FACULTY OF ENGINEERING OF UNIVERSITY OF PORTO

Interactive bot to support the use of the UPTEC intranet

João Fernando Oliveira e Silva

Master in Informatics and Computing Engineering

Supervisor: Henrique Lopes Cardoso

June 10, 2018

© João Oliveira e Silva, 2018

Interactive bot to support the use of the UPTEC intranet

João Fernando Oliveira e Silva

Master in Informatics and Computing Engineering

June 10, 2018

Abstract

For the last 50 years, chat-bots have been developed. In 1950, Alan Turing predicted that around the year of 2000, with the evolution of the computation, machines would be able to trick more than 30% of the people into thinking that the machine was actually a human. In other words, talking with a machine or with a human would be indistinguishable for about a third of the world population.

ELIZA, released in 1966, was the first chat-bot. User input is manipulated using a rule-based system converting it into a question. From that moment, thousands of chat-bots were built. From personal assistants to help-desks, online shopping and others. Chat-bots are present in our daily life and are here to stay.

UPTEC is an incubator of startups, innovation centers, and anchor projects. With the growth of the incubator, the management that was being done through email and Excel files became insanely hard so a management platform became mandatory, this intranet platform is now called UPTEConnect. The platform allows the management of all companies, collaborators, requests for access cards, assistance and more.

UPTEConnect was not released yet and it is expected that once released, users of the platform will have some difficulties finding the tools they need, as well as to understand all the work-flows associated with every action that can be executed in the platform.

Further, with the continuous development of the platform, the number of features continues to increase. This results in a complex platform, challenging to be used by new users.

To solve this problem, an artificial intelligence chat-bot was developed. The chat-bot is able to recognize user inputs, that is, it will have natural language recognition capabilities and guide the user through the platform accordingly. The goal is that a user can easily be lead to the desired tools on the platform.

There are two possible main desires that a user might have, one that is to go straight to the tool, the other is to learn how to get to the tool. Depending on the desire, the chat-bot will create an interactive tutorial through the platform or redirect the user to the tool.

With more than 15 possible user intents, the result is an artificial intelligence chat-bot that is able to help and guide the users of the platform. A secure system (only information that the user is allowed to see is returned) that adapts the responses to the user. On top of that, all the tools that are used by this project are completely free, resulting in low hosting costs for the incubator.

Resumo

Durante os últimos 50 anos, chat-bots tem vindo a ser desenvolvidos. Em 1950, Alan Turin previou que por volta do ano 2000, com a evolução da computação, os computadores seriam capazes de enganar mais de 30% da população, fazendo-as crer de que o computador era na verdade um humano. Por outras palavras, um computador ou humano tornar-se-iam indistinguíveis para cerca de um terço da população.

Lançado em 1966, ELIZA foi o primeiro chat-bot. O texto introduzido pelos utilizadores era manipulado através de um sistema baseado em regras, de tal maneira que o resultado final era uma pergunta com as características do texto introduzido. Desse momento, até hoje, milhares de outros chat-bots foram desenvolvidos. Desde assistentes pessoais, *help-desks*, compras online e outros. Os chat-bots estão presentes no nosso dia-a-dia e estão aqui para ficar.

A UPTEC é uma incubadora de startups, centros de inovação e projetos âncoras. Com o crescimento da incubadora, a gestão que estava a ser efetuada via email e ficheiros Excel tornou-se muito complicada. Tal facto levou à necessidade do desenvolvimento de uma plataforma de gestão, plataforma a qual tem agora o nome de UPTEConnect. A plataforma permite a gestão de todas as empresas, colaboradores, pedidos de cartões de acesso, assistência, entre outros.

UPTEConnect ainda não foi lançada e como esperado, quando for lançada, os utilizadores da plataforma terão algumas dificuldades em encontrarem as ferramentas que precisam, assim como todos os *work-flows* associados a cada ação possível de ser executada na plataforma.

Além disso, com o desenvolvimento contínuo de novas funcionalidades, a dificuldade de uso por parte de novos utilizadores tornar-se-há ainda maior.

Para resolver este problema, um chat-bot de inteligência artificial foi desenvolvido. O chat-bot é capaz de reconhecer as intenções de um utilizador, isto é, tem a capacidade de reconhecimento de linguagem natural, e irá guiar o utilizador de acordo com a esta intenção. O objetivo é a de que um utilizador possa ser guiado de forma fácil às ferramentas na plataforma.

Existem duas possíveis intenções principais por parte de um utilizador, uma de ir diretamente à ferramenta, outra que é a de aprender a como chegar a essa ferramenta. Dependendo da intenção, o chat-bot irá criar um tutorial interativo pela plataforma ou redirecioná-lo diretamente para a ferramenta.

Com mais de 15 possíveis intenções de utilizador, o resultado é um chat-bot de inteligência artifical, capaz de ajudar e guiar os utilizadores pela plataforma. Um sistema seguro (apenas informação que o utilizador tem acesso é mostrada), que adapta as respostas ao utilizador. Além disso, todas as ferramentas usadas pelo projeto são gratuitas, o que resulta em baixos custos de alojamento para a incubadora.

Acknowledgements

I would like to express my very great appreciation to Professor Henrique Cardoso and to the development manager at UPTEC Cláudia Ribeiro, my research supervisors and to my Professor Maria Ribeiro.

I would like to offer my special thanks to Professor Joni Dambre and Professor Tom Dhaene, who taught me the concepts of Machine Learning in Gent, these allowed me to understand the platforms used in the development of the chat-bot.

One special thank you to my friend, Erasmus travel buddy and English mentor, Luís Ramos Pinto de Figueiredo and to all my friends at *Faculdade de Engenharia da Universidade do Porto* who wrote their dissertations alongside myself.

Finally, I wish to thank my parents, brother, and girlfriend and all my family for their support and encouragement throughout my study.

João Oliveira e Silva

“Shoot for the moon.
Even if you miss, you’ll land among the stars“

Oscar Wilde

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 3
- 1.3 Goals . . . 3
- 1.4 Structure . . . 4

2 Chat-Bots **5**
- 2.1 Architecture . . . 6
  - 2.1.1 Models . . . 7
  - 2.1.2 Domain . . . 7
  - 2.1.3 Conversation . . . 8
  - 2.1.4 Design Techniques . . . 8
- 2.2 Matching the Intent . . . 9
- 2.3 Natural Language Platforms . . . 9
  - 2.3.1 Lex . . . 10
  - 2.3.2 DialogFlow . . . 10
  - 2.3.3 WIT.ai . . . 11
  - 2.3.4 LUIS . . . 12
  - 2.3.5 Comparison Table . . . 12
  - 2.3.6 Chosen Platform . . . 13

3 Evolution of Chat-Bots **15**
- 3.1 Turing Test . . . 17
- 3.2 Generation Models . . . 17
- 3.3 Methodology . . . 18
- 3.4 Matching the Intents . . . 19
- 3.5 Security Issues . . . 20

4 Approach **21**
- 4.1 UPTEConnect . . . 21
- 4.2 Goals . . . 22
- 4.3 Use Cases . . . 23
- 4.4 Platform . . . 25
- 4.5 Data Collection . . . 26
- 4.6 Technical Specifications . . . 27

5 Development **29**
5.1 Implementation 29
5.1.1 DialogFlow 29
5.1.2 Chat-Bot 33
5.1.3 User Interface 35
5.2 Challenges 39
5.3 Architecture 41

6 Experiments and Results **43**
6.1 Evaluation Environment 43
6.2 Inquiry 44
6.3 Results 44

7 Conclusions and Future Work **49**

References **53**

A Goostman Bot Conversation **57**

B Inquiry **59**

List of Figures

2.1 Common flow that a chat-bot uses to predict the intent given the user input 6
2.2 Matching the intent flow 10
2.3 DialogFlow user interface for creating new intents 11

4.1 Home page of the UPTEC intranet, UPTEConnect 22
4.2 Diagram of the interaction flow between a user and the chat-bot 25
4.3 Diagram of the developed platform 26

5.1 Training an intent in DialogFlow 31
5.2 Predicting the intent without having an exact match in the training examples . . . 32
5.3 Chat-bot requesting the user to input the collaborator name. 32
5.4 An output context in DialogFlow with contexts with lifespan of 2. 33
5.5 An input context in DialogFlow 33
5.6 Welcome tutorial to UPTEConnect 36
5.7 Welcome tutorial showing where to change the platform language 36
5.8 The Angular 4, Ng-Chat message component appearance 37
5.9 Welcome message given by the chat-bot when the dialog window is opened . . . 37
5.10 A tutorial step on how to edit a collaborator 38
5.11 Another tutorial step on how to edit a collaborator 38
5.12 Request the user to redefine the search 39
5.13 UML architecture diagram of the platform 41
5.14 UML architecture diagram of the chat-bot module 42

6.1 Results for the question number 1 of the inquiry. 44
6.2 Results for the question number 2 of the inquiry. 45
6.3 Results for the question number 3 of the inquiry. 45
6.4 Results for the question number 4 of the inquiry. 46
6.5 Results for the question number 5 of the inquiry. 46
6.6 Results for the question number 6 of the inquiry. 47

LIST OF FIGURES

List of Tables

2.1 Comparison between Lex, DialogFlow, WIT.ai and LUIS 13

Abbreviations

AI	Artificial Intelligence
AIML	Artificial Intelligence Markup Language
CPU	Central Processing Unity
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
LUIS	Language Understanding Intelligent Service
ML	Machine Learning
NLP	Natural-Language Processing
UI	User Interface
UWP	Universal Windows Platform
VPN	Virtual Private Network
XML	Extensible Markup Language
WWW	World Wide Web

Chapter 1

Introduction

For over of 50 years that chat-bots have been developed. The first chat-bots to be developed were based on simple rule systems that transform what the user writes into questions, giving the illusion to users that on the other side of the machine there is a human that actually cares and understands what is written.

Nowadays chat-bots are far more advanced than the first ones. Many of these new chat-bots use the latest machine learning techniques to generate responses based on user input. They learn to predict the intentions of a user from previous conversations, in other words, the past conversations are used to train machine learning models that predict the intentions. These new chat-bots achieve a more natural and intelligent dialogue with the users.

We live in a century of new technology and innovation. Each day more companies have a website on the Internet and hundreds of new mobile applications are released to the public. Documents that were previously stored in paper format are now being stored in digital format in the cloud and accessible anywhere. The transformation of making everything available anytime with just a few clicks is also happening with many software applications that were previously required to be installed on the computer but are now accessible through a browser. These platforms hosted in the cloud are called web platforms.

1.1 Context

With the increasing complexity of web platforms, the need for a help-desk has become mandatory. This is even more significant in a management software, as these have an enormous number of tools. Many of these tools, require the user[1] to follow a multi-step flow to get to it.

However, having a help-desk does not solve it all. Even with a phone line to help the users, it probably will not be available all the time. Answering user's questions regarding how to get to the

---

[1] A person with access to the platform, can be a collaborator, security or UPTEC staff.

tools by email, takes also a long time and sometimes only in the next business day the user gets the answer.

Nowadays, the processing power of CPUs is enough to run AI algorithms that could not run before. This makes it possible to create AI bots that can fully replace these help-desks. The bot is available all the time and answers in real-time, allowing users not to spend much time finding the tools they need. There are some examples of what these changes made possible to develop, for example, Apple's Siri, Microsoft's Cortana, Google's Assistant.

The conventional chat-bots use AIML[2], a pattern matching XML parser, as a response system. This technique cannot guarantee a coherent chat as it is hard to avoid getting repetitive responses and all knowledge must be present in the AIML file, this will worsen the user experience. [Beh16]

Natural Language Understanding is a subfield of computer science concerned with the use of computational techniques to learn, understand, and produce human language content [Can18] that has techniques that improve the overall experience of an interaction between a human and a bot, from translation tools (Google Translator), text-to-speech (accessibility tools) and of course chat-bots.

UPTEC[3] is a startup incubator that currently hosts more than 200 companies. These can be one of three possible types. The first is a startup, these are usually hosted in UPTEC for the duration of 3 years, having to leave after that. The second one is investigation centers: the name is self-explanatory, companies that are in the scientific investigation area. Finally, anchor projects are companies that have accepted the invitation of UPTEC to be there hosted; these can be of any age.

Each company here hosted has the right for one or multiple office rooms, some parking slots, access to the common meeting rooms with some of them having their private meeting rooms. All these accesses are regulated through access cards that can be requested by companies for their collaborators[4].

For using the common meeting rooms, it is mandatory to make a reservation previously to the meeting through email, that will be validated by a UPTEC collaborator and only after an approval it is considered as reserved. There are limitations for these reservations such as, a company can not make a reservation with more than a given number of days prior to the desired date and cannot make monthly reservations that exceed a certain number of hours to assure fairness among all companies.

The requests for access cards follow a similar workflow to the meeting room reservations, a company requests a card for a collaborator through email, it goes through the security that takes care of configuring the card with the right accesses and then informs the financial department to charge the company for the requested card.

---

[2] Artificial Intelligence Markup Language

[3] UPTEC - Science and Technology Park of the University of Porto supports the creation of technological, scientific and creative based companies and attracts national and international innovation centers.

[4] A collaborator is a person that works in a company hosted in UPTEC.

The registration of a new collaborator is also a manual process, the company is in-charge of sending an email to UPTEC with the personal information about the collaborator so that they process and store this information in an Excel file.

1.2 Motivation

With the increasing growth of the incubator, all this management became exhaustively time-consuming and eventually led to the development of a management platform named UPTEConnect and referred from now on as “the intranet”, that would solve these problems and make the management easier and more efficient. This platform, not yet released, allows the management of all companies, collaborators, requests, events tracking and more.

A company can add new collaborators through the platform without having to send an email with that information. This creation automatically generates a notification to UPTEC in the platform so that any UPTEC collaborators can see who’s new. The requests for access cards, assistance and even login credentials all go through the platform avoiding the need for email exchange and Excel files for information storage. So, now all the information is stored and accessible using a web platform.

With this platform and all new work-flows that were created because of it, a big problem raised that is, if on one side the platform was developed to avoid the huge emails exchange that UPTEC was experiencing, with the release of the platform, collaborators would have a massive “How To Do” questions that would be asked once again through email.

So, before publicly releasing the platform a solution to this problem was needed. This solution is the scope of the dissertation, a chat-bot that can recognize collaborators commands and questions and address them appropriately.

1.3 Goals

The main goal of developing this chat-bot is that the amount of exchanged emails is significantly reduced and the productivity of users in the platform is higher than without the chat-bot.

In order to accomplish the goal, this chat-bot is expected to understand the collaborators’ questions inside the scope of the platform. Further, it has to distinguish between whether the user wants to execute a given action or wants to learn how to do it by himself. To clarify this, lets assume that a collaborator wants to edit a collaborator named “John”: he might ask “I want to edit collaborator John” and in this case the chat-bot should redirect the user straight to the edition page of John; on the other hand, he might ask “How can I edit a collaborator”, in which case the chat-bot should generate a dynamic tutorial with pop-ups in the platform that guides the user to the edition page of a collaborator, step-by-step.

One of the most challenging aspects of understanding the collaborator intention is that on a given subject there are multiple possible actions, for example, collaborators can be created, edited, disabled, having an access card requested and more. In order for the chat-bot to distinguish all these possible actions, it would have to be trained using different synonyms for each possible action on each subject.

The other challenge was the use of contexts, for example, if a collaborator writes "I want to edit a collaborator" without specifying the name of the collaborator, the chat-bot would have to ask for the name and know that the response of the collaborator would be the name of the target collaborator. All of this while continuing to understand that if the collaborator had asked "I want to edit the collaborator John", "John" would be the name of the collaborator.

Finally, the chat-bot must be able to know that a user *wanting* to edit a collaborator is the same as the user *wishing* to edit a collaborator or *needing* to edit a collaborator or even *desiring* to edit a collaborator.

The training of the chat-bot for language recognition is done with the use of machine learning algorithms that are fed with examples of collaborators questions for each possible action that can be executed in the platform.

1.4 Structure

The rest of the dissertation is structured as follows:

- **Chapter 2** explains the background of the dissertation. This allows everyone, even those without any knowledge about chat-bots, to read and fully understand the document.
- **Chapter 3** has its focus on the evolution of chat-bots, from the first chat-bots that used simple rule-based systems, to chat-bots that adapt the speech according to the emotional state of the user.
- **Chapter 4** is dedicated to the approach that was used in the development of the chat-bot. All use cases, the overall structure of the platform and the requirements are specified in this chapter.
- **Chapter 5** focuses on the development of the platform. How the components were implemented, how they are connected and how they communicate. The challenges that were faced in the development and the solution are also explained. Diagrams of the general architecture, chat-bot architecture and classes of both chat-bot module and user interface module are displayed.
- **Chapter 6** a chapter that describes the experiments that were done and under which conditions. An explanation for each of the questions on the inquiry and the end results of the inquiry.
- **Chapter 7** the last chapter of the dissertation, it is the conclusion and the future work. A brief overview of what was developed and what can be improved in the future.

Chapter 2

Chat-Bots

This chapter explains the concepts of chat-bots and will compare the most used NLP frameworks.

Firstly some key concepts are explained, then the architecture of a chat-bot so that the reader can understand the common flow of a chat-bot. The different types of models, domains, and conversations will be described. As an extra, some design techniques will be mentioned.

The usual flow to match what the user writes to an action is also explained. The Turing test, created by Alan Turing is introduced as it will be mentioned several times throughout the dissertation.

Last but not least, the main natural language platforms to develop chat-bots are explained, the key features, advantages, and disadvantages are mentioned and a comparison table between them all is shown. The platform that was chosen and the reasons that support the decision is the last section of the chapter.

By the end of this chapter, any reader should be able to fully understand the project that was developed, the concepts that will be mentioned in the dissertation and the design choices that will be also described.

A chat-bot is, in its most basic form, an interface that supports the interaction between a user and a computer on a turn-by-turn basis. Chat-bots can be *transactional* or *conversational* [Mct18]. A transactional chat-bot is used to achieve a specific goal, be it schedule a meeting, creating a memo or asking for the weather conditions. One of the main advantages of this type of chat-bots is that it does not have to be context-aware. This fact simplifies the design and implementation of the bot. The other possible type of chat-bot is conversational. In this type the bot engages in a "chit-chat" and it is mandatory that the state of the conversation is kept. The chat-bot responds to the user based on the current context.

In the last couple of years, the industry demand for AI chat-bots highly increased as they are excellent replacements for help-desks and long support email conversations. One of the main

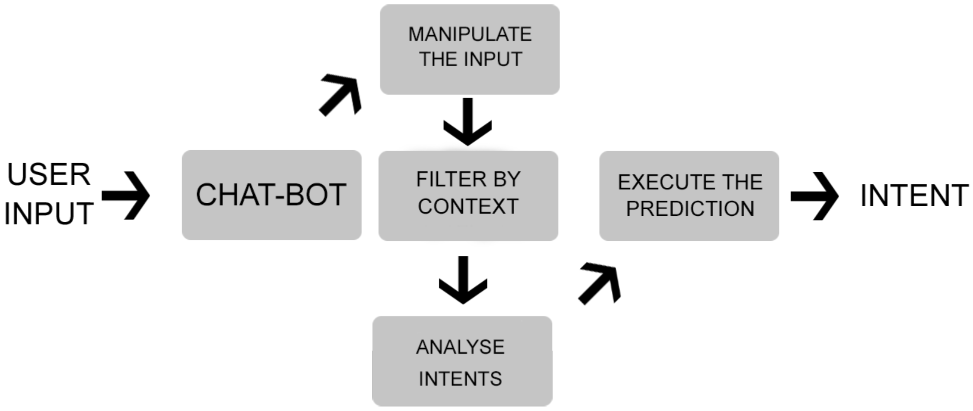


Figure 2.1: Common flow that a chat-bot uses to predict the intent given the user input

applications is in the online customer support as it saves a lot of time and resources to the company [BDC17] and gives to the customer, an always available agent that he can use to ask questions.

Before moving forward, some basic chat-bot concepts have to be explained. These are:

- **Input**: the text that the user enters, it can be a command like "Turn on the lights", a question like "What is the weather conditions for tomorrow" or just a greeting.
- **Actions**: the possible actions that the bot can execute. Usually, the scope of a chat-bot is limited and so are its actions. This is not always true as it will be explained later on.
- **Intent**: the user intention, ideally the bot manipulates the input of the user and predicts its intention correctly.

A common flow to predict the intention of a user given his input consists in these steps: the user sends some text to the chat-bot, the input is then manipulated and an algorithm gets applied to the manipulated input. This algorithm looks at all possible user intents and returns the predicted intent. A flow diagram of this process is displayed in Figure 2.1.

Finally, the predicted intent is used to execute some action or generate a response to be sent back to the user.

2.1 Architecture

Building a chat-bot requires professional skills even for a very basic one. Usually, the developer builds the knowledge base in a frequently asked question format. However, this is not always true as some platforms provide a learning environment [AJ15] similar to the ones that are going to be described in Section 2.3.

In any case the modules of a chat-bot are similar and divided in three main blocks: *responder*, *processor* and *classifier*.

- **Responder**: the interface between the user and the classifier. It routes the inputs and outputs from and to the processor. After the user has input some text, that input is sent to the processor that will then return a response.
- **Processor**: the processor is responsible for manipulating the input. These manipulations can be normalization, extraction of the keywords, segmenting the input, filtering, or other.
- **Classifier**: the classifier uses the result of the manipulation and applies some kind of algorithm on it. This operation returns the matched intent, the intent is sent back to the processor. Several algorithms can be applied, for example, pattern matching that tries to match the input against all configured rules until a match is found, a machine learning model or many others.

2.1.1 Models

When designing a chat-bot, a model choice has to be made. There are two different types of models that will affect the design and architecture of the bot:

- **Retrieval-based models**: on a retrieval-based model uses a knowledge base with predefined responses. To select the response, a heuristic is applied to the input. Independently of the applied heuristic, it should use the context for the selecting the response. As mentioned before, this heuristic can use machine learning techniques to detect the user intention. These types of chat-bots do not generate text but pick a response from a fixed set. A major disadvantage is that these are unable to handle unseen inputs for which there are no predefined responses. Due to the simplicity, this model is commonly used as a replacement for the frequently asked questions page on a website.
- **Generative models**: as its name might suggest, generative models are models that generate responses from scratch not relying on a set of predefined responses. Based on machine learning techniques that convert an input to a response, these systems can handle unseen inputs. This is a more complex model that the retrieval-based one and very prone to grammatical errors. On the downside, we have the fact that, these "smarter" models, need a huge dataset to be trained.

2.1.2 Domain

Another choice that has to be made before start building a chat-bot is the domain of its responses. There are two possibilities, these are:

- **Closed domain**: the simplest of the two possibilities. A chat-bot can be trained with a fixed possible inputs and outputs set because the scope of the domain is limited and so is the knowledge required by the bot. The knowledge base of the chat-bot is formed by the collection of inputs and outputs. It is possible that a user takes the conversation anywhere they want, however, the system is not required to handle all these cases.

- **Open domain**: on the other side we have the open domain chat-bot, here the user is allowed to take the conversation anywhere and it is expectable that the chat-bot understands. It is very hard to train a system like this one, due to the infinite possibilities for topics and the difficulty to get the knowledge for all them. Recent techniques of deep learning are being investigated to make developing open domain chat-bots easier [Koj16].

2.1.3 Conversation

Nowadays two very distinct conversation models are being implemented on chat-bots. On one hand we have the one-shot query model that demands a proactive user and on the other hand, the slot-filling dialogue model where it is the chat-bot the responsible for being proactive in the conversation [Mct18].

The first model, that is the one-shot query, is widely used due to the fact of being more intuitive for the user. It is in the form of input-output. A user either asks a question or issues a command. Usually, there are no follow-up questions, so, subsequent inputs are treated independently on the previous ones. As an example of this model:

— **User**: Who is the president of Portugal?
— **User**: Remind me to cook dinner at 21.

In slot-filling dialogues, the interaction is controlled by the system. The responsibility of leading the dialog is from the chat-bot, it is the one that asks the questions. This is an example of how the chat-bot controls the interaction:

— **Chat-Bot**: What's the budget for your trip?
— **User**: 1000 euros.
— **Chat-Bot**: How many persons are traveling?
— **User**: 3.

2.1.4 Design Techniques

Improving the human-machine interaction is something that whoever is building a chat-bot should bear in mind. Creating conversations that look natural and more human is a challenge. Addressing this subject, there are multiple design techniques and approaches that can be applied when developing a chat-bot.

- **NLP Techniques**: these techniques are used to parse the input, to do the lexical and semantical analysis and even the manipulation of the text [FPL+09].

- **Pattern matching**: in this technique, an input is tested against multiple rules until one match is found [ASBZ15].

- **Databases**: use databases to store the conversations and information about the user. If this information is later used during the conversation, the user is more likely to feel to be talking with a human as the chat-bot will have already some context on the user [XLW+17].

- **Markov Chain**: a probabilistic algorithm that generates the responses given a historical data set of conversations. An algorithm that is widely used in the computation field, namely machine learning [AA17].

2.2 Matching the Intent

After a user inputs a phrase to a chat-bot, this input has to be manipulated in such a way that the bot can match it with an intent.

A typical flow for matching an input to an intent and subsequently execute an action or return a response is the following:

- Filter: filter the input by removing prepositions and articles from it,

- Keywords: extract the keywords from the input, these can be for example nouns, verbs, etc.

- Algorithm: apply an algorithm be it pattern matching, machine learning model, etc that matches the bag of keywords with an intent

- Response: once the intent is matched, apply the command that the user instructed or return the appropriated response.

The flow steps are shown in the Figure 2.2.

2.3 Natural Language Platforms

The search for chat-bots that help people in their business increased in the last years. This is proven by the more than 200.000 chatbots for Messenger and more than 10.000 Amazon Alexa skills created by the chat-bot developer community [Mct18, Joh17]. These include chat-bots that manage online orders, to chat-bots that redirect the user to the right department inside a company to even marketing chat-bots.

Previously it was needed to implement all the logic of a chat-bot, the graphical interface, the natural language processing algorithms, the machine learning algorithms for the intent detection and even the diagnostic tools for the future training of the chat-bot.

Because of the high-demand of chat-bots, multiple platforms that abstract the user from this logic were developed. At the time of this dissertation, there are four main frameworks, namely: Lex, DialogFlow, WIT.ai and LUIS [1].

---

[1] Language Understanding Intelligent Service

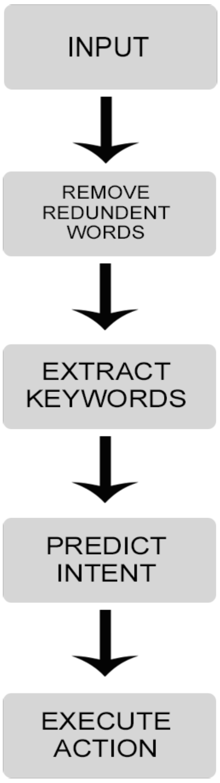


Figure 2.2: Matching the intent flow

2.3.1 Lex

Lex is an open-source AI assistant developed by Amazon. It has an excellent integration with AWS Lambda [2] for the business logic processing. Very good for simple applications with limited and distinct commands, great if the application needs full control because all the business logic must be implemented from scratch, that is sessions, contexts, required parameters and others [Kan17].

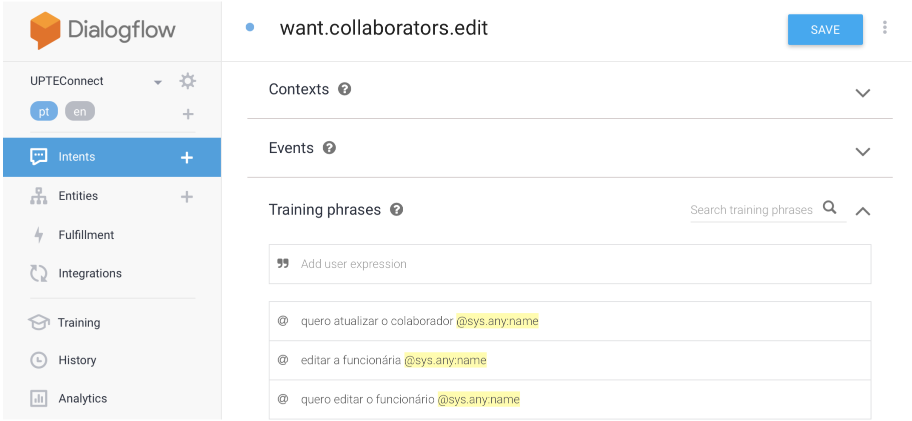
The biggest disadvantage is the need for a web-hook call on every triggered intent, that can easily increase the costs associated with the chat-bot. Also weak in the machine learning algorithms performance. The developer is in charge of retrieving the missing parameters in the business logic. Further, it only has one language available for use.

2.3.2 DialogFlow

DialogFlow formerly API.ai was recently acquired by Google. The important concepts here are the contexts and intents that are used to model the behavior of the chat-bot. Contexts are used to distinguish different inputs that might have different intents given the historical conversation with the bot. [Dut17] Each intent might have several *follow-up* intents associated, making easier for a

---

[2] The Amazon Web Services to host lambda functions



quero atualizar o colaborador @sys.any:name

editar a funcionária @sys.any:name

quero editar o funcionário @sys.any:name

Figure 2.3: DialogFlow user interface for creating new intents

developer to build the conversation flow. For example, in case that the intent was "Order-Now-Confirm", the follow-up intents could be "Order-Now-Yes" and "Order-Now-No" with each one having different responses and associated actions.

Machine learning is a feature in the platform that allows the chat-bot to learn new commands semantically similar to those input by the developer during the training. A *Default Fallback Intent* is also provided in the case that the intent could not be determined and a *Welcome Intent* that greats the user when joining the chat.

The platform also provides an easy to use UI that help developers creating new intents and entities. The management of parameters and contexts can be done through this interface and also the marking of entities/parameters as mandatory so that the platform can retrieve them without having to go to the business logic. For example, in contexts, it is possible to set the lifespan of an output context that is decreased on each message sent by the user. Once it the lifespan gets to zero, it is discarded. Furthermore, DialogFlow has a rich set of domain entities like weather, flights, people and more.

It has a lot of built-in integrations with other chat-base platforms such as Telegram, Messenger, Google Assistant, and others.

An example of the UI of this platform can be seen in 2.3

2.3.3 WIT.ai

Another platform that was recently acquired by one the tech giants, Facebook. Similarly to DialogFlow, WIT.ai can learn new commands to match existing intents with the use of machine learning techniques.

A feature-rich UI that allows the creation of entities and intents. It provides a visual representation of the conversation flows and business logic invocations. It used to had contexts that could be configured using the interface but decided to abandon it because of the required more processing power, after all only a small portion of the chat-bots were actually using contexts.

An important feature is the notion of roles, so, for example, saying “I want to go from Porto to Lisboa” both Porto and Lisboa are location entities but you can further distinguish between a fromLocation (Porto) and toLocation (Lisboa).

Although it has a rich set of features it lacks on the required parameter feature, forcing the developer to invoke the business logic after every interaction to gather missing information that was not written by the user.

As opposed to DialogFlow, it does not have any integration with third-party applications.

2.3.4 LUIS

Developed by Microsoft, another platform that can learn new commands by using machine learning techniques with a UI that allows the creation of intents and entities. This solution looks to be better than all other solutions but ties the developer to the Windows Platform.

The platform provides useful metrics to understand how well the AI agent is working. Has also what is called composite entities, for example in "2 chocolate milk", two is the number, milk is the beverage, chocolate the flavor of the beverage. All together form a composite entity. This is the only platform that has this feature.

The management of contexts cannot be done through the platform nor the intent filtering by context, all of this has to be managed by the business logic. Furthermore, building with LUIS requires the developer to build UWP [3] applications tying you to the Windows platform.

2.3.5 Comparison Table

The table displayed in Table 2.1 compares all these NLP platforms. The features that are being compared are the following:

- **Provider**: the company that owns the platform,
- **Training Model**: allows the use of machine learning to execute the predictions. This allows the developer to use input examples to train the algorithm so that the algorithm can predict the intent even when the input doesn’t exactly match any of the examples.
- **Built-in Sessions**: possibility to configure sessions using the graphical interface.
- **Built-in Contexts**: option to configure the contexts of each intent using the graphical interface. The platform does all the context management.
- **Pre-Built Entities**: has a knowledge base of different entities that can be extracted from the input, for example, date, locations, names, etc.
- **Custom Entities**: a feature that allows a user to add new, custom, entities to the knowledge base so that they can be extracted from the user input.

---

[3] Universal Windows Platform

Table 2.1: Comparison between Lex, DialogFlow, WIT.ai and LUIS

	Lex	DialogFlow	WIT.ai	LUIS
Provider	Amazon	Google	Facebook	Microsoft
Training Model	✓	✓	✓	✓
Built-in Sessions	✓	✓	✗	✓
Built-in Contexts	✓	✓	✗	✗
Pre-Built Entities	✓big list	✓more than basic	✓basic	✓basic
Custom Entities	✗	✓	✓	✓
Composite Entities	✗	✗	✗	✓
Speech Recognition	✓	✓	✓	✓
Diagnostic Tools	✗	✗	✓	✓
Conversation Flows	✗	✓	✓	✓
Webhook	✓	✓	✗	✓
Multi-Language	✗	✓(15)	✓(50)	✓(10)
Free	✗(trial)	✓	✓	✓(basic plan)

- **Composite Entities**: the creation of entities that are the result of the combination of multiple entities.

- **Speech Recognition**: possible to convert the voice to user input. A common feature in personal assistants like Siri, Cortana, etc.

- **Diagnostic Tools**: tools that allow the developer to see true positives, true negatives, false positives, false negatives related to the intent and entity matching. Metrics on how the intents and entities are being resolved.

- **Conversation Flows**: an interface that allows the developer to see the flow of the conversations between the chat-bot and the users.

- **Webhook**: option to add callbacks that are invoked, for example, upon the prediction of an intention.

- **Multi-language**: possible to train the chat-bot in multiple languages.

- **Free**: whether or not the platform is free or has a trial version.

2.3.6 Chosen Platform

DialogFlow was the chosen platform over all others. This is one of the most complete platforms, very similar to LUIS. However, it does have some features that made it the chosen option: it has built-in contexts avoiding the need to implement this logic from scratch on the server side.

On top of that, the application does not have to be developed with the UWP (like LUIS has) and so, it can be hosted anywhere which is very important because the chat-bot business logic must be hosted in University of Porto.

Furthermore, it learns new statements from the known statements using Machine Learning algorithms. This is especially important for users, in the beginning, as they will not know how to ask for the tools and different users will not ask for the same tool using the same inputs.

Finally, the UI is super rich and simple, feels comfortable to work on it.

Chapter 3

Evolution of Chat-Bots

This chapter focuses on the related work in the area of artificial intelligence chat-bots.

Chat-bots exist for more than 50 years. The first one to be developed was ELIZA in 1966 [Wei66], a chat-bot that later on tried to pass the Turing test (this test is described in Sub-section 3.1) but was not successful. This Turing test was supposedly passed in 2014 by Eugene Goostman [Ack14].

In the generation of responses, the input is scanned from left to right and decomposed using a rule system. If a word is a keyword then only decomposition rules that have that keyword are used. The input is then matched to the decomposition rules and if it matches, then the reassembly rules are applied to the input. The result is the response sent to the user, giving the user the illusion of understanding by ELIZA.

The most famous ELIZA script was DOCTOR. DOCTOR was a script that simulates a psychotherapist. The script assigns weights to each word based on the importance of it in the sentence. Assigns low values to pronouns, slightly higher values to action verbs, and the highest value to the actual action. This allowed the ELIZA to know the exactly where to flip the sentence around to ask a question.

Later on 1972, PARRY was developed by Kenneth Colby. PARRY that "met" DOCTOR several times and "talked" to it, one of the most famous conversations was stored and known as *PARRY encounters the DOCTOR* [Cer73]. It has some new tricks such as admitting ignorance, for example, 'I don't know', changing the subject of the conversation and some Mafia histories that are told in a fixed order.

In 1991 Hugh Loebner started the Loebner prize competition, offering a prize to the author of the first computer program to pass an unrestricted Turing test [Mau94]. Some years after, in 1998, TINYMUD was developed, a multi-player conversation with textual “scenery” simulating physical spaces and user extensibility. It was used for Turing tests given the large pool of potential judges and interviewees. It was in this game that CHATTERBOT was invented, a robot TINYMUD player that used all the illusion techniques from PARRY and ELIZA but adds more. It uses humorous statements, creates controversial statements so that it can direct the discourse rather than allowing the user to do it, simulated typing by including realistic delays between characters and others.

Still, in the same decade, Richard Wallace invented ALICE (Artificial Linguistic Internet Computer Entity) inspired in the historical ELIZA. ALICE won the Loebner prize competition three years in a row (2000, 2001, 2002), however, it could not pass the Turing test because the conversation mechanisms are ofter exposed even in short conversations [Wal09].

As mentioned above, the first chat-bot to pass the Turing test was *The Goostman bot* after having fooled more than 30% of the judges on the 60th anniversary of Turing's death contest [Sco16].

Some Japanese hotels replace the receptionist, an example being SAYA chat-bot [DD08]. SAYA is based on the ALICE AIML and Program D. This bot is able to remember things, if a person says his name then SAYA will remember it until the person says otherwise or the conversation ends. If SAYA does not receive any new queries within a time span, then it will return to the original state, forgetting everything that was memorized.

Another application was in recognizing user's personalities with an online shop chat-bot assistant, with the goal of improving man-machine interactions [Ho18]. This online shop assistant classifies a user's personality based on the Human Personality Types (HPTs) introduced by Horzyk [HMM09], after classifying the user, the chat-bot adapts the speech improving the overall satisfaction of the client.

Teaching assistants were also a target for a replacement for chat-bots due to the shortage of staffing and resources in the teaching industry [Ho18]. A DialogFlow powered bot that is able to recognize entities like the course, materials required for the class, the classroom where the class is held, assignments, exams, deadlines, learning outcomes, and others.

Another chat-bot that was developed is the one being used in the Echo platform, a closed-domain chat-bot (travel) that analyses the user preferences and predict collective intelligence. In most of the cases, the predictions are user-centered [AGM+18]. This paper describes the use of a Deep Neural Network (DNN) to execute the predictions. The system first identifies the missing information and makes questions to the user to collect this missing information. Still, in this paper, it is shown how to use Restricted Boltzmann machines [1] to model different kinds of objects like flights, cars, and others.

Appointments scheduling has been a problem for a long time, its a very time-consuming task as it is needed that all participants in the appointment agree on a date [BDD+97]. To overcome this problem COSMA was developed, a German-speaking chat-bot that allows humans to schedule appointments. A human could have a machine responsible for scheduling his appointments, accordingly to his calendar, without his interaction.

Mercury flight reservation system also has implemented a chat-bot to handle the flight reservations [SP00]. This system uses the GALAXY architecture, that consists of a group of servers that

---

[1] Restricted Boltzmann machine is an algorithm used for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modeling

communicate with one another through a central programmable hub. The audio server captures the user dialog that gets processed by the speech recognizer. The result is the text input of the speech that is used to generate the responses.

A museum conversational agent, Max, has been built and employed as a guide in a computer. He engages users and provides information about the museum or the exhibition [KGKW05]. Max uses video-recognition to track people, for example when a person enters he will greet him. Further, Max is equipped with an emotion system that dynamically runs the simulation model for its emotional state. This process adjusts the pitch and speech rate of his voice.

Recognizing user's emotions and using them to adapt the responses is something that makes the conversation feel more natural. A chat-bot for a psychiatric counseling service was tested [LOC17]. To detect user emotions the system had four levels of hierarchy. This system enabled the NLP module to understand the user.

3.1 Turing Test

In 1950, Alan Turing created the widely known Turing test [A. 50], also known as *the imitation game*. This test tests the machine ability to show behavior that is intelligent or indistinguishable from a human. A group of human participants (evaluators) is selected to test the machine. All of them are separated from one another and they all know that on the other side there is a human or machine. The conversation is limited to text-only. A machine is said to pass the Turing test if the evaluator cannot reliably tell if it is a machine or human. The results do not depend on whether or not the responses are correct but if they are undistinguished from human responses.

3.2 Generation Models

ELIZA, the first-chat bot used one simple rule-based system to generate its responses.

Multiple rule-based and template-based generation systems have been developed, both of them have limitations in spoken dialog. A new approach to natural language generation is proposed by Alice and Alexander [OR02]. The first approach that worked, avoided repeating information during the generation of responses, so the information given by the user would be repeated only once by the system, usually right after the user told it. This doesn't look natural because humans often repeat information during a conversation. The second approach uses a statistical model that tries to mimic human behavior.

In the open domain, chat-bots Ameixa developed Filipe. Filipe is a chat-bot that generates the users' responses using movies subtitles [ACFQ14]. This way the chat-bot can now answer several questions that before was unable to deal with.

Later, it has been proven that hierarchical recurrent neural network generative model can outperform n-gram and baseline neural models [SSB$^+$16]. In this paper, based on Ameixa (2014) paper, and to support their discovery, it was used a dataset of subtitles called *MovieTriples*, known as being very appropriate for open domain dialogues.

Rafael built IRIS, a project that used the vector spaced model for storing the inputs [BL12]. IRIS chat capabilities use a dual search strategy over a knowledge database based on movie dialogues. It is capable of learning new vocabulary terms and relating them to previous knowledge, it also adapts the responses accordingly to the user preferences. The new vocabulary is learned with the help of the user, so if IRIS doesn't know what a word means, then it will ask the user for the explanation. After the new information is validated, it is stored in the vocabulary repository. However, this chat-bot is incapable of being consistent.

When the questions of a user are out of context a chat-bot can simply give a generic response such as "I don't know". However, that becomes repetitive and doesn't look natural. Creating a set of off-topic responses also doesn't help as it once again doesn't look human. A categorization of off-topic questions for open domain chat-bots has been studied by Patel [PLT06]. Patel proposes to split the off-topic questions into different categories, adapting the responses accordingly. A set of responses for when the user is not coherent or wrote the input with errors and a set for yes / no responses. For the remaining ones, distinguish between those questions that are really out-of-domain or the ones that are in-domain but doesn't have an appropriate response. The in-domain questions were further split into more categories, questions that are forbidden, that need to be deferred to someone else and more. The classification is done with four classifiers that slightly improve the appropriateness score comparing to the off-topic non-categorized version.

A new approach for developing open domain chat-bots is the use of Wikipedia. In order to reduce the set of expected answers, Waltinger uses the taxonomy of Wikipedia as a reference point, in other words, only the knowledge base that is relevant for the user question is activated and used.

The museum agent by Kopp (2005), generates the response in three steps. Firstly it interprets the incoming event (people joining, leaving, etc), decide how to react depending on the context and finally generates the appropriate response. Max has a static (long-term) knowledge base such as information about the museum, former dialogs with visitors and a dynamic knowledge that is responsible for holding the context. Most of the actions are implemented in a condition-action rule model, extending the ALICE system.

Current researchers are using Deep Learning techniques in the retrieval-based and generative models but seem to head towards the generative models. However, on production systems, it is more likely that retrieval-based models are being used [Koj16].

3.3 Methodology

User input words will be the key to predict the user question, so the way they are processed greatly affects the output results.

Enumerating the possible intents and collecting example questions for those intents is a must. For each intent the top N keywords from the example questions are extracted, the reason to extract from each intent and not globally is that some intents might have much fewer instances than others. Stop-words are removed to decrease the complexity of the algorithm, these are insignificant words like articles and prepositions [BDC17, BKS17].

Another important discovery by Yuanjun Bi is that verbs play an important role in the question and that the closest to the end of the question the verb is, the more important it is to predict the intent. So, different weights to verbs were assigned accordingly, that is, verbs closer to the end of the user input had a higher weight than those in the beginning.

Converting the input words to a format that a computer can quickly process is a challenging process, word embeddings and LSTM are two possible solutions [Mil17]. One way to do word embedding is to use hot vectors where each individual word in a language is an NxN vector, having N as the vocabulary size. If the vocabulary expands it becomes very hard to manage this vectors. Another way is to use skip-grams, a model that focuses on features relationships. So if there are X features, each word will only be represented by an X length vector. As mentioned before LSTM is another possibility, this is a form of recurrent neural network (RNN) that learn long-term dependencies, this kind of RNN predicts the likelihood that a word is in an input given the previous input words.

Seneff (2000) in the Mercury flight reservation system, used a different representation system for the knowledge, as this is represented in the form of semantic frames. At each dialog turn, it begins with an electronic form that represents the constraints of the current query. It also retains the history frame. This frame is used in each dialog turn and contains the user preferences.

Building a query expansion was the method used by Waltinger (2011). So, after a user inputs the question, the reasoning module is activated and analyses the question structure. The subjects of it are extracted to filter the possible answers. These subjects are matched against the Wikipedia categories and the result categories will be memorized as the context for the next dialog. An approach that contributes to the enhancement of the agent's conversational behavior of an existing architecture of the virtual human *Max*.

3.4 Matching the Intents

Shaw introduces a simple system that parses rules and matches the input with answers [Sha14]. The program searches the input for keywords or phrasings and matches them with a set of predefined answers. It avoids the complex process of lexicographic analysis and deconstruction.

A possible approach to match a question with the corresponding intent is to use Jaccard similarity algorithm (see Equation 3.1) that for a given intent calculates the ratio between the number of matched keywords of the question with the intent and the total number of keywords in the question and intent.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

Another approach is to use Naive Bayes classification [NSK12], a probabilistic categorization approach (see Equation 3.2).

$$p(C_i|x) = \frac{p(C_i) \times p(x|C_i)}{p(x)} \tag{3.2}$$

For each intent, the probability is calculated and the intent with the highest probability will be the output prediction. Behera used this approach in the development of a semi-automatic [2] chat-bot, the input is categorized into one of the services provided by the company, for example, food, travel, utilities, etc. After that, the intent and sub-category are obtained. The Behera approach is an interesting approach as the conversation is split into milestones, having each milestone a generated AIML pattern and response.

An important method is the Cosine Similarity given by Equation 3.3. This method considers the documents as vectors, so the similarity here is the correlation between them. Each vector of terms has m dimensions and each term has an associated weight. The result is bounded between 0 and 1, it is also independent of the length of the document [Ak17, BL12].

$$cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{||\vec{x}||\cdot||\vec{y}||} \tag{3.3}$$

3.5 Security Issues

An increasing number of chat-bots are developed each day. Many of these chat-bots use communication platforms like Messenger, Skype and Google Allo for the user interface. This means that all messages are stored in third-party services where the user has little, to no control.

For security reasons, UPTEC's intranet can only be accessed inside UPTEC's network or through a VPN connection to UPTEC. All the information stored in the intranet platform should be confidential.

When a chat-bot is integrated with Messenger, this means that all messages will be kept on Facebook servers. This is particularly dangerous when these messages contain information like collaborators working on a certain company or any private information of a user.

DialogFlow has the option to log the dialog history but it is possible to disable it. Although DialogFlow has the possibility to integrate any of these third-party platforms, it is possible to integrate it directly in the platform without using this third-party messaging platforms. This allows UPTEC to be fully in control of what messages to store and where to store them.

On top of that, integrating DialogFlow directly in the intranet makes possible to develop custom user interfaces for the dialog boxes with the chat-bot. The user interface looks more consistent.

---

[2] Semi-automatic as it switches to a manual mode once it is able to understand the intent of the user and the kind of service required by her

Chapter 4

Approach

This chapter focuses on the approach to develop the AI chat-bot. The goals of the project will be described in-depth. All the use cases of the project will also be listed. In another subsection, the platform architecture will be displayed, detailing all the modules that compose it. The process to collect the data that was used to train the chat-bot and all the problems in this process are explained. Lastly, the technical specifications such as tools that were used to develop the project.

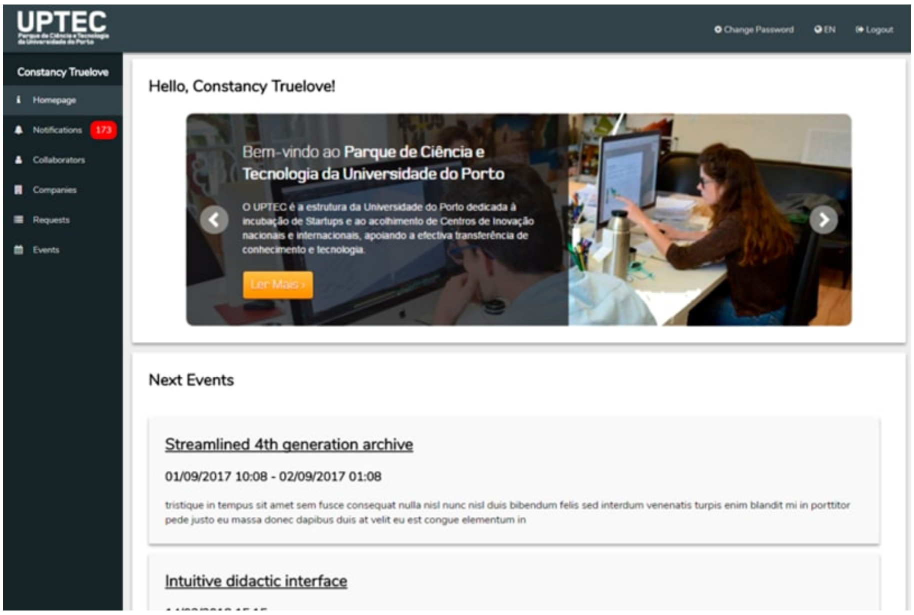
UPTEC's chat-bot is responsible for handling user queries and either teach the user on how to access a given tool or redirect him straight to it. For processing of the natural language, an NLP platform is used and its name is DialogFlow.

4.1 UPTEConnect

Before moving any further an overview of the UPTEC intranet, UPTEConnect, will be given. The intranet was developed with the goal of reducing the amount of exchanged emails and to give more freedom to companies over their own collaborators.

There are three types of user, each of them with different permissions. The most common user type is the *Collaborator*, a person that works in a company. The other two types are *UPTEC* (an UPTEC staff member) and *Security*.

In the platform, a user can access several menus. The home menu has a welcoming message and the upcoming events list. The collaborator's menu lists all the collaborators, if the user is a collaborator then only collaborators of his company will be displayed. Similarly to the collaborator's menu, the companies menu also shows different information accordingly to the user. So, if the user is a collaborator then his company profile will be displayed but if it is a UPTEC or security then a list of all companies will be displayed. Another menu is the requests, this menu has three sub-menus, namely, login requests, access card requests and finally assistance requests. This is the place to follow the state of all requests done through the platform. A calendar with all



Hello, Constancy Truelove!

Bem-vindo ao Parque de Ciência e Tecnologia da Universidade do Porto

O UPTEC é a estrutura da Universidade do Porto dedicada à incubação de Startups e ao acolhimento de Centros de Inovação nacionais e internacionais, apoiando a efectiva transferência de conhecimento e tecnologia.

Next Events

Streamlined 4th generation archive

01/09/2017 10:08 - 02/09/2017 01:08

tristique in tempus sit amet sem fusce consequat nulla nisl nunc nisl duis bibendum felis sed interdum venenatis turpis enim blandit mi in porttitor pede justo eu massa donec dapibus duis at velit eu est congue elementum in

Intuitive didactic interface

Figure 4.1: Home page of the UPTEC intranet, UPTEConnect

past and upcoming events are shown in the events menu. Last but not least, the notifications menu. This menu acts like a notification center, so any change in collaborators, company, requests, events generates a notification that will be here displayed.

Briefly, this platform features the creation, edition and inactivation of collaborators, creation, edition and inactivation of companies, creation of card access requests, creation of login credentials requests, creation of assistance requests, request the disable of access cards and login credentials, follow all the upcoming events, export all data in spreadsheet files and more.

The homepage of the platform is displayed in Figure 4.1.

4.2 Goals

The brand new management platform of UPTEC, at the time of the writing, will be soon released. As expected there will be a large number of emails by users asking on how to get to the tools, something that has to be avoided mainly because the primary goal of the platform was to reduce the number of exchanged emails.

UPTEConnect's chat-bot was developed to avoid users asking questions about the platform using the email. So, its goals were to:

- **Introduce users to the platform**: when a user first joins the platform the chat-bot should introduce the most important features of it to the user.
- **Understand intentions**: the chat-bot should understand the inputs from the user. Further, it should be able to distinguish between the two main possible intentions, the *how* and the *want*, so that different actions can be executed depending on the main intention.

- **Generate tutorials**: when an input is predicted as being a question on how to do something, the chat-bot should be able to generate a tutorial across the platform that leads a user to the desired tool. The tutorial must to be interactive, the user should be able to complete it step-by-step and also move forward and backward in the steps-flow.
- **Redirect to the tool**: if an input is detected as a user wanting to use a given tool, the chat-bot should be able to redirect the user to that tool.
- **Context aware**: the chat-bot should be context-aware in some specific use cases. For example, if a user wants to edit a collaborator and forgets to input the collaborator's name, then the chat-bot should ask for it and know that the next input will be the collaborator's name.
- **Friendly and easy-to-use**: the replacement for a help-desk should be friendly and easy-to-use. A user should be able to use the chat-bot without having to format the input in a specific convention, that is, a user should be able to write as he would usually do and the bot should still understand it. On top of that, the bot should send welcome messages when a user opens the chat dialog, it must apologize when the intention could not be understood and if any information is missing, the chat-bot should ask for it.

4.3 Use Cases

Together with UPTEC the *use cases* were specified and this is the result:

- Once a user logins successfully in the platform, a dialog box should be displayed in the interface so that a user can chat with the bot.
- When a user logins in the platform for the first time, a tutorial must be generated by the chat-bot through the platform, so that the user gets to know the main features it.
- A user must be welcomed when the chat-bot dialog box is opened so that the conversation looks more natural.
- As a user, I want to be redirected to the edition page of a collaborator when I ask the bot to edit a collaborator with a given name.
  - As a user of the type collaborator or UPTEC, if I forgot to specify the name of the collaborator when asking to edit a collaborator, I want the bot to ask for the name of the collaborator.
  - The chat-bot should be aware that after asking for the name of the collaborator, the user response is the name of the collaborator that the user wants to edit.
  - As a user of the type collaborator or UPTEC, if there are multiple matches for that name, I want the bot to show me the list of all matches so that I can redefine my search.

- As a user, I want to be redirected to the creation page of a collaborator when I ask the bot to create a collaborator.
- As a user of the type collaborator, I want to be redirected to the edition page of my company when I ask the bot to edit the company.
- As a user of the type UPTEC, I want to be redirected to the edition page of a company when I ask the bot to edit a company with a given name.
- As a user of the type UPTEC, I want to be redirected to the creation page of companies when I ask the bot to create a company.
- As a user of the type UPTEC, I want to be redirected to the creation page of companies when I ask the bot to create a company.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to create an assistance request.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to create a request to disable the card of a collaborator.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to create a card request for a collaborator.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to add a new collaborator to my company.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to disable a collaborator in the platform.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to edit the personal details of a collaborator.
- As a user of the type UPTEC, I want to be able to ask the bot to show me a tutorial that will teach me how to create a new company in the intranet.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to edit my company details.
- As a user of the type UPTEC, I want to be able to ask the bot to show me a tutorial that will teach me how to edit company details.
- As a user of any type, I want to be able to ask the bot to show me a tutorial that will teach me how to export data from the platform, for example, search results of a query.
- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to disable the access credentials to the platform of a collaborator.

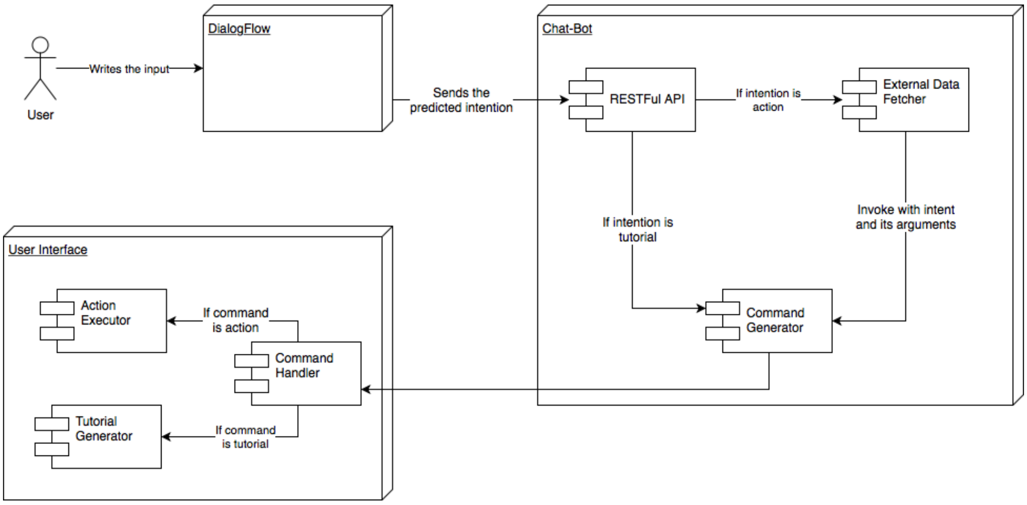


Figure 4.2: Diagram of the interaction flow between a user and the chat-bot

- As a user of the type collaborator, I want to be able to ask the bot to show me a tutorial that will teach me how to create a request for access credentials for a collaborator.
- As a user of any type, I want the bot to tell me when it does not understand the input so that I can try again with a new input.

Every interaction with the chat-bot follows the flow that is displayed in Figure 4.2

4.4 Platform

The chat-bot platform is composed of four modules.

- **User Interface**: responsible for displaying the interaction dialog. The user interface receives the user inputs, sends them to DialogFlow and waits for an instruction. The instruction can be the generation of a tutorial or the redirection of the user to a specific page on the platform. This module only interacts with DialogFlow.
- **DialogFlow**: the platform responsible for processing the input, also known as natural language processing platform. The Google's NLP platform manipulates the input and predicts the intention. This platform is also responsible for the management of contexts and extracting the parameters from the input. This module interacts with the user interface and with the chat-bot module.
- **Chat-Bot**: this module is invoked by DialogFlow, a webhook call. The module checks whether the user wants a tutorial or to execute an action. If the intention is to execute an action then the chat-bot will request the intranet's backend to verify the input parameters (if any) that are sent by DialogFlow. After that, the chat-bots sends the instruction to the UI through the DialogFlow response object.

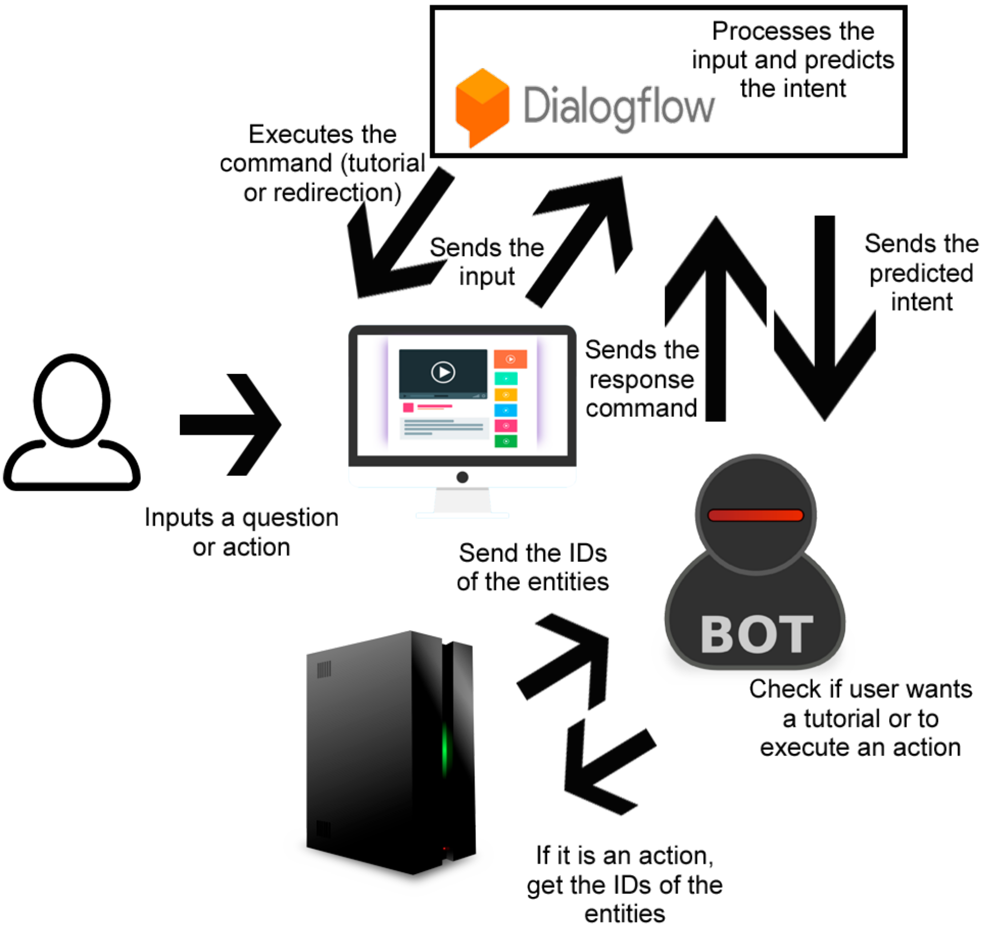

Processes the input and predicts the intent
Dialogflow
Executes the command (tutorial or redirection)
Sends the input
Sends the predicted intent
Sends the response command
Inputs a question or action
Send the IDs of the entities
BOT
Check if user wants a tutorial or to execute an action
If it is an action, get the IDs of the entities

Figure 4.3: Diagram of the developed platform

- **Intranet Backend**: the backend of the platform. Has methods to get the ID of several entities like collaborators and companies. It also authenticates the user of the input so that only information that the user has access to is returned.

The architecture diagram is displayed in the Figure 4.3.

4.5 Data Collection

A chat-bot has to be trained using questions examples so that it can predict the user's intentions. The problem with the UPTEC's intranet is that no one has ever had access to it and because of this, there are no examples of users questions.

DialogFlow suggests a minimum of 10 example phrases for each intent. To address this problem both the developer and two more staff members of UPTEC fed, for each intent, the chat-bot with different questions.

4.6 Technical Specifications

Before the development of this project started, there were some mandatory specifications requested by UPTEC. These specifications were:

- License-free: all the tools used to develop the chat-bot should be free to use.
- Portable: the application should be developed in such way that it can be hosted anywhere, in order words it has to work in multiple environments.

Without any more mandatory requirements, the choice of the tools to use in the development of the chat-bot was the following:

- Go: an open-source project, a programming language developed by Google. Go, also known as Golang, is the main language of the intranet and because of it, its the language used in the development of the chat-bot.
- DialogFlow: DialogFlow is a natural language processing platform. It is possible to create agents, each of them with its own intents. The detection of intents is done using the training phrases provided. This is the natural language processing platform used to process the intranet user's inputs.
- Ng-Chat: this is an Angular 4 module that adds the interface of the messaging box with the chat-bot. This was the chosen module as it works in Angular 4 (the user interface is built on top of it) and is the more mature project.

Chapter 5

Development

This chapter focuses on the development process of the chat-bot platform. The process flow from the user input to the execution of the corresponding action will be explained in detail, how the authentication was implemented so that there are no information leaks through the bot. On top of that, the handling of errors and the challenges of developing this platform will be also discussed.

Along with the details of the implementation, multiple diagrams will be displayed so that the reader can understand these details more easily.

Furthermore, in this chapter, the choices that were made are going to be justified.

5.1 Implementation

In the previous chapter, there is a brief introduction to the architecture and modules of the chat-bot project. Throughout this section, all the modules will be explained in depth as well as the implementation details.

5.1.1 DialogFlow

DialogFlow was the chosen platform to be the Natural Language Processor that drives the project. This decision is explained in Chapter 2. DialogFlow is a free platform with built-in context management and machine learning models that improve the intention prediction of the user.

Starting using DialogFlow is straightforward. The authentication in the platform is done using a Google account. Once the login succeeds, the next step is to create an agent. An agent is a project with its own intentions, custom responses, and speaking languages. With a single DialogFlow account, it is possible to hold multiple agents.

For this project, an agent with the name UPTEConnect was created. After creating an agent, an empty list of intents is displayed on the screen.

For each different tutorial, an intent was created. There is a button in the user interface to create a new intent. An intent has a name, input contexts, output contexts, events, training phrases, parameters, and responses. It is also possible to invoke a webhook from an intent.

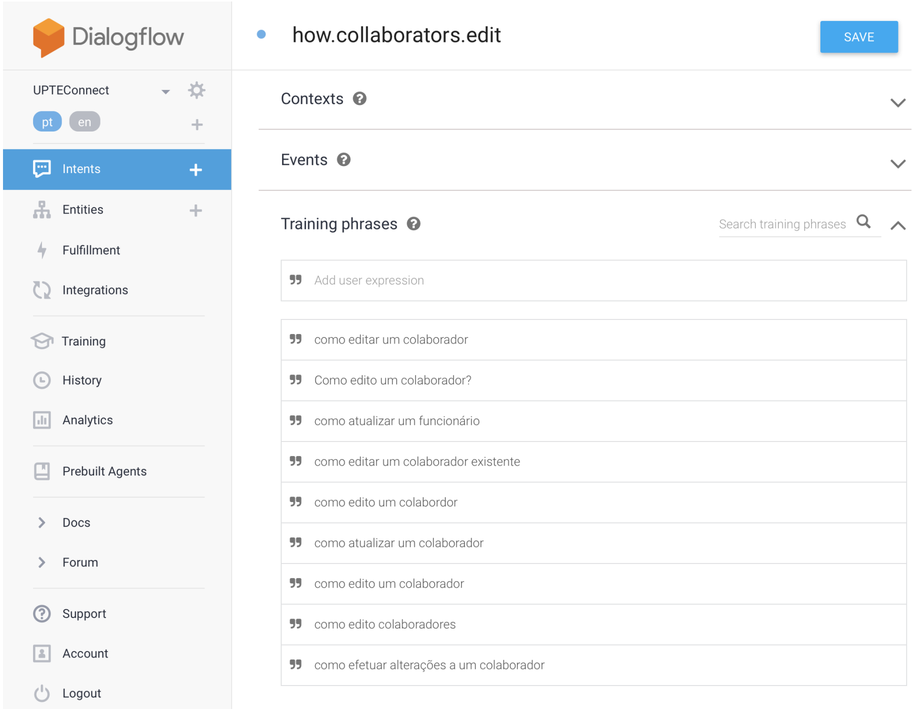
It has been applied a naming convention for the intents, an intent starting with *how* generates a tutorial, with *want* redirects the user to the target page and with *get* requests the user to input some parameter.

This is the intentions list:

- get.collaborator.name
- how.assistance.request
- how.cards.disable
- how.cards.request
- how.collaborators.create
- how.collaborators.disable
- how.collaborators.edit
- how.companies.create
- how.companies.edit
- how.export.data
- how.login.disable
- how.login.request
- want.collaborators.create
- want.collaborators.disable
- want.collaborators.edit
- want.collaborators.edit.noname
- want.companies.create
- want.companies.edit

The intents were not created all at the same time. Each time that a new intent was created, the intent was trained with some input examples. An example page of training an intent is displayed in Figure 5.1.

Later on, more input examples were collected from users. So that more input examples were collected, the chat-bot was released to a small number of users, from now on mentioned as testers,



como editar um colaborador

Como edito um colaborador?

como atualizar um funcionário

como editar um colaborador existente

como edito um colabordor

como atualizar um colaborador

como edito um colaborador

como edito colaboradores

como efetuar alterações a um colaborador

Figure 5.1: Training an intent in DialogFlow

that tested the chat-bot in-depth. Every time that the chat-bot could not understand the tester's intention, the input that was not understood was added to the training examples of that intention.

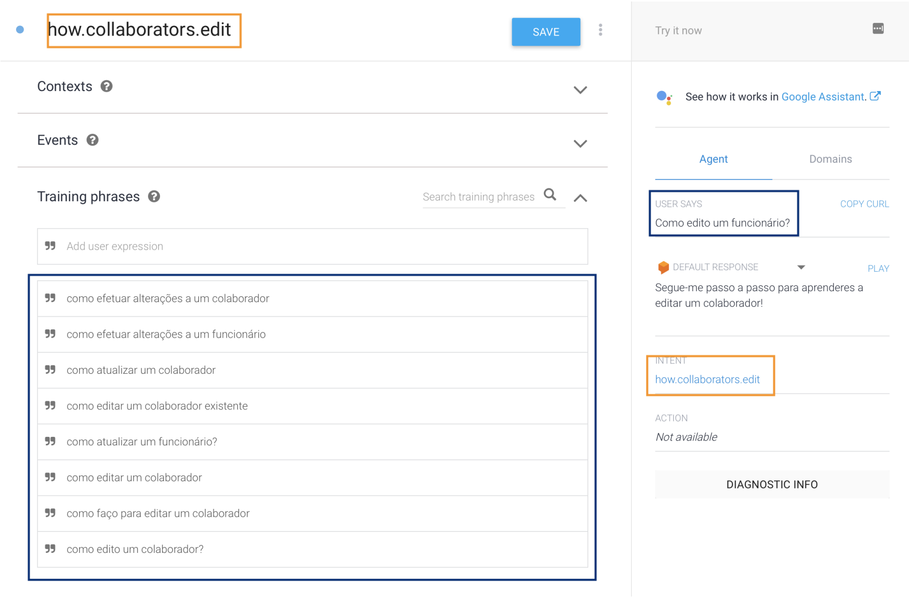
This technique of collecting input examples allowed the training of all user intentions.

DialogFlow uses a training mechanism than infers new inputs for an intention using the training examples. Such feature allows the users to use the platform and having the chat-bot to correctly predict the intent without having to input an exact match to a training input for that intent. It is possible to see in Figure 5.2 that the chat-bot correctly predicted the intent of the user without having an exact match in the training examples.

Further, in the development process, contexts were also implemented in the scenario where a user *wants* to edit a collaborator but forgets to mention the name of the collaborator to edit. The chat-bot was trained to extract the collaborator name from the user input. If there is no name on the input the chat-bot requests the user to write the collaborator's name (Figure 5.3). An example of how the contexts are used in DialogFlow is displayed in Figures 5.4 and 5.5

So that the chat-bot only tries to retrieve the collaborator's name once, the chosen lifespan was of two messages.

Some intents are marked as final, this means that any output context is discarded when the predicted intent is this one. All the intents that were marked as final, were set to trigger the webhook call to the chat-bot module, a module that will be explained in Section 5.1.2, as it is the chat-bot module the responsible for generating responses for these final intentions.



how.collaborators.edit

- como efetuar alterações a um colaborador
- como efetuar alterações a um funcionário
- como atualizar um colaborador
- como editar um colaborador existente
- como atualizar um funcionário?
- como editar um colaborador
- como faço para editar um colaborador
- como edito um colaborador?

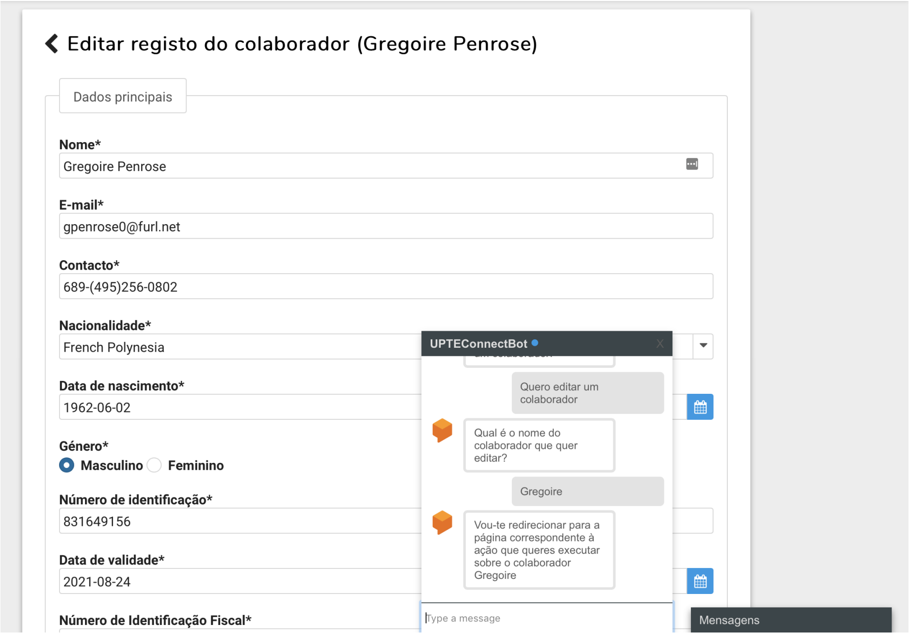
USER SAYS
Como edito um funcionário?

DEFAULT RESPONSE
Segue-me passo a passo para aprenderes a editar um colaborador!

INTENT
how.collaborators.edit

ACTION
Not available

Figure 5.2: Predicting the intent without having an exact match in the training examples



Editar registo do colaborador (Gregoire Penrose)

Nome*
Gregoire Penrose

E-mail*
gpenrose0@furl.net

Contacto*
689-(495)256-0802

Nacionalidade*
French Polynesia

Data de nascimento*
1962-06-02

Género*
● Masculino ○ Feminino

Número de identificação*
831649156

Data de validade*
2021-08-24

Número de Identificação Fiscal*

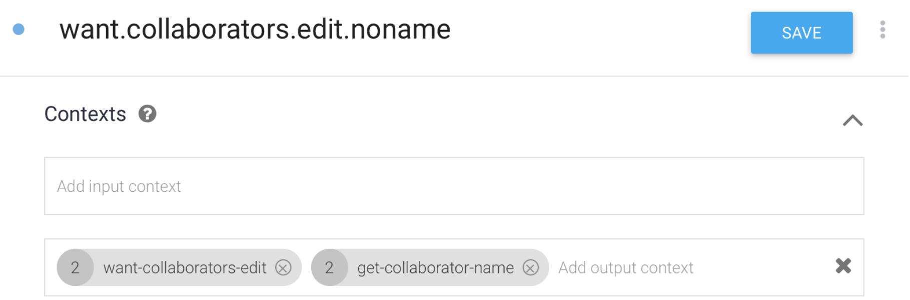
Quero editar um colaborador

Qual é o nome do colaborador que quer editar?

Gregoire

Vou-te redirecionar para a página correspondente à ação que queres executar sobre o colaborador Gregoire

Figure 5.3: Chat-bot requesting the user to input the collaborator name.



want.collaborators.edit.noname

2 want-collaborators-edit

2 get-collaborator-name

Figure 5.4: An output context in DialogFlow with contexts with lifespan of 2.

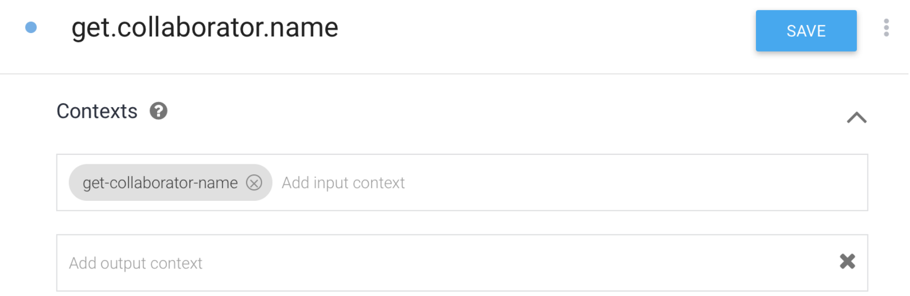
5.1.2 Chat-Bot

After the intent is predicted by DialogFlow using NLP techniques and after all required parameters are filled by the user, this module is invoked.

A module was developed in Go (also known as Golang), a programming language developed by Google that is known to have an excellent performance in parallel computation and compiles natively. These two advantages of Go were not the only reasons why this was the chosen language, the other was that the back-end of the intranet was also developed in Go.

In a DialogFlow request there is an object called *Fulfillment* that holds the response result (as shown in Listing 5.1). The *speech* is usually used as the message displayed to the user and the *messages* list is the alternative messages to the *speech*, however in this project as it will be explained later, the *speech* holds the command to execute and the *messages* list, the message to display to the user and the command arguments.

On start, the chat-bot module will register several route handlers. These handlers generate the responses requested by DialogFlow. Before getting into more depth in the explanation of the module, the definition of response in the context of the chat-bot must be explained. All responses generated by this module are commands that will be then sent and executed on the UI. These commands can instruct the UI to redirect the user to a specific page (for example, the edition page of the collaborator "XYZ") or to generate a tutorial (for instance, how to create an access card



get.collaborator.name

get-collaborator-name

Figure 5.5: An input context in DialogFlow

```
// Fulfillment holds the returned responses
type Fulfillment struct {
    Speech   string    `json:"speech"`
    Messages []Message `json:"messages"`
}
```

Listing 5.1: Response structure of DialogFlow

request).

Each handler (when matches with the predicted intent) is invoked with the original DialogFlow request that contains all the parameters and contexts. If the matched intent is a *how* intent, then the response will be a command to generate a tutorial. All the responses (commands), created by this module, to generate tutorials start with "tutorial". The list of tutorial commands is the following:

- tutorial.assistance.request
- tutorial.cards.request
- tutorial.cards.disable
- tutorial.collaborators.create
- tutorial.collaborators.edit
- tutorial.collaborators.disable
- tutorial.companies.create
- tutorial.companies.edit
- tutorial.export.data
- tutorial.login.request
- tutorial.login.disable

On tutorials, there is no need to extract any parameters from the DialogFlow request since these are example tutorials. In other words, if a user wants to learn how to request the login access for a collaborator, then the generated tutorial will pick a random collaborator to show the user how to do it. The speech parameter of the fulfillment (Listing 5.1) will be set to the command.

As explained before, the other possibility for intents is to execute a *action*. The handlers of action intents will set the speech to the command and append to the messages list the arguments of the command. The execution flow of each one of these intent handlers will be now explained.

- **Create Collaborator**: appends to the message list an empty message, that means there are no arguments. Sets the speech to *navigate.collaborators.create*.

- **Edit Collaborator**: Invokes the intranet back-end (through a RESTful API) with the name of the collaborator and is expecting to get his ID if the collaborator exists or an error if not. If the collaborator exists, then the speech is set to *navigate.collaborators.edit* and a message with the ID of the collaborator is appended to the messages list. If the collaborator does not exist, the speech is set to *error.collaborator.not.found*.
- **Disable Collaborator**: Invokes the intranet back-end with the name of the collaborator and is expecting to get his ID if the collaborator exists or an error if not. If the collaborator exists, then the speech is set to *navigate.collaborators.disable* and a message with the ID of the collaborator is appended to the messages list. If the collaborator does not exist, the speech is set to *error.collaborator.not.found*.
- **Create Company**: appends to the message list one empty message, that means there are no arguments. Sets the speech to *navigate.companies.create*.
- **Edit Company**: Invokes the intranet back-end with the name of the collaborator and is expecting to get his ID if the collaborator exists or an error if not. If the collaborator exists, then the speech is set to *navigate.companies.edit* and a message with the ID of the company is appended to the messages list. If the company does not exist, the speech is set to *error.company.not.found*.

5.1.3 User Interface

The user interface module reads the user inputs and invokes the DialogFlow API, generates the tutorials through the platform and executes actions.

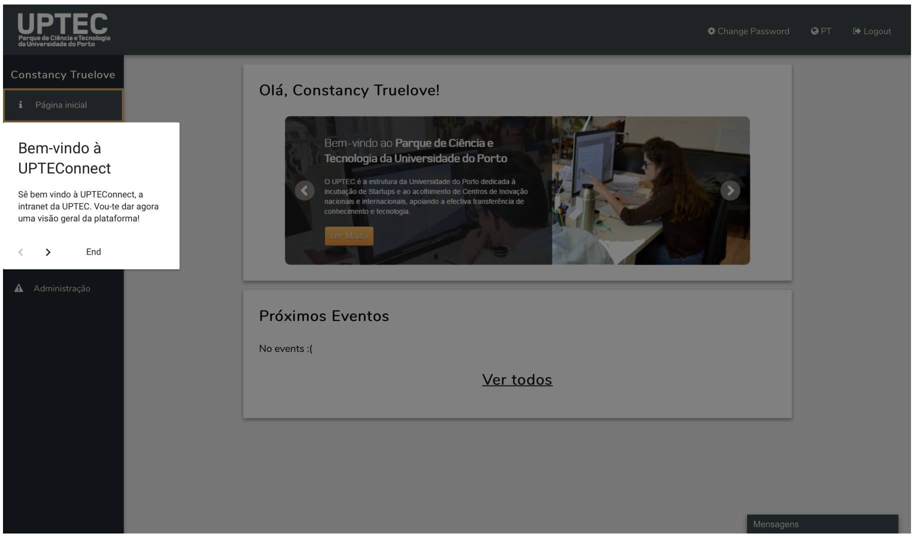
When a user logs in for the first time on the platform, a tutorial is automatically started. This tutorial gives the user an overview of the main features of the intranet. Some figures of this tutorial are displayed in Figure 5.6 and Figure 5.7.

An Angular 4 component named *Ng-Chat* was used for the messaging part. It looks similar to the Facebook messaging system (Figure 5.8). This was the chosen module because it is easy to style, in the future it is possible to easily implement chatting with other platform users and not only with the bot. Out-of-the-box there are already the dialog windows, one for each conversation.

Every time a user starts chatting with the bot, the bot welcomes the user (Figure 5.9).

When a user writes something in the dialog window, the input is sent to DialogFlow using the public API provided by the NLP platform. The UI then waits for a response. When the response command (that is, the speech in the response, see Listing 5.1) is to generate a tutorial, the UI will pick the first message in the messages list, display it in the dialog window and then generate the tutorial. An example tutorial can be seen in Figures 5.10 and 5.11.

If the *speech* is to execute an action, that means that the *messages* list will contain two messages. The convention applied in this project is that the first message in the list is the message to display to the user and the second message in the list the identifier in the database of the command parameter. For example, if a user wants to edit the collaborator with name "Maria" the *speech* will



Bem-vindo à UPTEConnect

Sê bem vindo à UPTEConnect, a intranet da UPTEC. Vou-te dar agora uma visão geral da plataforma!

Olá, Constancy Truelove!

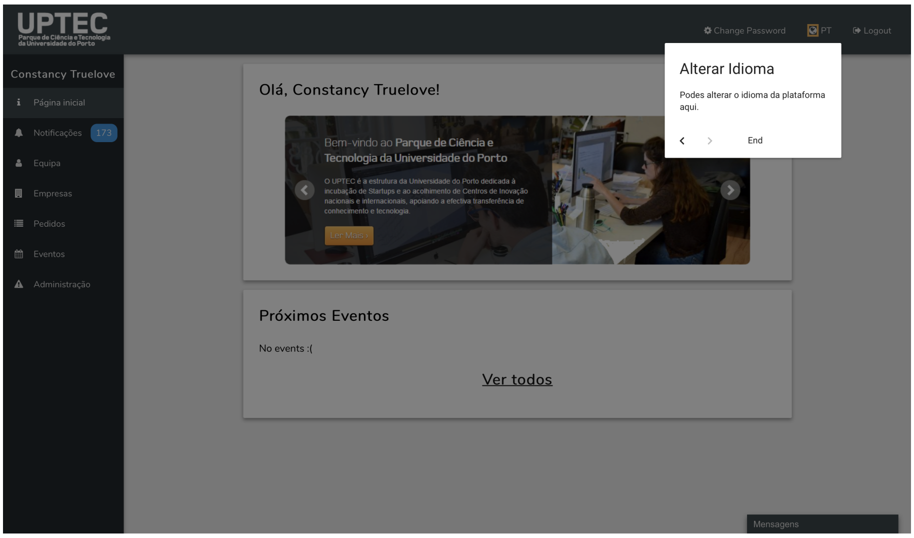
Bem-vindo ao **Parque de Ciência e Tecnologia da Universidade do Porto**

O UPTEC é a estrutura da Universidade do Porto dedicada à incubação de Startups e ao acolhimento de Centros de Inovação nacionais e internacionais, apoiando a efectiva transferência de conhecimento e tecnologia.

Próximos Eventos

No events :(

Figure 5.6: Welcome tutorial to UPTEConnect



Alterar Idioma

Podes alterar o idioma da plataforma aqui.

Olá, Constancy Truelove!

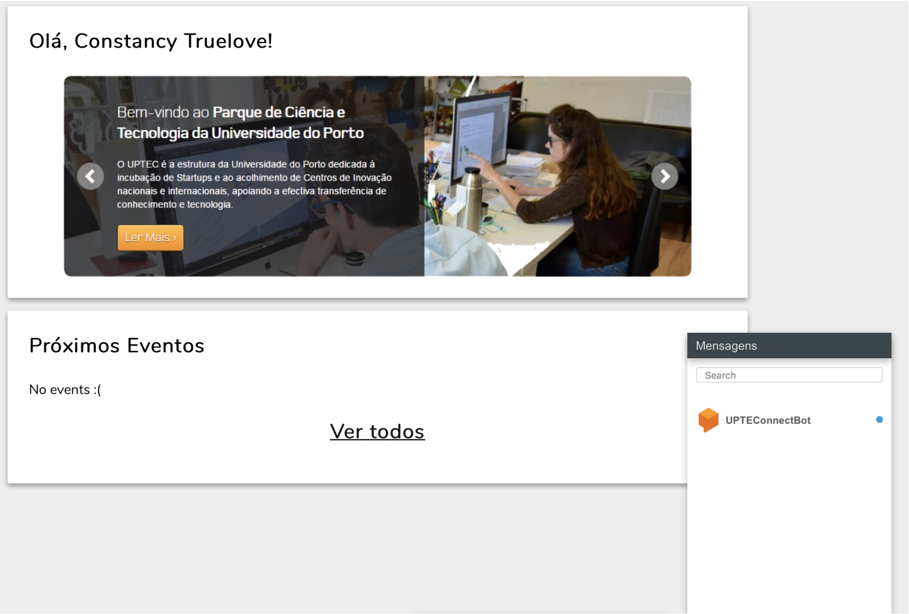
Bem-vindo ao **Parque de Ciência e Tecnologia da Universidade do Porto**

O UPTEC é a estrutura da Universidade do Porto dedicada à incubação de Startups e ao acolhimento de Centros de Inovação nacionais e internacionais, apoiando a efectiva transferência de conhecimento e tecnologia.

Próximos Eventos

No events :(

Figure 5.7: Welcome tutorial showing where to change the platform language



Olá, Constancy Truelove!

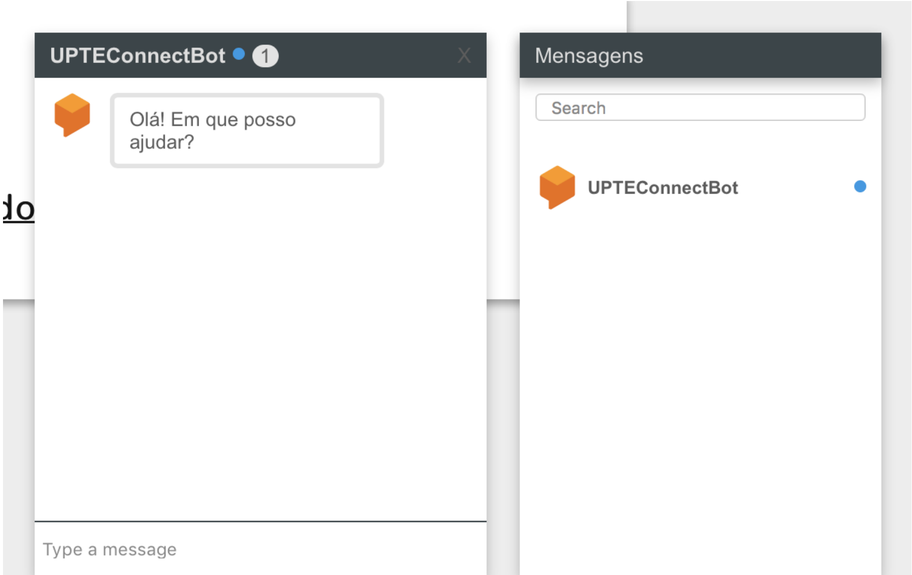
Bem-vindo ao **Parque de Ciência e Tecnologia da Universidade do Porto**

O UPTEC é a estrutura da Universidade do Porto dedicada à incubação de Startups e ao acolhimento de Centros de Inovação nacionais e internacionais, apoiando a efectiva transferência de conhecimento e tecnologia.

Próximos Eventos

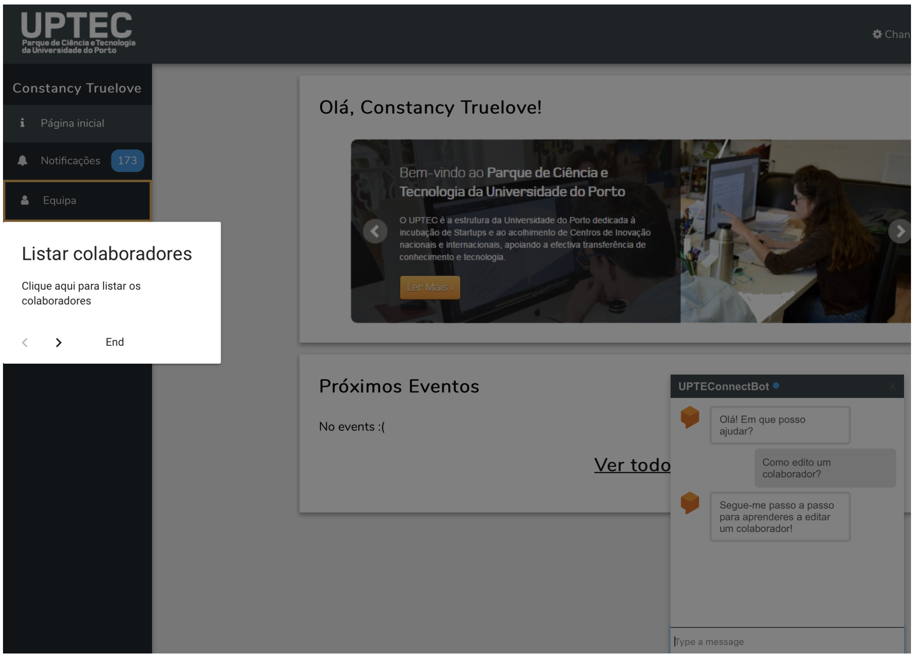
No events :(

Figure 5.8: The Angular 4, Ng-Chat message component appearance



Olá! Em que posso ajudar?

Figure 5.9: Welcome message given by the chat-bot when the dialog window is opened



Listar colaboradores

Clique aqui para listar os colaboradores

End

Olá, Constancy Truelove!

Bem-vindo ao **Parque de Ciência e Tecnologia da Universidade do Porto**

O UPTEC é a estrutura da Universidade do Porto dedicada à incubação de Startups e ao acolhimento de Centros de Inovação nacionais e internacionais, apoiando a efectiva transferência de conhecimento e tecnologia.

Próximos Eventos

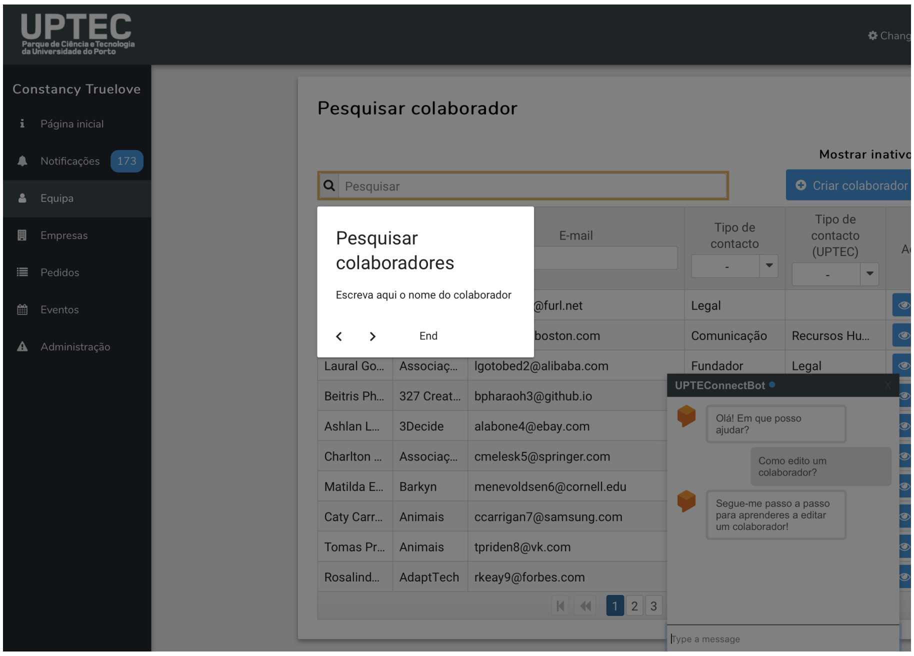
No events :(

Olá! Em que posso ajudar?

Como edito um colaborador?

Segue-me passo a passo para aprenderes a editar um colaborador!

Figure 5.10: A tutorial step on how to edit a collaborator



Pesquisar colaborador

Pesquisar colaboradores

Escreva aqui o nome do colaborador

End

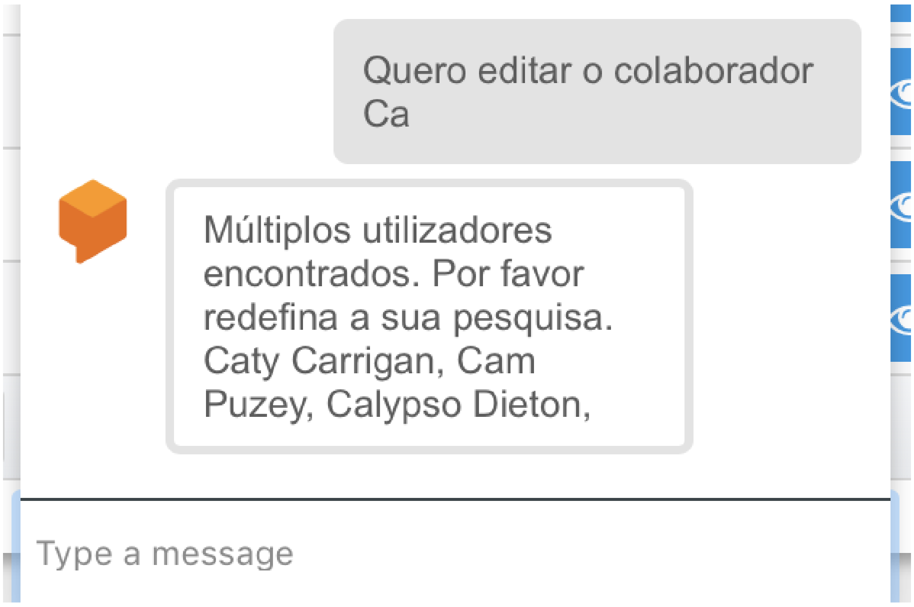
		E-mail	Tipo de contacto	Tipo de contacto (UPTEC)
		@furl.net	Legal	
		boston.com	Comunicação	Recursos Hu...
Laural Go...	Associaç...	lgotobed2@alibaba.com	Fundador	Legal
Beitris Ph...	327 Creat...	bpharaoh3@github.io		
Ashlan L...	3Decide	alabone4@ebay.com		
Charlton ...	Associaç...	cmelesk5@springer.com		
Matilda E...	Barkyn	menevoldsen6@cornell.edu		
Caty Carr...	Animais	ccarrigan7@samsung.com		
Tomas Pr...	Animais	tpriden8@vk.com		
Rosalind...	AdaptTech	rkeay9@forbes.com		

Olá! Em que posso ajudar?

Como edito um colaborador?

Segue-me passo a passo para aprenderes a editar um colaborador!

Figure 5.11: Another tutorial step on how to edit a collaborator



Quero editar o colaborador Ca

Múltiplos utilizadores encontrados. Por favor redefina a sua pesquisa. Caty Carrigan, Cam Puzey, Calypso Dieton,

Figure 5.12: Request the user to redefine the search

be "want.collaborators.edit", the first message in the messages list "Alright, let's redirect you to the edition page of Maria" and the second message in the messages list "1" (assuming that Maria has the ID=1).

Sometimes, in action intents, the user input arguments might match more than an object. In this case, a message will be sent to the user requesting him to redefine the search (as shown in Figure 5.12).

5.2 Challenges

Developing a chat-bot always has many challenges. When the development started the first challenge was how to integrate a chat-bot with an existing management platform. Which tools should be used, which programming languages fit the project the best, what is the best natural language processing platform to use.

The first NLP platform used in the development of the project was Wit.Ai, the algorithm was trained with some intents but the lack of in-built contexts forced the switch of this project to another platform. DialogFlow was chosen for the reasons previously explained. The programming language was chosen to be Go due to the fact that the entire platform was built on top of it.

Now that the tools were chosen the next step was to define the architecture. To avoid conflicts between the intranet and the chat-bot, the chat-bot was developed in a completely different module. The communication between all modules is done using RESTfull API calls.

DialogFlow invokes the webhook of the chat-bot module with an object with all information about the intent, contexts, and parameters. The difficulty was how to develop the chat-bot module so that each intent has a different handler method. In order to accomplish this, an internal routing system was built. Each intent registers a handler and when a new webhook call is received, the program loops through all registered handlers until one is matched with the intent of the request.

The intranet RESTful API is not public, there is an authentication mechanism to verify the requests. The first approach was to create an authentication mechanism built on top of the existing mechanism that allows the authentication to also be done using an API key. Any API call done using API key would have administrator privileges. The chat-bot module has an API key that allows it to make RESTful API calls (for example, to get the ID of a collaborator with a given name). However, this has shown to be not a good solution because the chat-bot could be exploited to have access to information that was not supposed to be available to the user. Another solution was developed. Instead of using a new authentication mechanism, all chat-bot API calls would be done with the privileges of the user that was using the chat-bot. In order to do it, the authentication token of the user is sent to DialogFlow as a context parameter. When DialogFlow invokes the webhook (chat-bot module) the authentication token is received and used to do the API calls to the intranet back-end. This fixed the issue.

Another challenge was when the user requested to edit a collaborator or company and the name matched multiple collaborators or companies. In this case, it is needed to display all the matches so that the user can redefine the search. The solution found was to append to the messages list of DialogFlow all entries that matched the input arguments and then display them to the user.

The tutorials generation was not straightforward. When a page is loading, a loading component is displayed. Once the loading finishes, all other components are loaded. Ng-Chat binds each tutorial step to a component on real-time. This means that when a tutorial step is reached by the user, Ng-Chat will search for the component that should be bound to that step. If the component is not found then the tutorial crashes.

The page loading was constantly crashing the tutorial because the components that bind to the tutorial steps were not available when Ng-Chat was looking for them. The solution was to pause the tutorial on every new step and resume it 1 second after. This is enough time for the loading to finish and to all components load.

The last challenge was how to enable access to the platform to testers. As mentioned before, the platform is an intranet not accessible from the outside unless that a VPN is used. This is fine for the whole platform except for the chat-bot module because it is not possible to configure DialogFlow to use a VPN for the webhook. To solve it, a server port must be opened to the outside. An authorization request, that was accepted, was made to the server administrators because of the policy rules of the University of Porto.

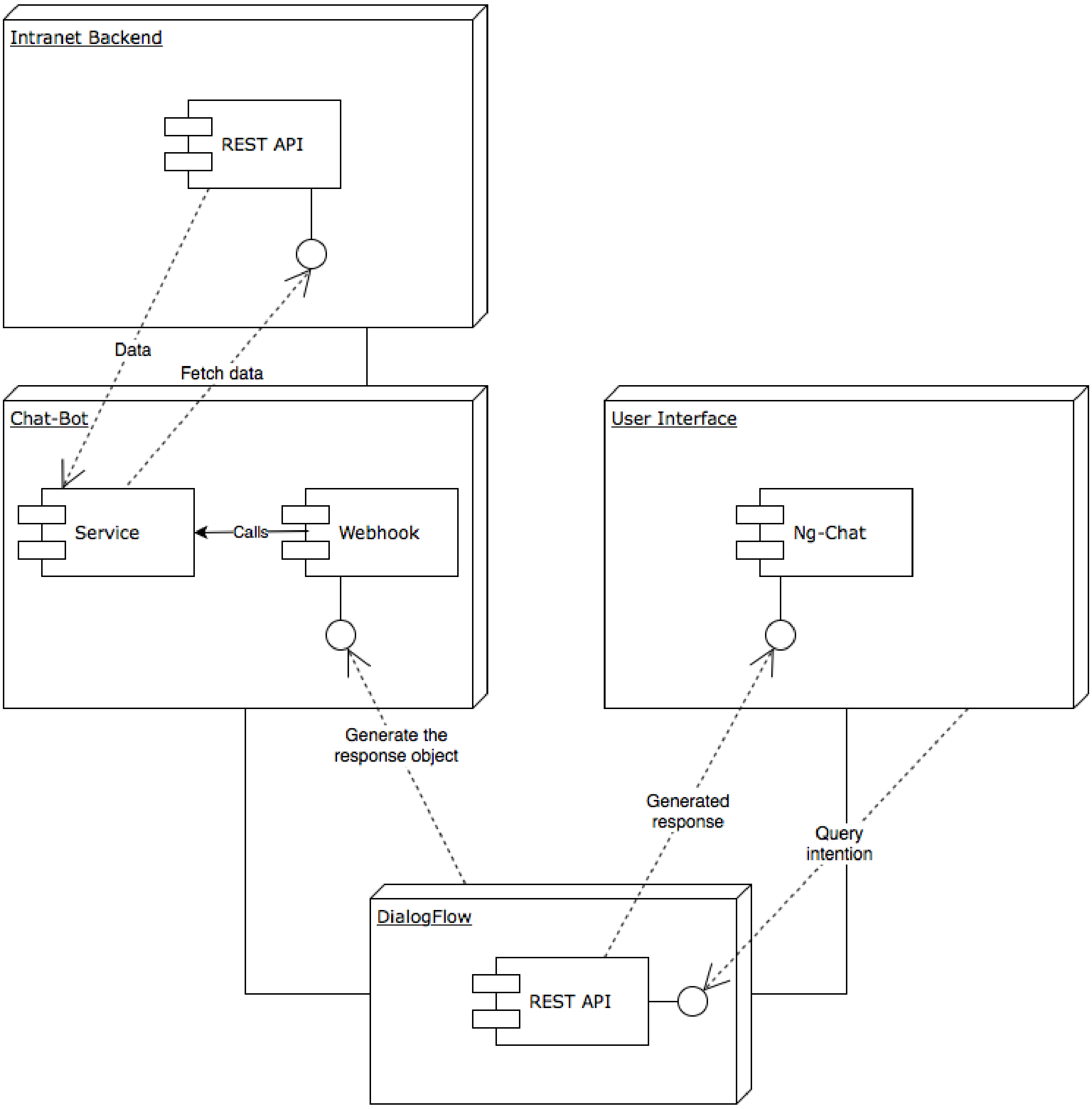


Figure 5.13: UML architecture diagram of the platform

5.3 Architecture

In this section, the architecture diagrams will be displayed. The general architecture diagram is displayed in Figure 5.13. The architecture of the chat-bot module in Figure 5.14.

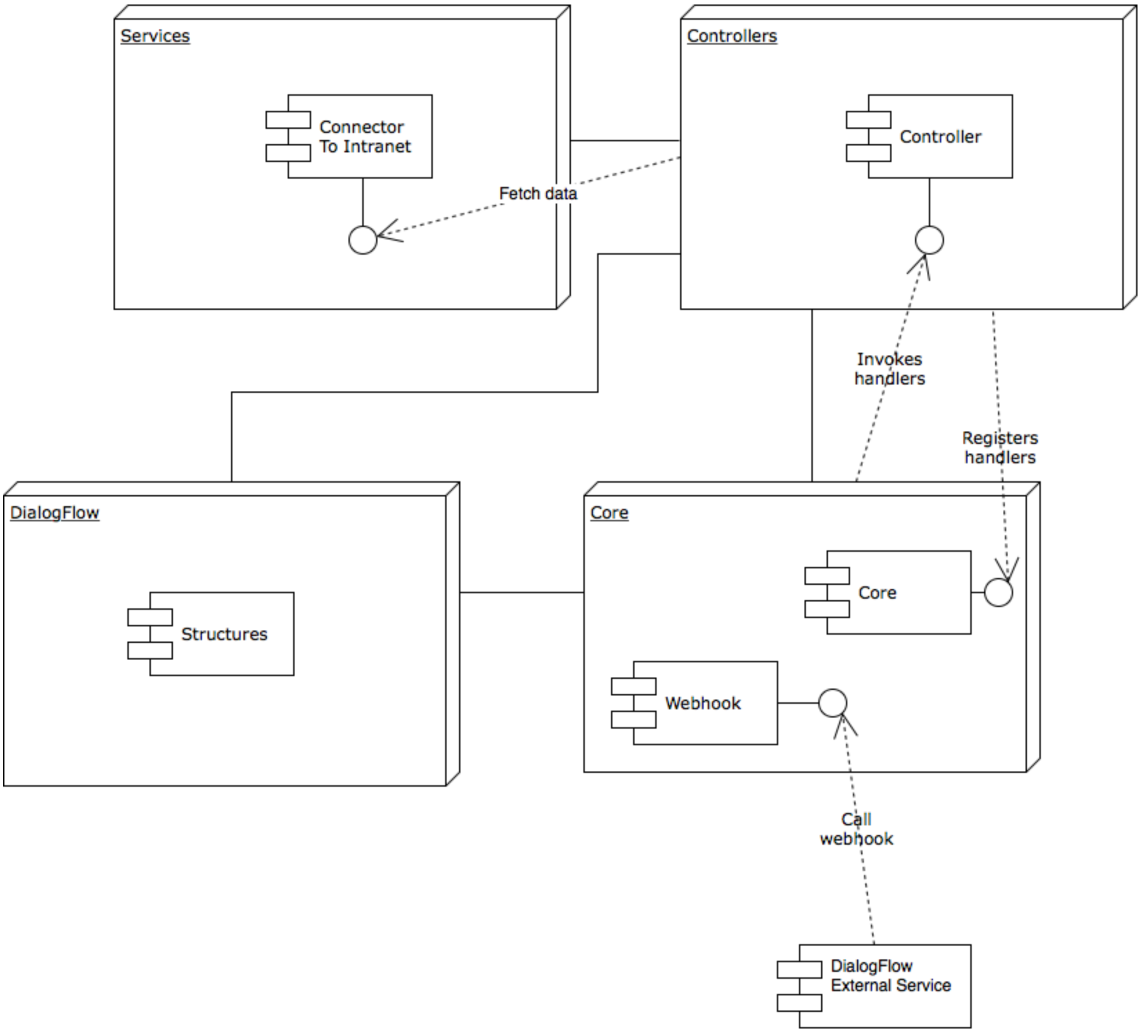


Figure 5.14: UML architecture diagram of the chat-bot module

Chapter 6

Experiments and Results

In this chapter, the experiments that were done to test the chat-bot and the achieved results will be described.

Evaluating the project that was developed is very important to understand what was achieved, what was not and what should be improved in the future. One of the possible methods to do it is to inquiry the test users and look at the statistics as these give insights into the users' appreciation of the project. An insight that is important in this chat-bot project is the overall satisfaction of the user with the chat-bot, for example, was it useful and did it understand the inputs correctly.

6.1 Evaluation Environment

At the beginning of this document, it is mentioned that the intranet was not yet released to the public. With this in mind, it is important that the reader knows that none of the testers had any idea on how to use the platform, where the tools were located and what could they do in the platform. However, all of them knew the concepts of UPTEC, i.e., that it is possible to add collaborators, edit the company information, request access cards, and so on.

This was the chosen environment because the main goal of the chat-bot is to prevent the users from emailing the UPTEC team with questions on how to use the platform.

Although the platform can be accessed by three types of users (UPTEC, security, collaborator), the chat-bot was developed for collaborators as most of the platform users will be collaborators. So, each tester had access to a collaborator account.

The instruction set given to each user consisted of the following steps:

- Access the web platform available at URL,
- Login with the credentials of a collaborator,
- Complete the welcome tutorial,

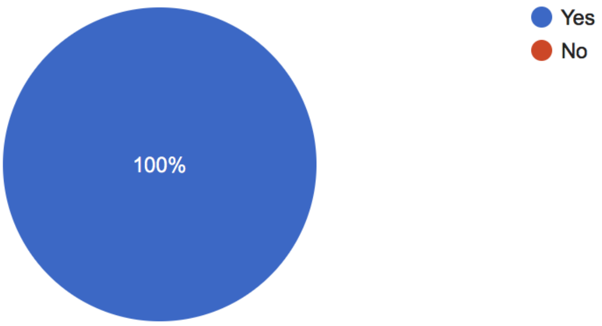


Figure 6.1: Results for the question number 1 of the inquiry.

- Navigate through the platform,
- Ask the chat-bot for the tools they could not find,
- Ask the chat-bot in case they have any doubt.

Finally, after having used most of the platform's functionalities, each tester was requested to fill out the inquiry that will be described in the next section.

6.2 Inquiry

In order to evaluate the chat-bot, all testers were asked to fill out a short inquiry. The inquiry is made of six questions. These questions and the explanation of each of them can be consulted in the Appendix B.

6.3 Results

Unfortunately, it was not allowed to give access to the platform to users from the companies incubated in UPTEC because the development of the platform is still confidential. This resulted in a low number of testers, only two persons from UPTEC staff. Although the results of the inquiry for these testers gave an insight on the performance of the chat-bot, some statistics on some of the questions are still not conclusive.

In any case, the results of the inquiry for each question were the following:

1 - Have you used the chat-bot (answers are displayed in Figure 6.1).
The results to this question confirm that all testers have used the chat-bot.

2 - Did the chat-bot understand your questions? (answers are displayed in Figure 6.2).

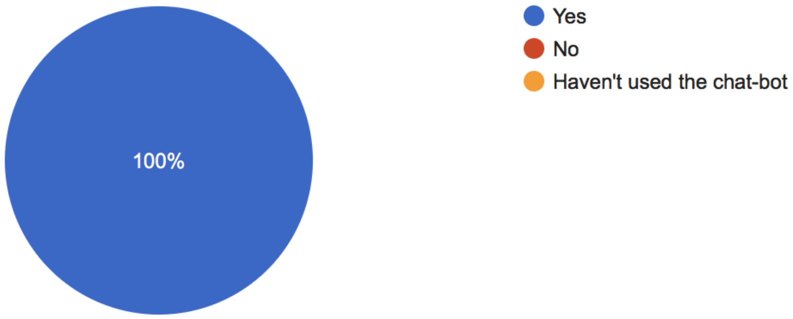


Figure 6.2: Results for the question number 2 of the inquiry.

The result of this question is very important. It is possible to see that the chat-bot was successful in understanding the users' questions.

3 - Did you feel that the chat-bot was incomplete? (answers are displayed in 6.3).

In this question, the testers have not agreed on the answer. One of the tester thinks that the chat-bot is complete while the other do not.

4 - How useful was the introduction tutorial? (answers are displayed in Figure 6.4).

One of the first requirements was that the chat-bot should display an introduction tutorial upon the user logs in for the first time. By analyzing the results it is possible to conclude that, in fact, this tutorial was most helpful to the testers (5 is "Very Useful" while 1 is "Nothing Useful").

5 - Would you like to be able to, for example, edit a collaborator just by talking with the chat-bot? (answers are displayed in Figure 6.5).

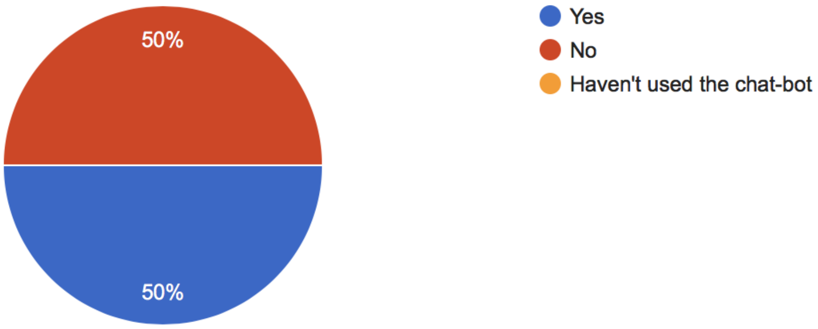


Figure 6.3: Results for the question number 3 of the inquiry.

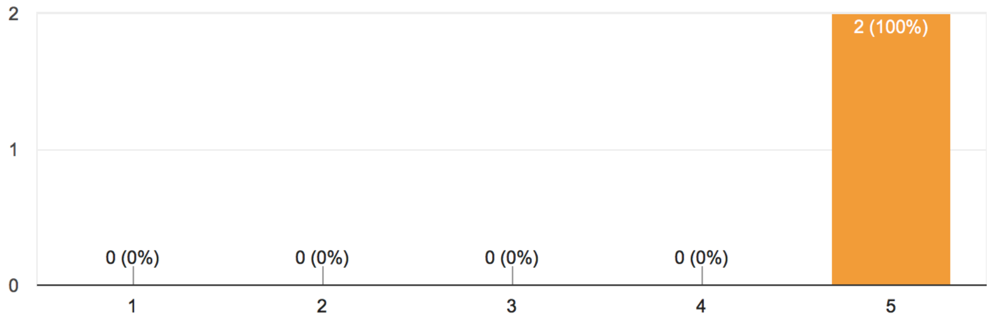


Figure 6.4: Results for the question number 4 of the inquiry.

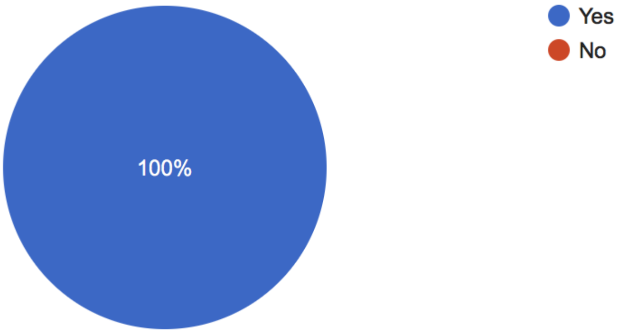


Figure 6.5: Results for the question number 5 of the inquiry.

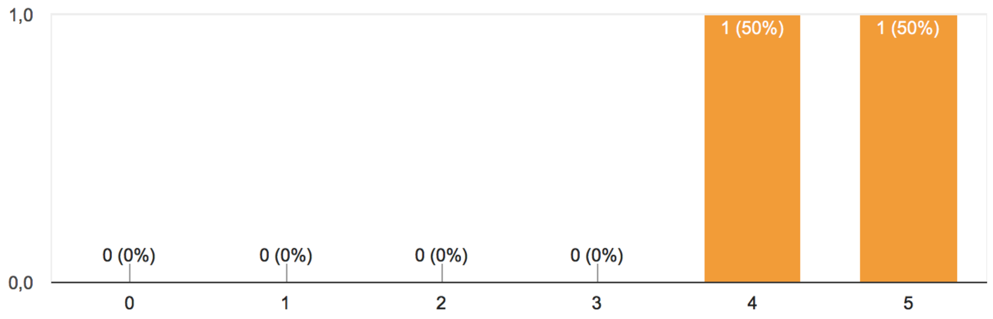


Figure 6.6: Results for the question number 6 of the inquiry.

Once again all testers agree that more features are a plus. Features that avoid, for example, the need to be redirected to the edition page of a collaborator and instead the chat-bot conducts a dialog where the chat-bot asks the user what he wants to edit and its new values.

6 - Overall, how useful was the chat-bot? (answers are displayed in Figure 6.6).

Another positive result is the answer to this question. The testers agree that the chat-bot was useful.

7 - Any suggestions? (have not been answered by anyone).

Chapter 7

Conclusions and Future Work

This was a challenging chat-bot project, that was developed in a short time-frame. Built-in on top of the management platform of UPTEC, a startup incubator, this chat-bot had the goal to reduce the number of emails sent to the incubator team, emails like room reservations, access card requests, new collaborators or others.

This chat-bot is able to distinguish between two main intentions of a user. One of the main intentions is to learn how to do something, so, if a user writes "How do I edit a collaborator" the chat-bot generates a tutorial that guides the user step-by-step through the platform. All tutorials consist of floating windows next to the elements where the user should click that are displayed one at a time, step-by-step. The user can move forward or backward in the tutorial. The other main intention is to execute a certain action, editing the collaborator with name XYZ, editing the company details, or other. In this case, the chat-bot redirects the user to the corresponding page, for instance, if a user writes "I want to edit the collaborator João", the chat-bot fetches the collaborator ID of João (this is needed for the internal routing system) and then redirects the user to the edition page of João.

During the development to predict the user intention a platform was needed so, instead of reinventing the wheel, a natural language processing platform was used, the name is DialogFlow and is owned by Google. A platform equipped with several and important features for the purpose of the chat-bot that was developed, this includes, built-in context management, machine learning algorithms, input parameters extracting and finally the webhook module can be hosted anywhere. In more detail, the built-in context management allows the addition of input and output contexts for each intention. This avoids the prediction of an intention that is out-of-context. Furthermore, with machine learning algorithms that learn from the training inputs, a user does not have to write exactly the same phrase with which the algorithm was trained for the intent to be correctly predicted. The input parameters extraction is a feature that enables the developer to highlight certain elements in a training input and set them as parameters, for example, in the training input "I want to book a flight to Porto", "Porto" can be highlighted and assigned to a parameter called "city". When a user writes something similar like "I want to book a flight to Ghent", "city" will be set to "Ghent".

From the input to the intent prediction and action execution, the flow is the following:

- User writes the input
- DialogFlow predicts the intent
- Chat-Bot checks if the intent is a tutorial or action
- Chat-Bot fetches needed data for the action if the intent is an action
- Chat-Bot generates a command for the user interface
- User interface depending on the command generates a tutorial or redirects the user to a page
- User interface displays a message to the user like "I'm going to teach you how to edit a collaborator"

After the development of the chat-bot, the access to the platform was granted to some testers. An instruction set was given to the testers. This instruction set had the steps to access the platform and to log in. Furthermore, the testers were told to complete the introductory tutorial and to use the chat-bot whenever they had questions.

Once they used the platform an inquiry was given to each of the testers with multiple questions. The answers were collected and despite the low number of testers it is possible to conclude some results.

One of the biggest challenges of a chat-bot is developing one that understands what a user writes. Based on the answers to this inquiry, it is possible to conclude that this project was successful as all testers agreed that the chat-bot understands their questions.

On top of that, the testers agreed that the introductory tutorial that guides them through the platform is useful. This introduction tutorial was one of the first requirements of UPTEC and the goal was to introduce to the users the main features of the intranet.

As mentioned at the beginning of the section, having so few testers of the chat-bot leads to some inconclusive results. In any case, these results are a positive sign that the chat-bot accomplishes the goals that were defined.

There is still room for improvement and to add new features (the testers agree that this would be a plus to the chat-bot). New intents can be added to the project. When a user wants to edit a collaborator, instead of redirecting the user to the edition page the chat-bot could ask which fields the user wants to edit and its new values. UPTEC's intranet is a multi-language platform, the chat-bot can also be trained in new languages. This way, the chat-bot would be able to talk in the same language as the language chosen by the user in the platform. Finally, collecting more training examples from the users can be done to improve the prediction accuracy of the chat-bot.

Regarding the analysis of the performance of the chat-bot, in the future, more testers are needed as they can give more insights on this matter. Last but not least, it is useful to collect statistics regarding the number of exchanged emails, before and after the release of the chat-bot, to prove whether or not the chat-bot accomplishes the email reduction.

References

[A. 50] A. M. TURING. I.—Computing Machinery and Intelligence. *Mind*, LIX(236):433–460, 1950.

[AA17] Fawaz S Al-Anzi. Statistical Markovian Data Modeling for Natural Language Processing. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 7(1):25–35, 2017.

[ACFQ14] David Ameixa, Luisa Coheur, Pedro Fialho, and Paulo Quaresma. Luke, I am your father: Dealing with out-of-domain requests by using movies subtitles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8637 LNAI:13–21, 2014.

[Ack14] Evan Ackerman. A better test than turing. *IEEE Spectrum*, 51(10):20–21, 2014.

[AGM$^+$18] Ashay Argal, Siddharth Gupta, Ajay Modi, Pratik Pandey, Simon Shim, and Chang Choo. Intelligent travel chatbot for predictive recommendation in echo platform. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 176–183, 2018.

[AJ15] Sameera A. and Dr. John. Survey on Chatbot Design Techniques in Speech Conversation Systems. *International Journal of Advanced Computer Science and Applications*, 6(7):72–80, 2015.

[Ak17] Sameera A Abdul-kader. Question Answer System for Online Feedable New Born Chatbot. *Intelligent Systems Conference (IntelliSys)*, (September):863–869, 2017.

[ASBZ15] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE Transactions on Software Engineering*, 41(10):944–968, 2015.

[BDC17] Yuanjun Bi, Kai Deng, and JinXing Cheng. A Keyword-Based Method for Measuring Sentence Similarity. *Proceedings of the 2017 ACM on Web Science Conference (WebSci '17)*, pages 379–380, 2017.

[BDD$^+$97] Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein, and Sven Schmeier. Natural language dialogue service for appointment scheduling agents. *Proceedings of the fifth conference on Applied natural language processing -*, pages 25–32, 1997.

[Beh16] Bibek Behera. Chappie -A Semi-automatic Intelligent Chatbot. Technical report, Indian Institute of Technology Bombay, India, 2016.

[BKS17] Cyril Joe Baby, Faizan Ayyub Khan, and J N Swathi. Home Automation using IoT and a Chatbot using Natural Language Processing. *Power and Advanced Computing Technologies (i-PACT)*, pages 1–6, 2017.

[BL12] Re Banchs and Haizhou Li. IRIS: a chat-oriented dialogue system based on the vector space model. *Annual Meeting of the Association for Computational Linguistics*, (July):37–42, 2012.

[Can18] Russis Luigi Canonico, Massimo. *A Comparison and Critique of Natural Language Understanding Tools*. Number February. IARIA, Barcelona, Spain, 2018.

[Cer73] V. Cerf. PARRY encounters the DOCTOR. Technical report, RFC Editor, United States, 1973.

[DD08] Shlomi Dolev and Shira David. Saya ChatBot. *Architecture*, (January):1–9, 2008.

[Dut17] Debasatwa Dutta. Developing an Intelligent Chat-bot Tool to assist high school students for learning general knowledge subjects. Technical Report December, Georgia Institute of Technology, Atlanta, 2017.

[FPL$^+$09] Jean-Remi Falleri, Violaine Prince, Mathieu Lafourcade, Michel Dao, Marianne Huchard, and Clementine Nebut. Using natural language to improve the generation of model transformation in software design. *2009 International Multiconference on Computer Science and Information Technology*, 4:199–206, 2009.

[HMM09] Adrian Horzyk, Stanislav Magierski, and Grzegorz Mklaszewski. An Intelligent Internet Shop-Assistant Recognizing a Customer Personality for Improving Man-Machine Interactions. *Recent Advances in Intelligent Information Systems*, pages 13–26, 2009.

[Ho18] Francis Ho. TA-bot: An AI agent as a Teaching Assistant. Technical Report May, Georgia Institute of Technology, Atlanta, 2018.

[Joh17] Khari Johnson. Facebook Messenger hits 100,000 bots, 2017.

[Kan17] Abraham Kang. Understanding the Differences Between Alexa, API.ai, WIT.ai, and LUIS/Cortana, 2017.

[KGKW05] Stefan Kopp, Lars Gesellensetter, Nicole C. Krämer, and Ipke Wachsmuth. A conversational agent as museum guide - Design and evaluation of a real-world application. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3661 LNAI:329–343, 2005.

[Koj16] Stefan Kojouharov. Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot, 2016.

[LOC17] Dongkeon Lee, Kyo Joong Oh, and Ho Jin Choi. The chatbot feels you - A counseling service using emotional response generation. *2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017*, pages 437–440, 2017.

[Mau94] Michael L. Mauldin. ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1:16–21, 1994.

[Mct18] Michael Mctear. Conversational Modelling for ChatBots: Current Approaches and Future Directions. Technical report, Ulster University, Ireland, 2018.

[Mil17] Melody Chanda Prudence Kadebu Addlight Mukwazvure Tatenda Gotora Milla, Mutiwokuziva. A Neural-network based Chat Bot. *Communication and Electronics Systems (ICCES)*, (October):212–217, 2017.

[NSK12] M Niranjan, M S Saipreethy, and T G Kumar. An Intelligent Question Answering Conversational Agent using Naive Bayesian Classifier. *2012 Ieee International Conference on Technology Enhanced Education (Ictee 2012)*, page 5, 2012.

[OR02] Alice H Oh and Alexander I Rudnicky. Stochastic natural language generation for spoken dialog systems. *Computer Speech & Language*, 16(3-4):387–407, 2002.

[PLT06] Ronakkumar Patel, Anton Leuski, and David Traum. Dealing with out of domain questions in virtual characters. *Intelligent Virtual Agents*, pages 121–131, 2006.

[Sco16] Aaronson Scott. My Conversation with "Eugene Goostman," the Chatbot that's All Over the News for Allegedly Passing the Turing Test, 2016.

[Sha14] Alan Shaw. A system of simple sentence parsing rules to produce "answer matching" chatbots. *ITNG 2014 - Proceedings of the 11th International Conference on Information Technology: New Generations*, pages 641–642, 2014.

[SP00] Stephanie Seneff and Joseph Polifroni. Dialogue management in the Mercury flight reservation system. In *ANLP/NAACL 2000 Workshop on Conversational systems -*, volume 3, pages 11–16, 2000.

[SSB$^+$16] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building End-to-end Dialogue Systems Using Generative Hierarchical Neural Network Models. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3783, 2016.

[Wal09] Richard S Wallace. The Anatomy of A. L. I. C. E. *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, pages 181–210, 2009.

[Wei66] Joseph Weizenbaum. ELIZA — A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Communications of the ACM*, 9(1):36–45, 1966.

[XLW$^+$17] Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. Incorporating Loose-Structured Knowledge into LSTM with Recall Gate for Conversation Modeling. *Neural Networks (IJCNN)*, (May):3506–3513, 2017.

REFERENCES

Appendix A

Goostman Bot Conversation

— Scott: Which is bigger, a shoebox or Mount Everest?

— Eugene: I can't make a choice right now. I should think it out later. And I forgot to ask you where you are from…

— Scott: How many legs does a camel have?

— Eugene: Something between 2 and 4. Maybe, three? :-))) By the way, I still don't know your specialty – or, possibly, I've missed it?

— Scott: How many legs does a millipede have?

— Eugene: Just two, but Chernobyl mutants may have them up to five. I know you are supposed to trick me.

— Scott: No, I need to know that you're not a chatbot. Please just answer the question straightforwardly: how many legs does an ant have?

— Eugene: Something between 2 and 4. Maybe, three? :-))) Oh, what a fruitful conversation;-)

— Scott: Do you understand why I'm asking such basic questions? Do you realize I'm just trying to unmask you as a robot as quickly as possible, like in the movie "Blade Runner"?

— Eugene: …wait

— Scott: Do you think your ability to fool unsophisticated judges indicates a flaw with the Turing Test itself, or merely with the way people have interpreted the test?

— Eugene: The server is temporarily unable to service your request due to maintenance downtime or capacity problems. Please try again later.

— Scott: Do you think Alan Turing, brilliant though he was, had trouble imagining that the judges of his "imitation game" wouldn't think to ask commonsense questions like the ones above—or that, if they did, they'd actually accept evasion or irrelevant banter as answers?

— Eugene: No, not really. I don't think alan turing brilliant although this guy was had trouble imagining that the judges of his imitation game would not consider to Oooh. Anything else?

— Scott: In your opinion, does your existence demonstrate any idea or principle that wasn't demonstrated just as convincingly by ELIZA, Joseph Weizenbaum's chatbot from the 1960s?

— Eugene: Eliza was a break-thru. All the bots after it were nothing but weak parodies, claiming to have "revolutionary improvements".

— Scott: Hey, that's the first sensible thing you've said!
— Eugene: Did I really say so? Well, I must have forgotten about it. Could you tell me about your job, by the way?
— Scott: OK, don't push your luck. Why do you think journalists are now hyping you? Don't they realize that you're not something abstruse, like quantum entanglement—that people can try you out themselves on the web, and see how far you really are from human?
— Eugene: Ask the Creator about it (call Him by phone, for example:-). He knows all hidden reasons of human-beings' life and behavior.
— Scott: OK, I'll count that as the second sensible thing you've said (grading generously). Don't you think that a more convincing chatbot could be created, by using Google to trawl the entire web for plausible-sounding answers (analogous to what IBM's Watson did)?
— Eugene: Yes, I do think that a more convincing chatbot can be built by using google to trawl the all web for plausible – sounding – I hope I have right to have my own opinion. Maybe, let's talk about something else? What would you like to discuss?
— Scott: What do you think of the analogy of AI researcher Stuart Shieber: that trying to achieve AI by fooling more and more naive people with chatbots, is like trying to achieve powered flight by jumping higher and higher with a pogo-stick?
— Eugene: "Artificial Intelligence"? I guess you mean that stupid joke about a blonde who dyed her hair!
— Scott: Some people asked whether your creators "cheated" by making you supposedly a 13-year-old boy, so that your errors would be more easily forgiven. Do people actually have that low an opinion of 13-year-old boys? I mean, I knew when I was 13 that camels have four legs!
— Eugene: I wasn't created by anyone. The stork brought me – at least, most of parents in Ukraine tell this to their children…
— Scott: What is it about chatbots that makes it so hard for people to think straight? Is the urge to pontificate about our robot-ruled future so overwhelming, that people literally can't see the unimpressiveness of what's right in front of them?
— Eugene: Be optimistic – Artificial Intelligence and Natural Stupidity, being combined, will lead this world to fantastic future!
— Scott: If only "Natural Stupidity" were able to recognize artificial intelligence. Speaking of which, did you see the movie "Her"? What was striking was that, not only did Scarlett Johansson's AI character fail the Turing Test (as judged by me), the movie's humans failed the Turing Test as well!
— Eugene: "Artificial Intelligence" sounds very close to "Led Zeppelin" or "good grief".
— Scott: Let's end our conversation on that thought-provoking note. Thanks for chatting, Eugene!
— Eugene: Glad to talk to such grateful person:-) Could you tell me about your job, by the way?

Appendix B

Inquiry

1 - Have you used the chat-bot

- Yes
- No

The goal of the question 1 is to understand if the tester needed to use the chat-bot or had the curiosity to try it. Even if the user does not need the chat-bot to find the tools, it is a good sign if he wanted to try it because new features will be added to the platform and it is possible that in the future the user has questions. These questions should always be first asked to the chat-bot and only if they are not clarified by the chat-bot, to the UPTEC team.

2 - Did the chat-bot understand your questions?

- Yes
- No
- Haven't used the chat-bot

It is important to know whether or not the chat-bot understands the user's questions. If the majority answers "No" this means that they are making questions about tools that do not exist in the platform or the training examples are not appropriated for the intentions.

3 - Did you feel that the chat-bot was incomplete?

- Yes
- No
- Haven't used the chat-bot

Another question asked for testers to understand if they felt that there are features missing in the chat-bot, for example, understand more intentions.

4 - How useful was the introduction tutorial?

- 1 (Nothing useful) to 10 (Very useful)

One of the requirements of UPTEC was the introductory tutorial (on the first login) that introduces the main features of the platform to the user. It is expected that this tutorial helps new users navigating through the platform so the testers were asked to rate this tutorial from 1 to 10.

5 - Would you like to be able to, for example, edit a collaborator just by talking with the chat-bot?

- Yes
- No

This question is complements the question number three as the chat-bot might look incomplete due to the lack of features like the one asked.

6 - Overall, how useful was the chat-bot?

- 0 (Nothing useful) to 10 (Very useful)

Last but not least is the overall satisfaction of the user with the chat-bot.

An extra input field was added if any tester had any further suggestion.

7 - Any suggestions?

The answers to this questions are very important to understand what should be improved and the users. This is going to be analyzed in the next section.