Universidade de Aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Sistemas Inteligentes (2021/22)

Mini Project 1

Group n.º 6

Daniel Gomes, Nmec: 93015
Mário Silva, Nmec: 93430
João Carvalho, Nmec: 89059
Rui Fernandes, Nmec: 92952
Gonçalo Passos, Nmec: 88864

Contents

Introduction

This report aims to describe the final work developed in the context of the Mini Project 1 from the Intelligent Systems Course, in which the main goal is the development of a Conversational Agent (a Chatbot). Thus, in the report, it will be covered the key concepts behind the main library for the development of the chatbot, as well as decisions that were taken over the main tasks that will be performed by the bot, supported by the respective flowcharts that describe the flow of interaction with the users. Finally, a description of the most important aspect of the final implementation provided will also be presented.

Natural Language Processing (*NLP*)

Natural language processing - *NLP* - is a branch of AI concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. However, *NLP* is a generally hard process.

- **Highly ambiguous** - not easy to program disambiguation;
- **Vague** (the principle of minimal effort) - difficult to program the context and a priori knowledge;
- **Universal** (domain independent) - hard to program general knowledge representation

Whether the language is spoken or written, natural language processing uses artificial intelligence to take real-world input, process it, and make sense of it in a way a computer can understand.

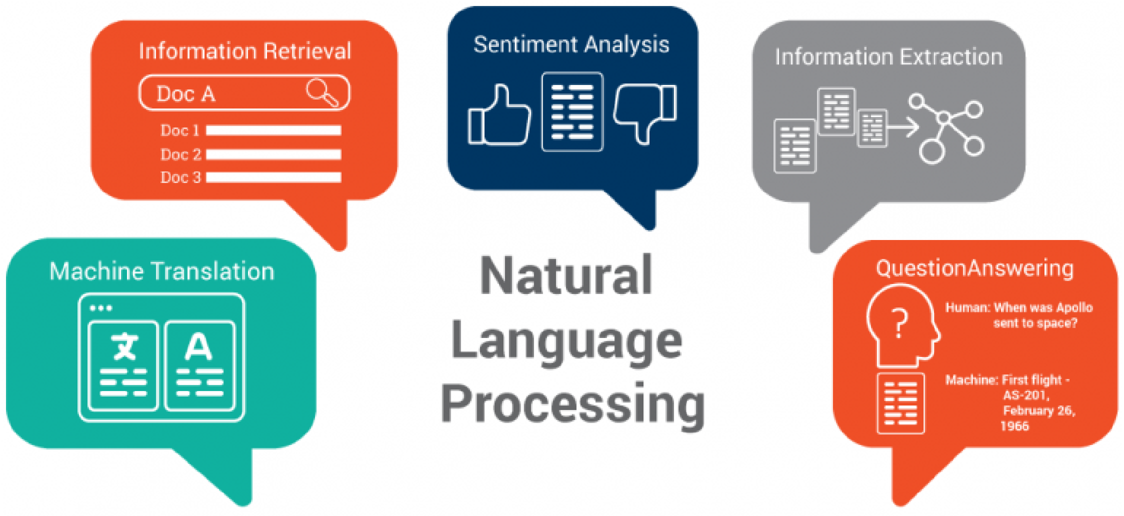


Fig. 1: Some types of Natural Language Processing

Spacy Library

As mentioned previously, the library that we chose to start the implementation of Conversational Agent is **Spacy.** Spacy can be seen as a free and open-source library for *NLP* in Python with a lot of in-built capabilities that has recently become really popular for processing and analyzing data in *NLP*. Besides this, it has been designed specifically for production use, allowing the creation of applications that process and “understand” large volumes of text. Thus, it can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning. It should also be mentioned that Spacy is not an “out-of-the-box” chatbot engine, which means that, for this assignment, we will only take use of the text processing capabilities provided.

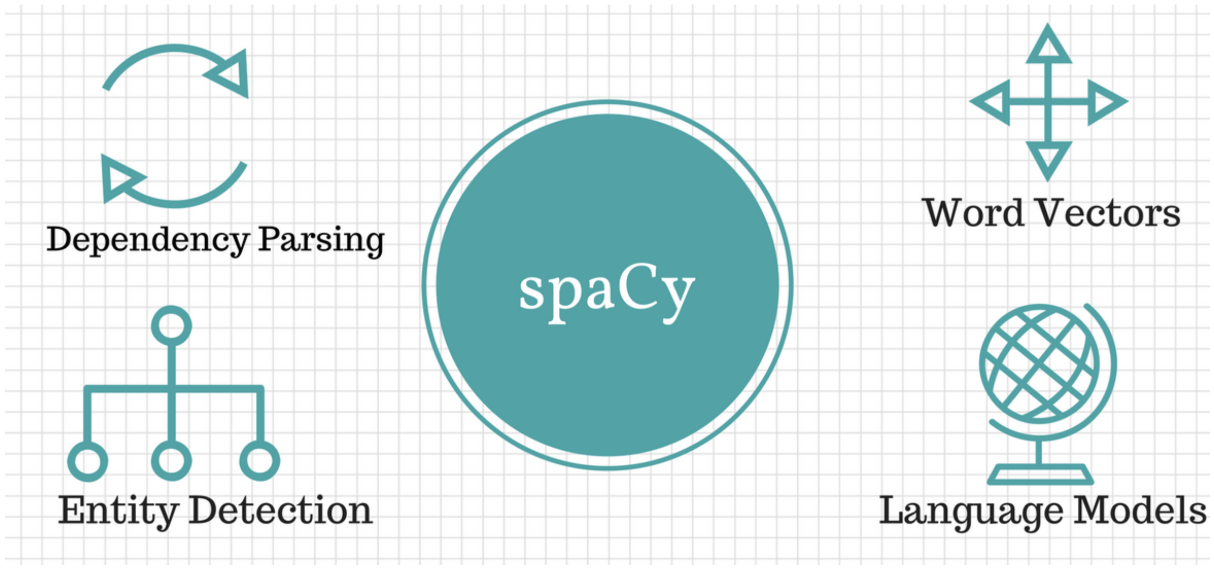


Fig. 2: Spacy Functionalities

How Spacy works?

Regarding the implementation over Spacy, there are three central data structures: the ***Language*** class, the ***Vocab*** and the ***Doc*** object.

The Language class may be seen, from a high level approach, as the core component responsible for processing the Text and turning them into *Docs* objects. These objects are a sequence of tokens and all their annotations. On the other hand, the Vocab is responsible for the storage of data in a vocabulary, which will then be shared by multiple documents. By centralizing strings, word vectors and lexical attributes in the Vocab, we avoid storing multiple copies of this data, which allows memory saving.

To better visualize this, the figure below represents, in a simple way, the workflow in Spacy. The *nlp* method from Spacy, will first tokenize a text producing a Doc, which then will be processed in multiple steps on what Spacy calls a "Processing pipeline", which usually contains a *Tagger*, *Lemmatizer*, *Dependency* Parser and an Entity Recognizer. Each pipeline component returns the processed *Doc*, which is then passed on to the next component.

However, the components of the pipeline may be customized, which can be helpful to suit the interests of the Library Users.

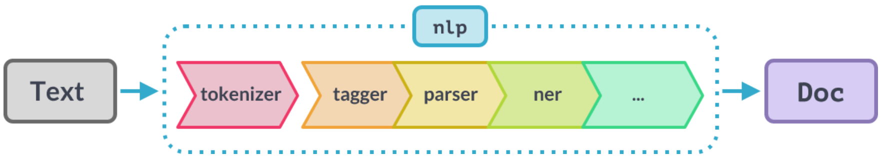


Fig. 3: Spacy Processing Pipeline

Taking into account what has now been mentioned, regarding the components of Spacy, its duties and their interactions between each other, the complete architecture may now be comprehended. Thus, the following figure represents the Spacy Architecture.

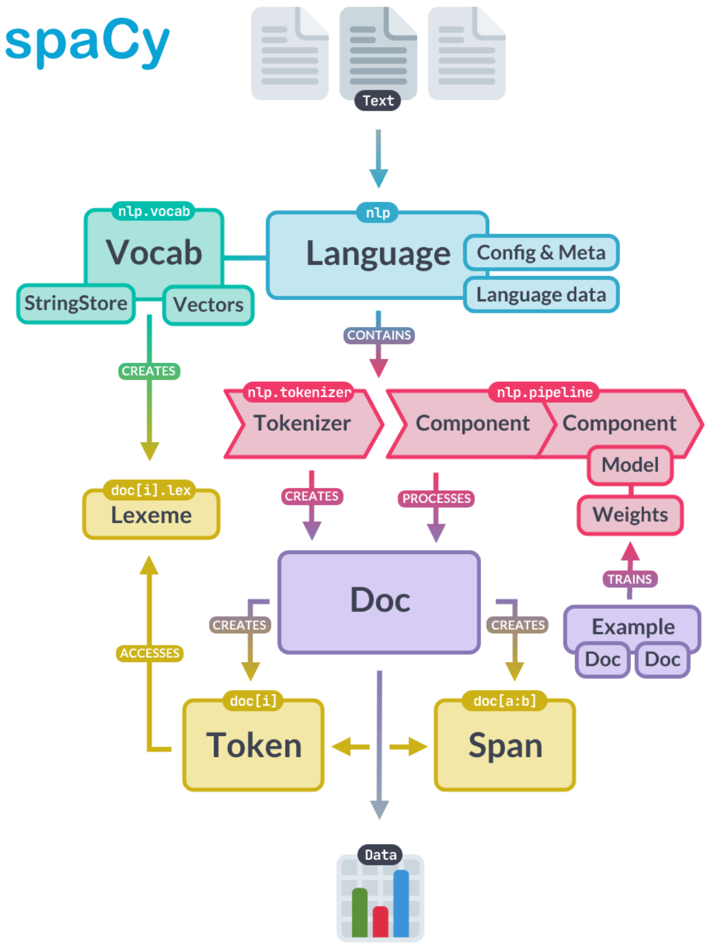


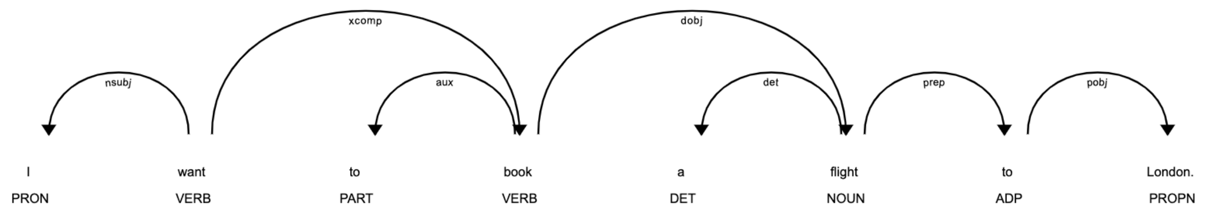
Fig. 4: Spacy Architecture

Text annotations are also designed to allow a single source of truth: the *Doc* object owns the data, and *Span* and *Token* are views that point into it. As mentioned, the *Doc* object is the result of the *Tokenizer* actions, and it's then modified/processed in place by the components of the pipeline. The Language object coordinates these components, since it takes raw text and sends it through the pipeline, returning an annotated document.

Spacy's Syntactic Structure Analysis of a Sentence

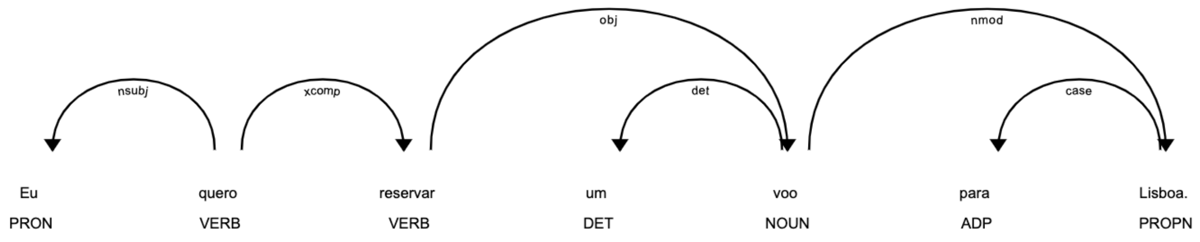
Spacy provides pre-trained models that contain information about sentence syntaxes and tokenization mechanisms.

Here's an example of the syntactic structure analysis with the English model:



As it can be seen above, it correctly executes the **Part of speech tagging** or **POS tagging**, which is the process of marking a word in the text to a particular part of speech based on both its context and definition. In simple language, we can say that POS tagging is the process of identifying a word as nouns, pronouns, verbs, adjectives, etc. "London" is identified as a proper noun, a noun that designates a particular being or thing, usually capitalized.

On the figure below, there's a similar example, but this time with a Portuguese pre-trained model.



Chatbot Design

General Concept

As a base, our chatbot is an AI system that users can seemingly communicate with, intended to be a friendly experience akin to talking to someone else. It is an informative bot, so it mainly exposes information that the user needs upon its inquiries.

In terms of thematic, we decided to develop a chatbot centered on flight travel, so an assistant that can provide information about traveling by plane, detailed aspects of possible destinations and available flights from one place to another.

Interaction Diagram

The simple diagram below divides our system in blocks and shows how they interact with each other.

On one side is the user that will directly interact with the chatbot through its interface. The chatbot in turn resorts to external API's for some up-to-date information and is built upon the already explained spacy library, which is reliant on various pre-trained models for ease of use.

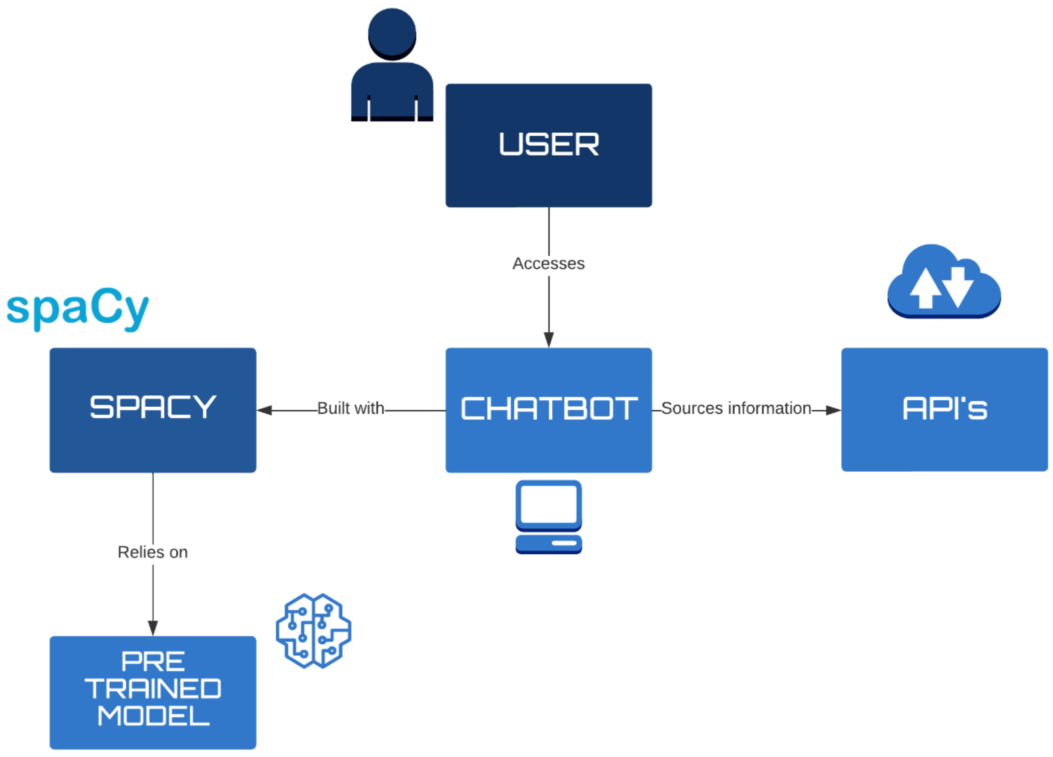


Fig. 5: Interaction Diagram of our Solution

Flow of Communication

To better understand how we intend our chatbot interaction to happen with the user, we will analyze the following flowcharts that represent three main states the chatbot stays in.

Firstly, the bot starts with a generic greeting state to accommodate the user, introducing itself and asking the user to do the same. Upon receiving the user's name, the bot will store and call back on it to provide a friendlier experience throughout the interaction.

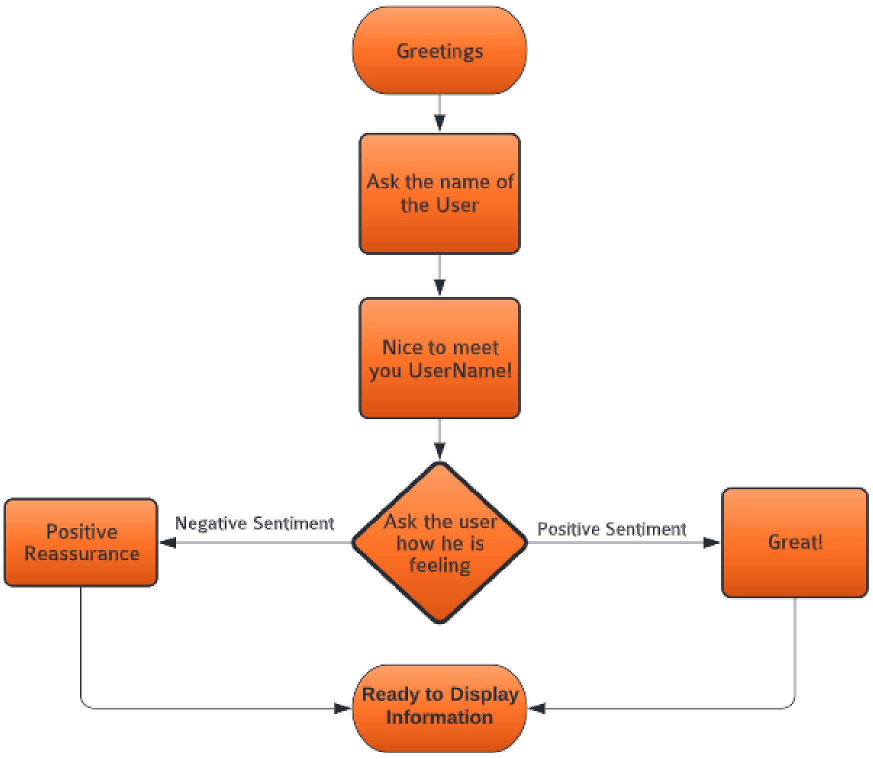


Fig. 6: FlowChart - Greetings State

Furthermore, the bot attempts to perceive how the user is feeling, changing the way it addresses the user depending on the perceived sentiment, guiding them to the core state of the bot where it serves as a travel assistant.

In the informative state, the bot starts by listing the functionalities it has in a digestive way and asks the user what it can help with.

Faced with the functionalities, the user can ask anything and if the phrase is assimilated to a recognized capability, it will follow from there.

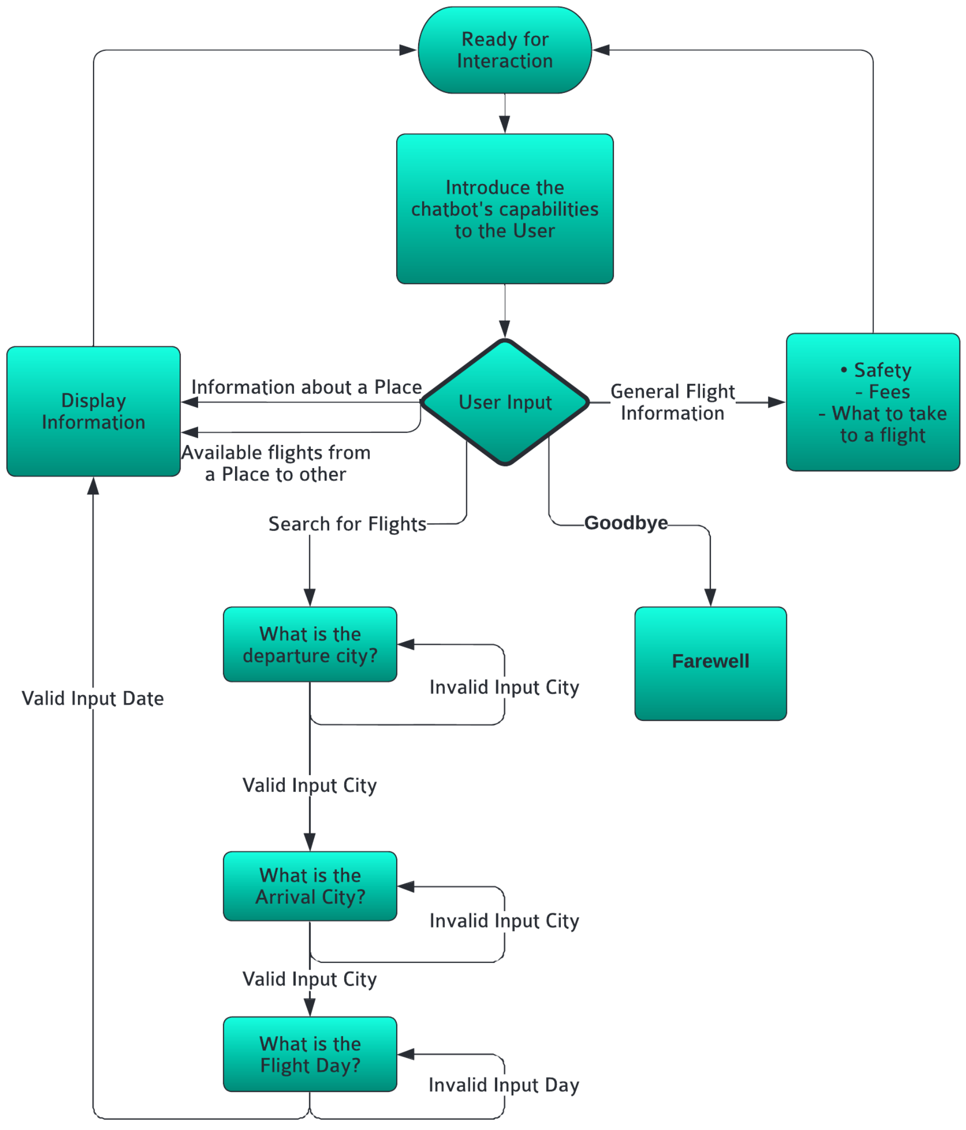


Fig. 7: FlowChart - Interaction State

One option is to ask for information about flying, to which the bot can provide information about the safety of flying, the fees associated with traveling besides the flight itself, passports for example, what a person can take on a plane as well as the kinds of luggage.

One can also inquire about a certain destination to know more about it, the bot is intended to supply general information about a city or country and what one can do there so the user can refine their decision.

Finally a user can find flights to book, doing so in two different ways, the first, in a very straightforward manner, by asking for flights from a city to another on a certain day, Lisbon to Paris tomorrow, for example, and the bot should provide all available flights if they exist. In a more general tone, the user can say they want to travel and the bot shall ask from where and upon input of a valid city (cities without airports shouldn't be recognized) it will proceed to ask for a destination and maybe give suggestions.

After the user finishes using the chatbot to obtain the information they need, they should end their conversation with a farewell, to which the bot will enter a final state.

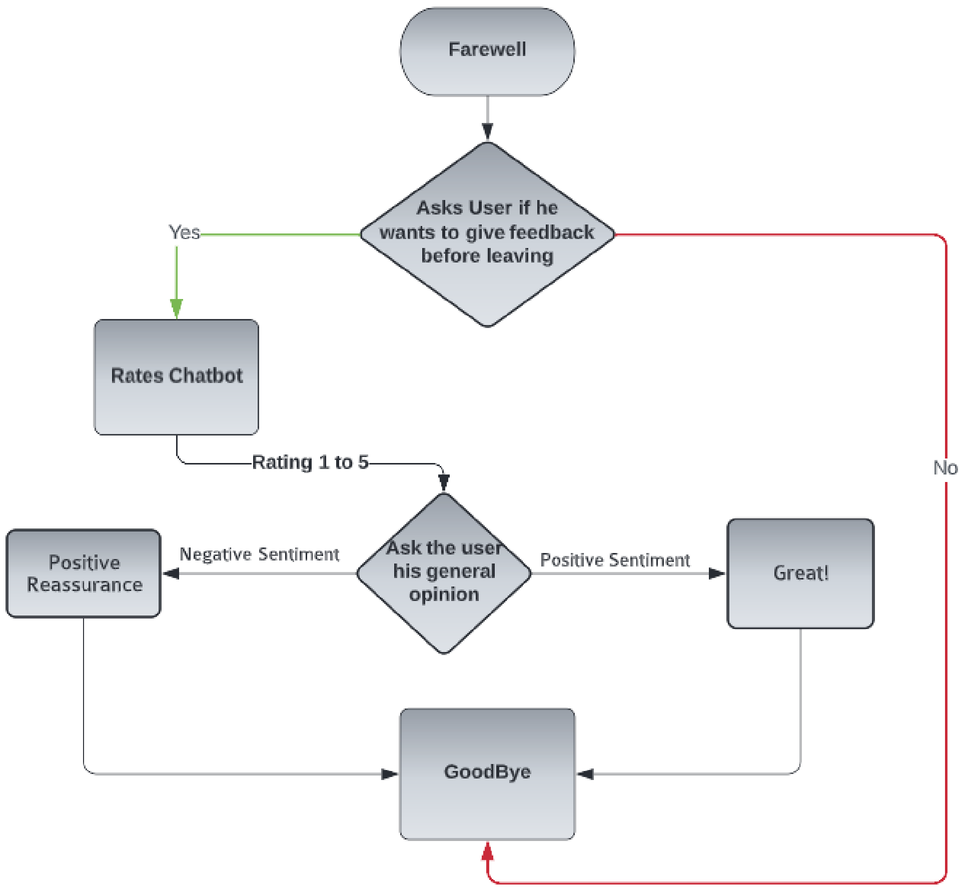


Fig. 8: FlowChart - Farewell State

This state serves as feedback collection, a useful tool for the production side. It's quite a simple interaction, the bot will ask the user if they can leave some feedback, and if not it will simply give its goodbyes and shut off. On the contrary, if the user accepts, it will first ask for a rating from 1 to 5, that's useful for statistics, and then request a general evaluation or opinion of their interaction with the chatbot. Depending on the sentiment behind the answer, the chatbot can thank the feedback in an apologetic or grateful tone before saying it's goodbyes.

Implementation

Chatbot Logic

The way the bot understands the user input is by having a defined set of sentences that best describe an intent, and then, by using a function provided by Spacy, calculate the intent that has the highest similarity value with the user's input.

For example, we defined a set of phrases for the intent "greet", such as: "Hello!", "Hi", "Hey", "Good morning!", etc. If the user then writes something similar to these sentences, it will be identified that the user intends to greet the bot.

For each intent, there is also a defined set of answers the bot can give, while some require additional processing, others are pretty straight forward and the bot can simply return an answer from the answers set.

For example, if the user wants to know more about flight safety, the bot can just return one of the answers from the set that we defined for that specific intent.

But if the user wants to know about a specific flight, the bot must then be provided with a destination, arrival city and the day of the flight, so it knows which flights are available. In order to obtain these, we will use the knowledge base data of Spacy to perform a syntactic and semantic analysis of the user input, with this, it is possible to recognize entities and their associated labels, in the case of a city, it is identified as a Geopolitical Entity.

External API's

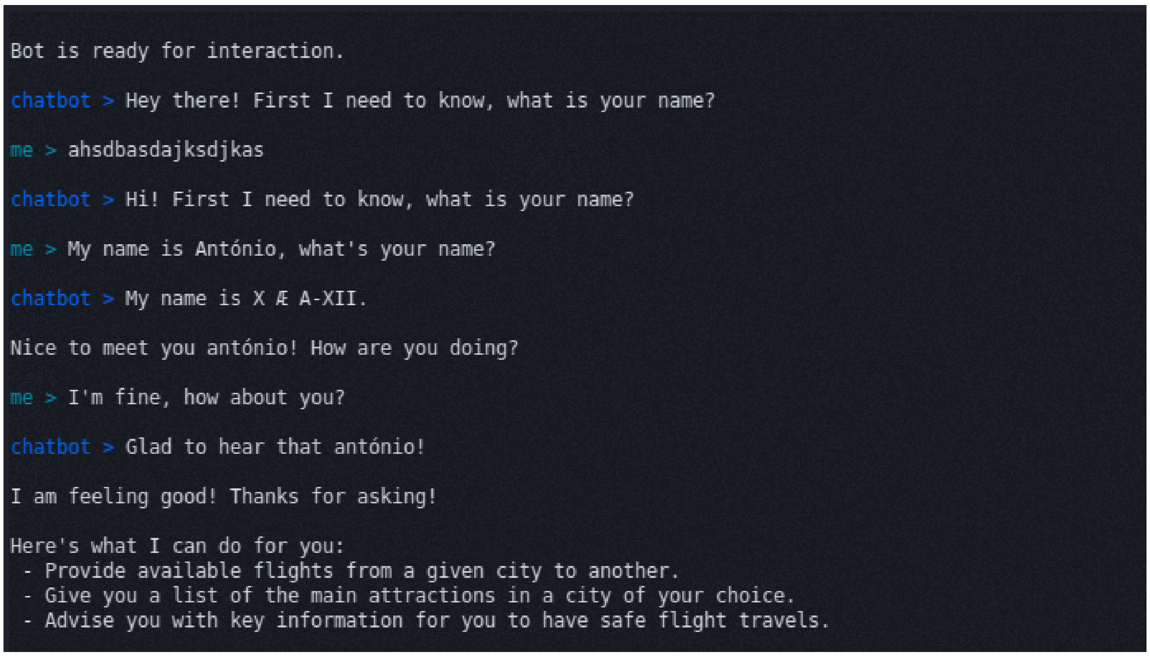
For some of the information the chatbot can provide the user, we gather it from external api's that fit our interest.

In the case of displaying information about a certain city, the chatbot accesses the *OpenTripMap API.* While the API provided the information we wanted, it didn't do so with a single call to an endpoint and so a specific, yet simple class was made to handle this interaction. When given a place name it will first find the coordinates of the given place, then make another call to find attractions in it, and finally make calls for each attraction for their details.

Regarding the flights search, we used a dataset that contains the names and IATA codes of the most famous airports in the word by city. When the user chooses the parameters for the flight(departure and arrival cities and flight day), the chatbot calls the *Amadeus Air API* with all combinations of each departure and arrival airport(using IATA codes) of the given cities and, finally, it will return and display the available flights to the user.

Obtained Results

Regarding the defined goals for this project, we believe that we are able to implement with success all the proposed workflows for the Chatbot. The following figure shows an example of the **Greetings** Interaction with the Chatbot.

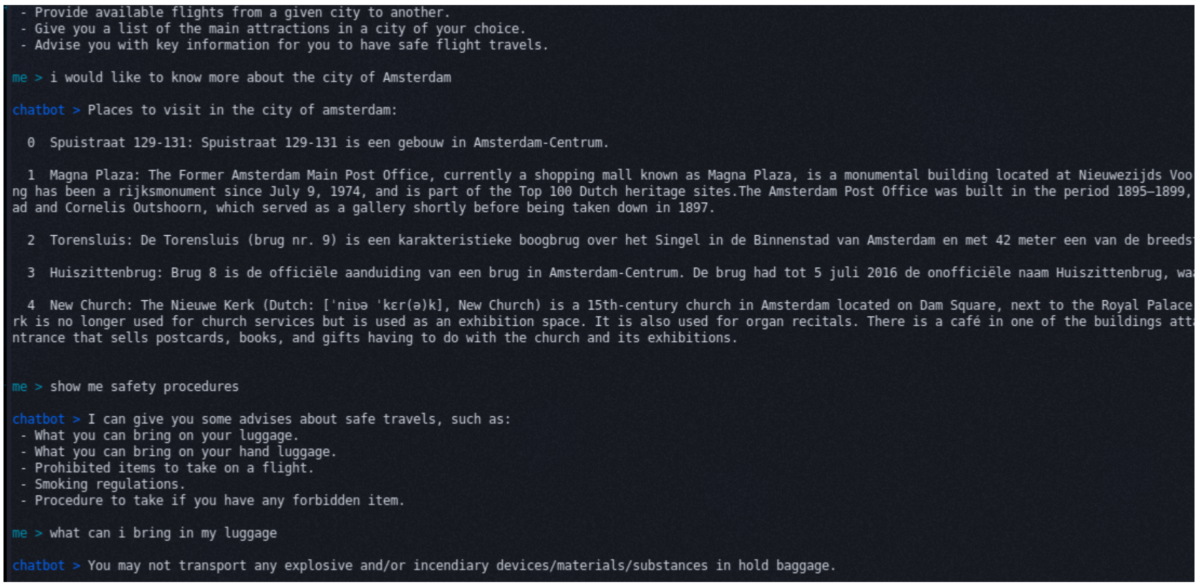


```
Bot is ready for interaction.

chatbot > Hey there! First I need to know, what is your name?

me > ahsdbasdajksdjkas

chatbot > Hi! First I need to know, what is your name?

me > My name is António, what's your name?

chatbot > My name is X Æ A-XII.

Nice to meet you antónio! How are you doing?

me > I'm fine, how about you?

chatbot > Glad to hear that antónio!

I am feeling good! Thanks for asking!

Here's what I can do for you:
 - Provide available flights from a given city to another.
 - Give you a list of the main attractions in a city of your choice.
 - Advise you with key information for you to have safe flight travels.
```

Fig. 9: Bot Interaction example - Greetings

As we may see from the Figure above, initially the Bot will ask for the User's name. If the Bot identifies the name as not valid it will ask again the same question, until the user provides a valid one, as for example, *António*. The bot will then try to recognize the user's current mood by asking how the user is feeling. Finally, the bot shows all the tasks that it is able to do. These tasks, as mentioned above, are providing available flights from a departure city to a arrival city, list the main attractions of a given city, and show the user key information for a safe flight.

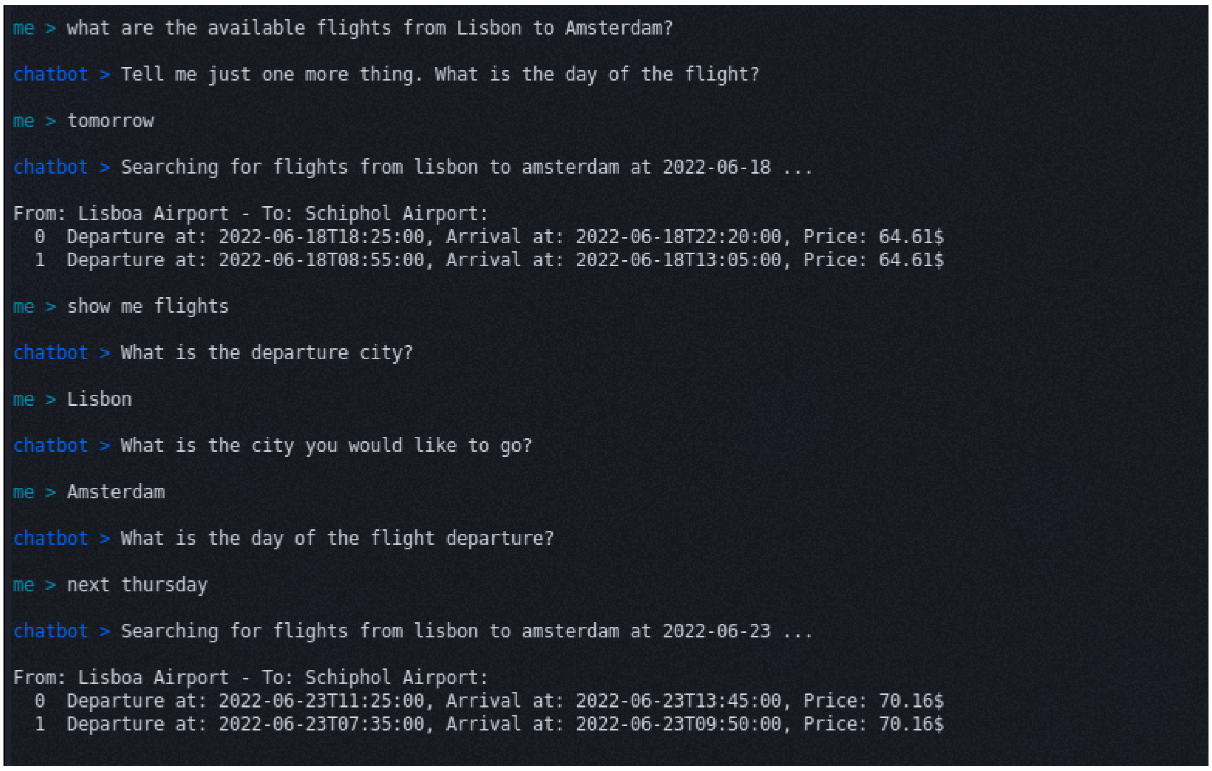
The next figure represents the continuation of the interaction with the Bot, where the users asks information of the City of Amsterdam and nextly, asks for safety procedures when flying. This last one will show to the user a menu of options about safe traveling as it may be seen.



```
 - Provide available flights from a given city to another.
 - Give you a list of the main attractions in a city of your choice.
 - Advise you with key information for you to have safe flight travels.

me > i would like to know more about the city of Amsterdam

chatbot > Places to visit in the city of amsterdam:

  0  Spuistraat 129-131: Spuistraat 129-131 is een gebouw in Amsterdam-Centrum.

  1  Magna Plaza: The Former Amsterdam Main Post Office, currently a shopping mall known as Magna Plaza, is a monumental building located at Nieuwezijds Voo
ng has been a rijksmonument since July 9, 1974, and is part of the Top 100 Dutch heritage sites.The Amsterdam Post Office was built in the period 1895–1899,
ad and Cornelis Outshoorn, which served as a gallery shortly before being taken down in 1897.

  2  Torensluis: De Torensluis (brug nr. 9) is een karakteristieke boogbrug over het Singel in de Binnenstad van Amsterdam en met 42 meter een van de breeds

  3  Huiszittenbrug: Brug 8 is de officiële aanduiding van een brug in Amsterdam-Centrum. De brug had tot 5 juli 2016 de onofficiële naam Huiszittenbrug, wa

  4  New Church: The Nieuwe Kerk (Dutch: ['niʋə 'kɛr(ə)k], New Church) is a 15th-century church in Amsterdam located on Dam Square, next to the Royal Palace
rk is no longer used for church services but is used as an exhibition space. It is also used for organ recitals. There is a café in one of the buildings att
ntrance that sells postcards, books, and gifts having to do with the church and its exhibitions.


me > show me safety procedures

chatbot > I can give you some advises about safe travels, such as:
 - What you can bring on your luggage.
 - What you can bring on your hand luggage.
 - Prohibited items to take on a flight.
 - Smoking regulations.
 - Procedure to take if you have any forbidden item.

me > what can i bring in my luggage

chatbot > You may not transport any explosive and/or incendiary devices/materials/substances in hold baggage.
```

Fig. 10: Bot Interaction example - City and Flight Security Informations

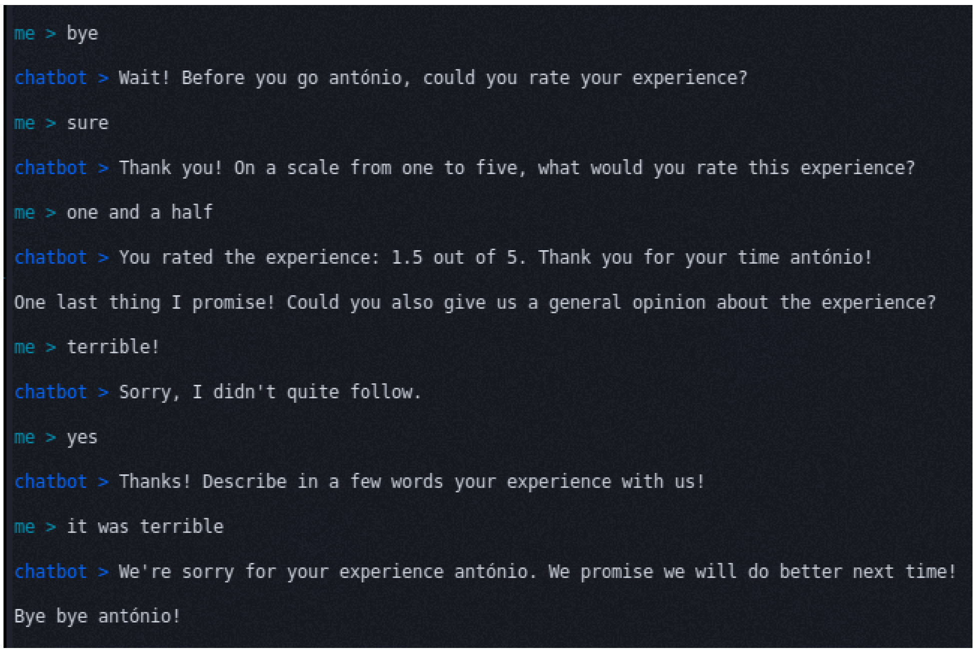
As it can be seen on the next figure, when asking for flights, the user may provide directly the departure, arrival city and the day of the flight, or let the bot ask the user each of these pieces of information necessary to show available flights. Regarding the day of the flight, the bot is able to understand and parse dates in different formats: for instance we may provide a date in number format, referring to the next day or weekdays (“*tomorrow*”, “*next Thursday*”, for example).



```
me > what are the available flights from Lisbon to Amsterdam?

chatbot > Tell me just one more thing. What is the day of the flight?

me > tomorrow

chatbot > Searching for flights from lisbon to amsterdam at 2022-06-18 ...

From: Lisboa Airport - To: Schiphol Airport:
  0  Departure at: 2022-06-18T18:25:00, Arrival at: 2022-06-18T22:20:00, Price: 64.61$
  1  Departure at: 2022-06-18T08:55:00, Arrival at: 2022-06-18T13:05:00, Price: 64.61$

me > show me flights

chatbot > What is the departure city?

me > Lisbon

chatbot > What is the city you would like to go?

me > Amsterdam

chatbot > What is the day of the flight departure?

me > next thursday

chatbot > Searching for flights from lisbon to amsterdam at 2022-06-23 ...

From: Lisboa Airport - To: Schiphol Airport:
  0  Departure at: 2022-06-23T11:25:00, Arrival at: 2022-06-23T13:45:00, Price: 70.16$
  1  Departure at: 2022-06-23T07:35:00, Arrival at: 2022-06-23T09:50:00, Price: 70.16$
```

Fig. 11: Bot Interaction example - Search for Flights

Concerning the **Farewell** workflow, when the user says “goodbye” related expressions, the Bot will try to collect information about the user experience when interacting with it. Thus, if the user agrees on this procedure, the bot will ask the user to rate it on a scale of one to five. The rating value can be provided in number or written in full: in this example the number used was “*one and a half*” which was recognized successfully by the bot as 1.5. Finally, in addition to the rating the bot asks the user for feedback. On the figure below it can be seen that the Bot was able to understand the user feedback as negative and apologizes the user for the experience.



```
me > bye

chatbot > Wait! Before you go antónio, could you rate your experience?

me > sure

chatbot > Thank you! On a scale from one to five, what would you rate this experience?

me > one and a half

chatbot > You rated the experience: 1.5 out of 5. Thank you for your time antónio!

One last thing I promise! Could you also give us a general opinion about the experience?

me > terrible!

chatbot > Sorry, I didn't quite follow.

me > yes

chatbot > Thanks! Describe in a few words your experience with us!

me > it was terrible

chatbot > We're sorry for your experience antónio. We promise we will do better next time!

Bye bye antónio!
```

Fig. 12: Bot Interaction example - Farewell

Conclusion

With this assignment, we managed to have a better understanding of technologies and strategies that may be used in order to implement a Conversation Agent, such as Spacy and NLP, respectively. Despites the challenges that we faced when developing the bot, it was interesting to see the final result as the Bot is completely functional and robust for its main purposes. Finally, it should be mentioned that we believe that we managed to achieve all goals of this assignment taking into account what was proposed to do.