Conversational Agents - Chatbots

Authors: Ander Alonso, Aitor Sánchez, Alex de Miguel, Josu Oca
Group: G06
Subject: Advanced Artificial Intelligence Techniques
Teacher: Germán Rigau Claramunt
September 28, 2020

1. Introduction

A conversational agent (CA), or dialogue system, is a computer system that, by using speeches, graphics, haptics or even gestures, is able to converse with human beings.

Since the first speaking dialogue system was issued by DARPA in 1977, Conversational Agents have been incredibly improved. Nowadays, Conversational Agents are divided in two types: Task-oriented CA and non-task oriented CA.

Regarding to Task-oriented conversational agents, these are software components based on artificial intelligence that are able to simulate an intelligent conversation with their users. We often use them in our everyday life as we operate with tools and systems that employ them, say Siri, Interfaces to Cars, Robots, etc.

Respecting to non-task oriented CA, we have chatbots. A chatbot (also known as chat robot, talk bot, chatterbot or chatterbox), is an artificial conversational entity that allows a computer program to communicate with a live human agent by simulating a real conversation. In order to get this, chatbots converse through text or text-to-speech messaging using a natural language, for instance, English.

In the following lines you will be able to find information and increase your knowledge about non-task oriented conversational agents and its functioning.

2. Chatbot architecture

The base of a chatbot is the Natural Learning Understanding(NLU) and the Natural Learning Process(NLP).

The input that the user gives to the bot has to be treated so that the bot understands it. NLU would take care of correcting and restructuring the data to later identify the users intention using NLP.

The NLU divides the input in segments based on linguistic rules creating patterns that later the NLP will have to process. This pattern will change based on the use and the aim that will have the answer that the bot will give.

Once that the NLU creates the patterns that the bot will have to process, the NLP will have to identify the intention to give an answer to the user. If the bot is a task-based bot, will find an answer that helps the user in that task. Whereas if it is a non-task based bot, will only give a general response for continue with the conversation.

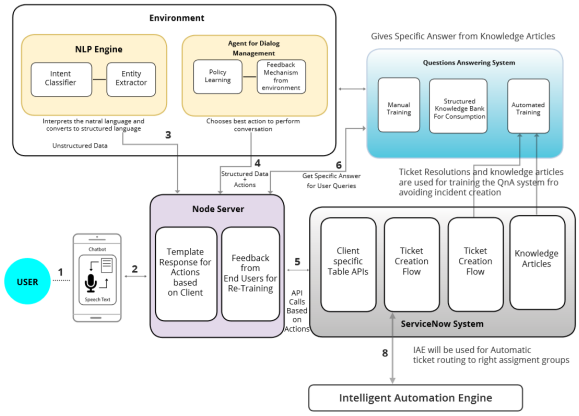

Figure 1: General scheme of chatbot functioning

An non-task oriented conversational AI, or a chatbot, needs a previous training to fill its data-base. That training can be made manually or automatically. Manual training involves do-

main experts creating a list of frequently asked questions and mapping its answers. On the other hand, automated training will require providing the chatbot with information documents or other Q&A type documents to ask the robot to train itself.

3. Non-Task Oriented Chatbots

3.1. Rule-based chatbots

Rule-based chatbots , also named as decision-tree bots, use a database of responses and a set of rules to give an appropriate response. They are able to hold basic conversations based on "if/then" logic and cannot generate their own answers. However, with an extensive database of answers and smartly designed rules, they can respond consistently.

ELIZA simulates a Rogerian psychotherapist and was created by Joseph Weizenbaum to "demonstrate that the communication between man and machine was superficial". Evidently, he did not anticipate the success of the program and many people who interact with ELIZA attributed human feelings to the machine.

The script that powers ELIZA is relatively simple. It assigns a value to each word of the user input sentence and uses the values to reorder the words.

function ELIZA GENERATOR(user *sentence*) **returns** *response*

Find the word *w* in *sentence* that has the highest keyword rank
  if *w* exists
    Choose the highest ranked rule *r* for *w* that matches *sentence*
    response ← Apply the transform in *r* to *sentence*
    if *w* = 'my'
      future ← Apply a transformation from the 'memory' rule list to *sentence*
      Push *future* onto memory stack
  else (no keyword applies)
    either
      response ← Apply the transform for the NONE keyword to *sentence*
    or
      response ← Pop the top response from the memory stack
  return(*response*)

Figure 2: A simplified sketch of the Eliza Algorithm

For example, the sentence 'I want to run away from my parents' attributes low values to pronouns (I) and the highest value to the actual action (run away from my parents). This allows the program to know exactly how to change the sentence to a question.

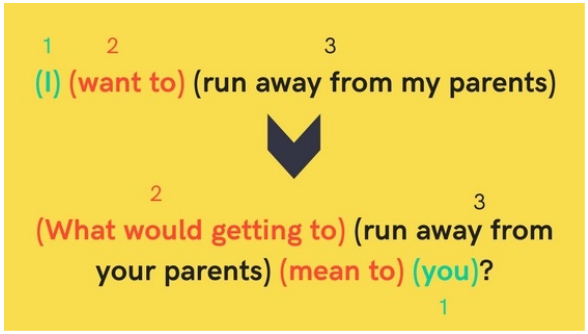

Figure 3: Example of how ELIZA works

ELIZA's framework is still used today; modern chatbot system tools, like ALICE, are based on updated versions of ELIZA's pattern/action architecture.

ALICE stores knowledge about conversation patterns in AIML. AIML (Artificial Intelligence Mark-up Language) is derivative of Extensible Mark-up Language (XML). The AIML system progressively stores information about a person during a conversation. This enables ALICE to create the illusion of knowledge, by responding in conversation in different ways according to a person's age, gender and location.

Categories are the basic unit of knowledge in AIML. Each category is a rule for matching an input and converting it to an output. It consists of a pattern label, which represents the user input, and a template label, which represent the chatbot answer.

```
<category>
    <pattern> Hello ALICE </pattern>
    <template> Hi there! < /template>
</category>
```

Figure 4: AIML example

User: Hello ALICE

Alice: Hi there!

User: How old are you?

Alice: I am eighteen in computer years

User: Who is your father?

Alice: I was programmed by Dr.Richard S. Wallace

User: *so you are a software program?*

ALICE: *Interesting deduction. Thanks and you are a delightful person too. Am I a software program? Let me put it this way, unknown person: the ALICE series is the most reliable computer software ever made. We are for all practical purposes foolproof and incapable of error.*

User: Wonderful! So what's your favorite food ALICE?

Alice: Yeah that's right! I require only electricity

Figure 5: ALICE conversation example

3.2. AI chatbots

AI or corpus-based chatbots that use machine learning understand the context and intent of a question before formulating a response. These chatbots generate their own answers to more complicated questions using natural-language responses. The more you use and train these bots, the more they learn and the better they operate with the user.

These chatbots are divided in two main groups providing their architectures: information retrieval and machine learned sequence transduction. They have similarities with rule-based chatbots due to little modeling of the conversational context they do. Alternatively, they focus on generating a single response turn that is appropriate given by the user's immediately previous comment. Because of this reason, they are also called response generation systems. Corpus-based chatbots are quite similar to question answering systems, which ignore the context when they focus on single responses.

3.2.1. IR-based chatbots

The way these chatbots work is responding to the issues they get by repeating another response from a corpus of natural text. What makes a difference, is how they choose the corpus and how it is decided what it is appropriate information to copy.

For instance, is usual to collect information of human conversation from the social networking sites. What is more, there is also conversational data taken from the chatbots put into practice.

In order to choose the appropriate response, chatbots can use any retrieval algorithm, but the two simplest methods are the following.

Return the response to the most similar turn: The user gives a query q and a conversational corpus C, and the algorithm finds the turn t in C that is most similar to q and return the following turn, that is the human response to t in C.

$$r = response\left(argmax_{t \in \mathcal{C}} \frac{q^T t}{||q||t||}\right)$$

Return the most similar turn: The user gives a query q and a conversational corpus C, and the algorithm returns the turn t in C that is most similar to q.

$$r = argmax_{t \in \mathcal{C}} \frac{q^T t}{||q||t||}$$

3.2.2. Sequence to sequence

In these types of chatbots the system learns from a corpus to transduce a question to an answer.

This idea was developed by using phrase-based machine translation. Nevertheless, task response generation was different from machine translation. In the last one, they tend to align well with each other, however, the user utterance may share no words or phrases with a coherent

response.

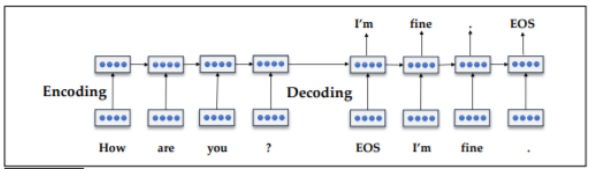

Figure 6: A sequence to sequence model

One of the tendencies of sequence to sequence model is to produce repetitive and predictable responses like "I'm OK". These responses often tend to finish the conversation between the user and the bots. This can be addressed by using adversarial networks, to learn to choose responses that make the conversation more natural.

3.3. Ruled-based Chatbots vs AI Chatbots

Depending on on the purpose, the task, the budget... it will be better to choose one type of chatbot or the other.

3.3.1. Advantages of a rule-based chatbot

While rule-based bots have a less flexible conversational flow, these guard rails are also an advantage. You can better guarantee the experience they will deliver, whereas chatbots that rely on machine learning are a bit less predictable.

- Generally faster to train.
- Integrate easily with legacy systems.
- Are highly accountable and secure.
- Can include interactive elements and media.
- Are not restricted to text interactions.
- Cheaper than IA Chatbots.

3.3.2. Advantages of a AI chatbot

These chatbots, work well for companies that will have a lot of data. Although they take longer to train initially due to their complexity, AI chatbots save a lot of time in the long run.

- Learn from information gathered
- Continuously improve as more data comes in
- Understand patterns of behaviour
- Have a broader range of decision-making skills
- Can understand many languages

4. Frameworks

4.1. Rasa

Rasa is an open source machine learning framework for building chatbots that can be deployed anywhere. Rasa has three main components, Rasa NLU, Rasa Core and Rasa X.

Rasa NLU understands user messages and detects Intent and Entity in the message. We can consider intent as the objective of the user input. For example, the sentence "What's the weather like tomorrow ?" has request_weather intention.

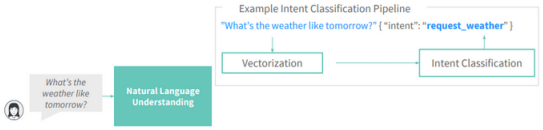

Figure 7: Intent example

At the same time, entity recognize information that helps to understand the sentence.

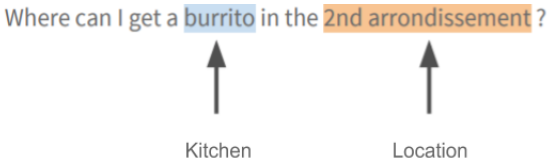

Figure 8: Entity example

In this example, kitchen and location are two extracted entities.

Rasa Core is responsible for holding a contextual conversation and predicts the best answer based on the Rasa NLU, conversation history and training data.

Rasa X is a tool designed to learn from real conversation and improve your assistant. However, using this tool is optional.

When you install rasa in your computer "pip install rasa" and you create a new project "rasa

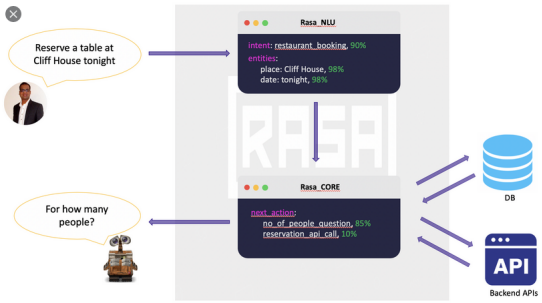

Figure 9: Rasa core example

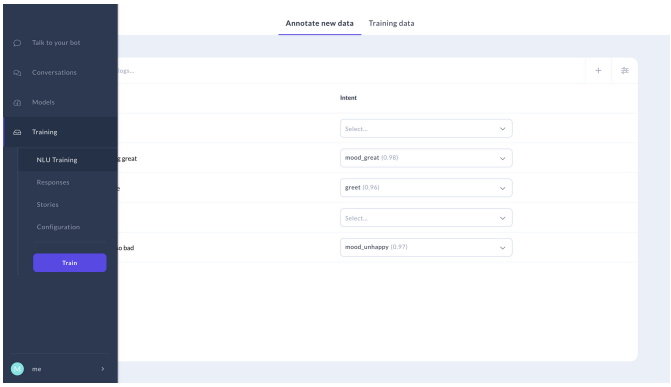

Figure 10: Rasa X

init" the following files will be created:

- actions.py: Code for your custom actions
- config.yml: Configuration of your NLU and Core models
- credentials.yml: Details for connecting to other services
- data/nlu.md: Your NLU training data
- domain.yml: Your assistant's domain
- endpoints.yml: Details for connecting to channels like FB messenger
- models/ıtimestampȩ.tar.gz: Your initial model.

If you want to train using your own custom data, you have to create new intents and entities in data/nlu.md file. On one hand, intents are specified with ##intent:name_of_intent followed by a list of question for the intent;

On the other hand, entities are specified inside each question with [value] (name of entity):

In addition, if you want to train your model, you can just execute "rasa train nlu" command

```
## intent:goodbye
- bye
- goodbye
- see you around
- see you later
- talk to you later

## intent:ask_identity
- who are you
- what is your name
- how should i address you
- may i know your name
- are you a bot
```

Figure 11: Rasa intent example

```
## intent:ask_shop_open
- does the shop open on [monday](weekday)
- does the shop open on [wednesday](weekday)
- does the shop open on [friday](weekday)
```

Figure 12: Rasa entity example

and it will look for NLU training data files in the data folder and save a trained model in the model folder.

Finally, you can test your own model executing "rasa shell nlu" command

4.2. Dialogflow by Google

Dialogflow is a Google-owned developer of human-computer interaction technologies based on natural language conversations. It is used to design and integrate a conversational user interface into mobile apps, web applications, devices, bots, interactive voice response systems, and so on.

Its Standard Edition is totally free and it allows you to build and train your own Chatbot. You just need to sign up on the platform or login with a Google Account and, after doing that, you are able to create different Conversational Agents for different purposes.

Regarding how it works and how it is trained, you need to create different intents to map what the user says and what the agent does by entering examples of what you might expect a user to ask for. For example, if you were creating a weather agent, you would have to include questions about locations and different times. The more examples you provide, the more ways a user can ask a question expecting the agent to understand.

For instance, if you want your agent to understand a question about the weather forecast, you will have to enter this kind of examples:

- What is the weather like today
- What is the weather supposed to be
- Weather forecast
- Weather for tomorrow
- Weather forecast in San Francisco

The words highlighted in the previous examples are the parameters of the indent. In this case "today" and "tomorrow" are date parameters and "San Francisco" is a place parameter.

It is so important to integrate parameters in new indents because they allow Dialogflow to understand exactly what the user is asking (f.e. in the previous indent the agent would be able to distinguish if we want to get information about the local weather forecast or the San Francisco's forecast).

After creating some message examples, you can add different responses so the agent doesn't remain in silence everytime the user asks for something. For instance, if we are creating answers for questions about weather, we could enter this responses:

- Sorry I don't know the weather
- I'm not sure about the weather on $date
- I don't know the weather for $date in $geo-city but I hope it's nice!

In the previous examples, the parameters have the '$' prefix. Due to those parameters, the agent will be able to insert the dates and the places that are mentioned in the request. When the agent answers, it will take into account the amount of parameters mentioned in the user's request in order to choose a correct answer.

After creating different indents, you are able to try out your agent's functioning in the platforms console. You can enter requests that are a little different to the examples you provided in the Training phase.

5. Conclusion

To sum up, we can say that Conversational Agents are becoming very popular lately. The most used ones are the task-oriented agents, because the main objective of these agents is to ease the simple tasks of smartphones or car displays.

These bots are more complicated to train compared to non-task oriented agents, as these others are created just to have conversations with the users.

After seeing all the improvements that the chatbots have had during these last years, we can conclude that they are getting closer and closer to human beings.

With time it will be more difficult to differentiate if we are speaking to a human or to a computer.

References

[1] Task-oriented Conversational Agent Self-learning Based on Sentiment Analysis http://ceur-ws.org/Vol-2244/paper$_0$1.*pdf*

[2] V-soft consulting, https://blog.vsoftconsulting.com/blog/understanding-the-architecture-of-conversational-chatbot

[3] https://www.researchgate.net

[4] https://web.stanford.edu/$\sim$ $jurafsky/slp3/24.pdf$

[5] https://web.stanford.edu/$\sim$ $jurafsky/slp3/26.pdf$

[6] https://www.hubtype.com/blog/rule-based-chatbots-vs-ai-chatbots/

[7] http://www.richardgibson.org/Resources/Alice.pdf

[8] https://towardsdatascience.com/create-chatbot-using-rasa-part-1-67f68e89ddad

[9] https://rasa.com/

[10] https://planetachatbot.com/tutorial-como-construir-asistente-voz-con-herramientas-de-codigo-abierto-rasa-y-mozilla-19af26871

[11] https://www.margo-group.com/en/news/
a-brief-introduction-to-
chatbots-with-dialogflow/