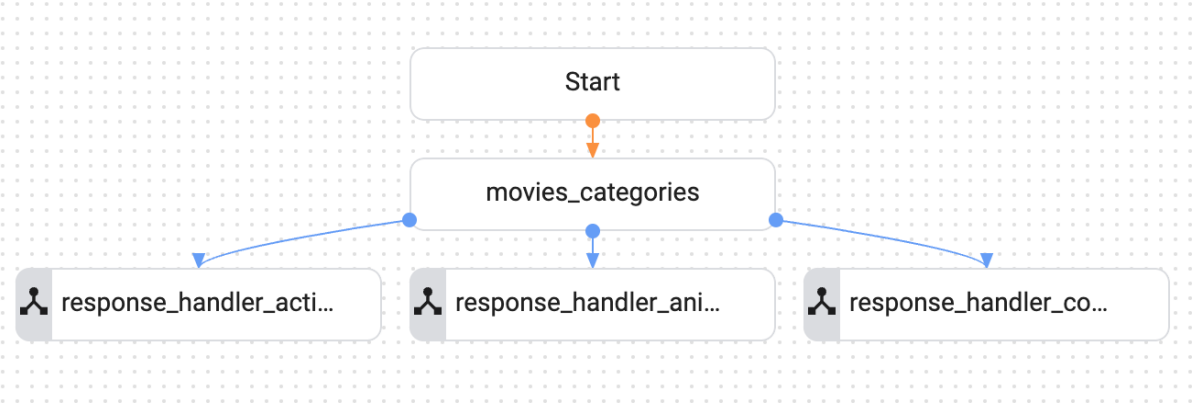
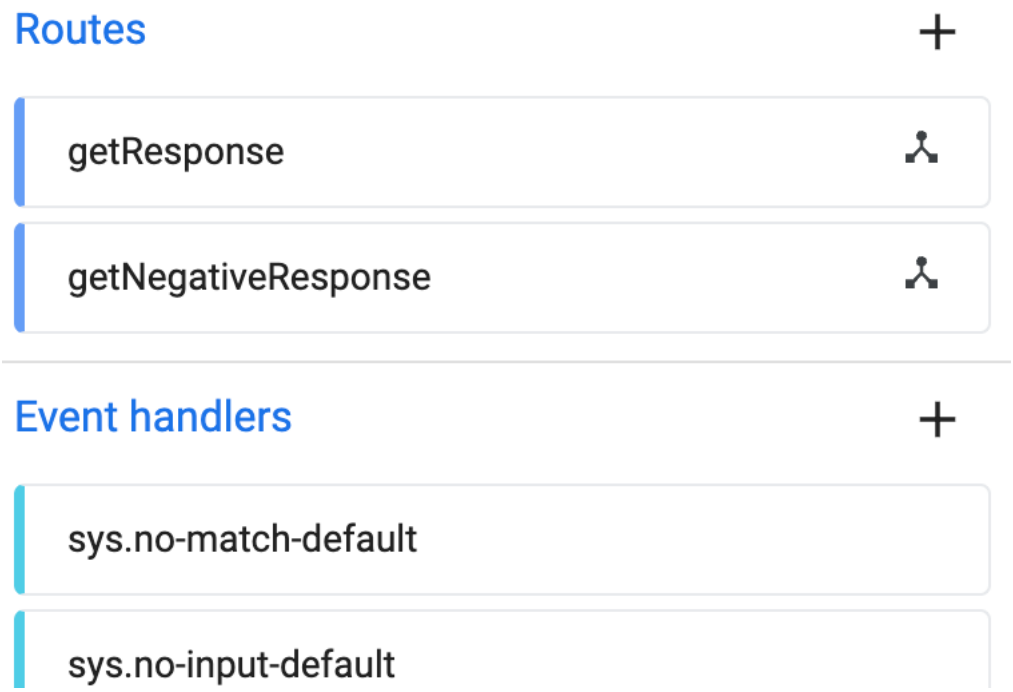


Figure 9 Movies flow

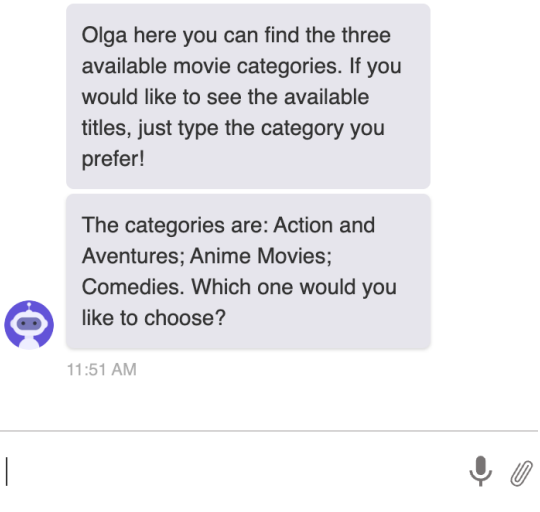
Figure 10 Choice of the movies categories

For interacting with my chatbot I have used Kommunicate,[13] a live-chat and chatbots powered customer support software. Kommunicate allows to add live chat on websites as widgets. It provides a codeless integration with Dialogflow, just by enabling the Dialogflow API of the project and creating a service account key. It also offers a wide range of commands for the customization of the widget that will be displayed into your project website.

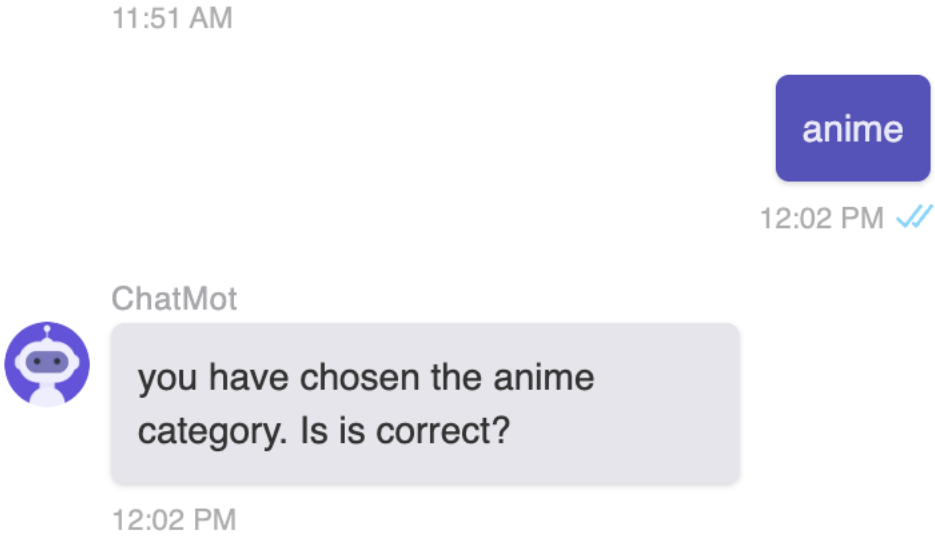
For interacting with the user, I have decided to create three flows, which handle the response of the user, by using two intents: getResponse, to check positive answers, and getNetagiveResponse to catch negative answers.



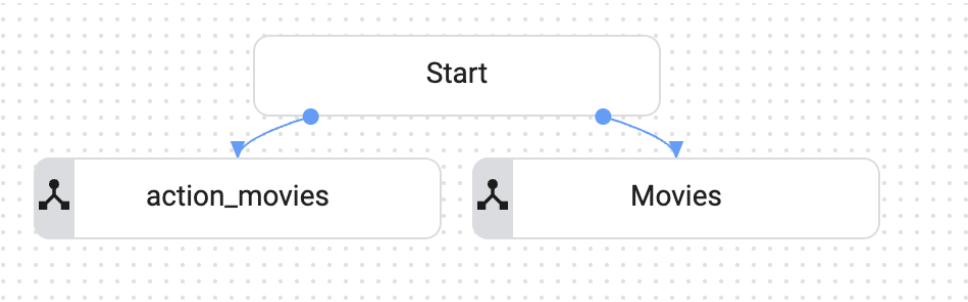
[13] Kommunicate documentation, available at https://docs.kommunicate.io/docs/.



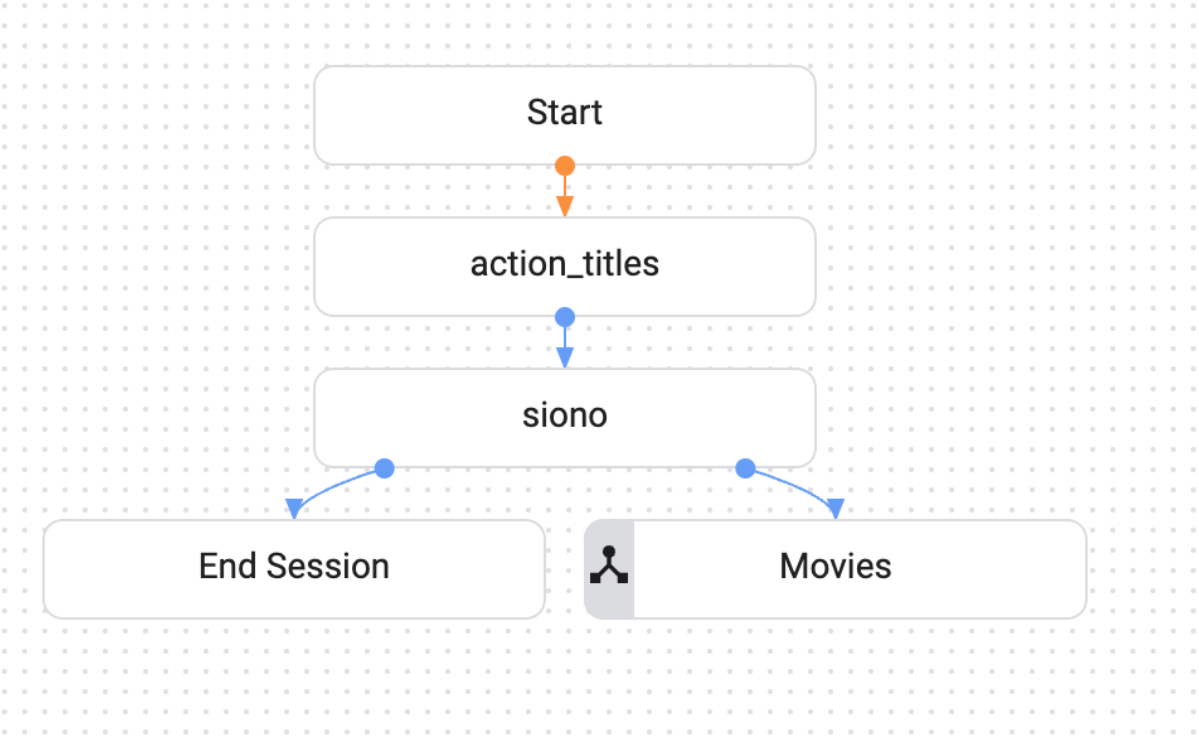
These flows are response_handler_action, response_handler_anime and response_handler_comedies. They oversee asking to the user if the category he/she typed is correct.

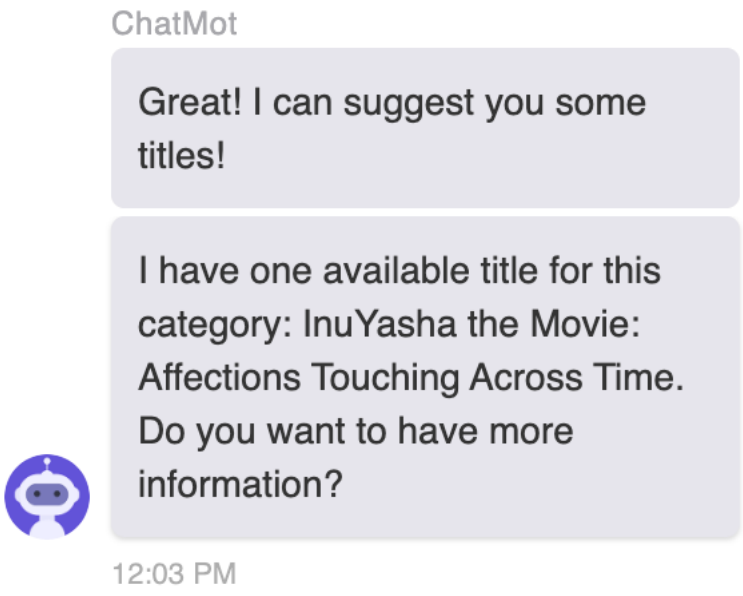


The structure of these flow is the following:

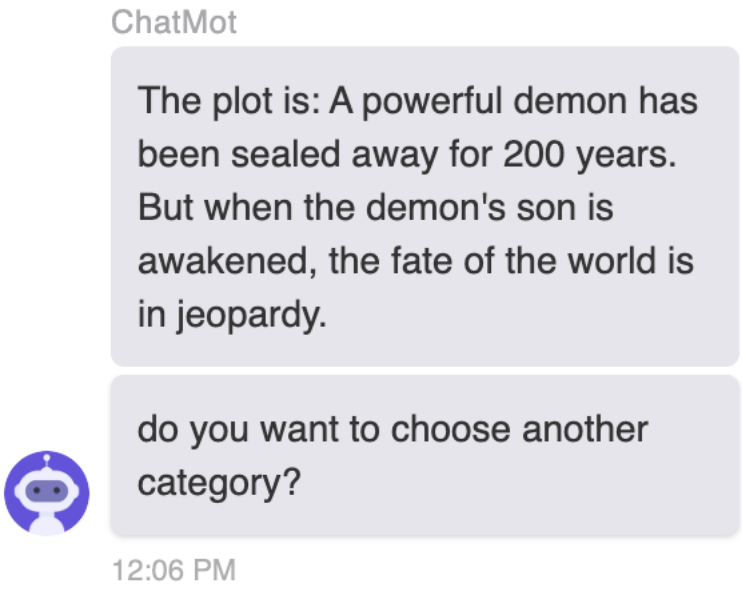


If the answer is positive, the next flow displayed is for example action_movies, otherwise the user has the chance to choose again the category, going back to the Movies flow.

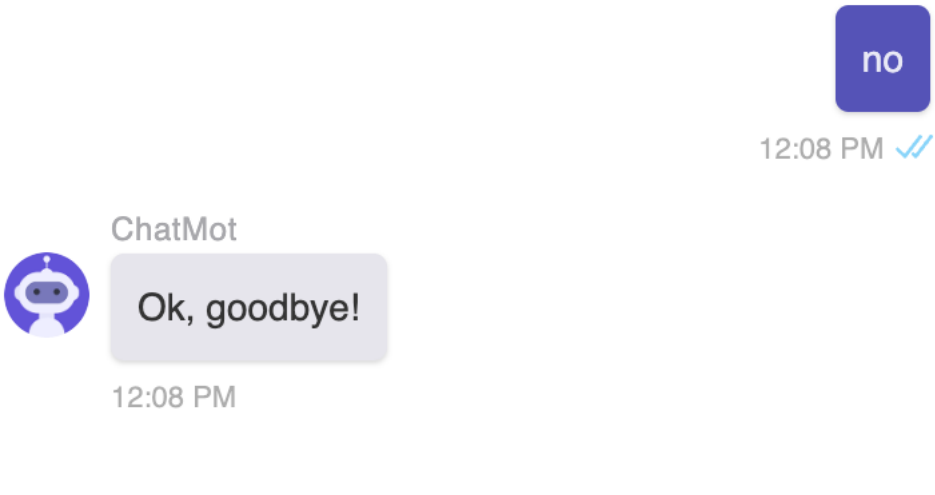




The bot asks If the user wants to receive more information about the movie. Again, an intent catches the answer of the user and if the answer is positive this message is displayed:



If the answer is negative, only the question “do you want to choose another category?” is displayed. In both scenarios, if the answer is yes, we go back to the Movies flow, otherwise a goodbye message is displayed, and the conversation ends.



It is also possible to talk with the chatbot, since a microphone is available to dictate the answer.