Masters in Informatics Engineering
Internship
Final Report

Software Platform to support chat assistants and smart replies using NLP

Mariana Leal
mleal@student.dei.uc.pt

DEI Coordinator:
Ernesto Costa
ernesto@dei.uc.pt

WIT Coordinator:
Jorge Sousa
jorge.sousa@wit-software.com

Data: 3 de Julho de 2017

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Software Platform to support chat assistants and smart replies using NLP

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra
Pólo II, Pinhal de Marrocos, 3030-290 Coimbra

Tel: +351239790000 | Fax: +351239701266 | **info@dei.uc.pt**

WIT Software, S.A.
Centro de Empresas de Taveiro
Estrada de Condeixa, 3045-508 Taveiro, Coimbra

Tel: +351239801030 | Fax: +351239801039 | **info@wit-software.com**

Intern:
Mariana Gomes Leal
mleal@student.dei.uc.pt

DEI Coordinator:
Ernesto Costa
ernesto@dei.uc.pt

Júri:

Henrique Madeira
henrique@dei.uc.pt

Luís Paquete
paquete@dei.uc.pt

WIT Coordinator:
Jorge Sousa
jorge.sousa@wit-software.com

Abstract

Recent years have shown the boom of the "app market", with thousands of mobile applications being developed, many of which intending to provide helpful service to customers. Banks, telecommunication companies and others alike seek to improve the customer experience through a personal application. However, this market has now reached a saturation state, where users tend to refuse to install one application for each company. In addition, these applications are usually limited in their offer and resort too much to tactile input, which is at times not the most appropriate way to provide customer service. A recent trend to counter these limitations of the app strategy is to rely on virtual assistants, or chatbots , which integrate directly with existing applications such as Facebook Messenger or Slack and can drive a conversation using natural language. Thus, the development of a bot enables custom support without an additional application and with a different user input scheme, mostly focused on text, leading to overall company benefits. The company spends less money providing customer support and the customers enjoy a better, faster and easier to use service which fits their needs. At WIT Software, there have been efforts to develop a new bot platform with state of the art technologies to offer their mobile customers a set of automated customer support mechanisms. In this report, we provide details on the implementation of a new bot technology, directly integrated with this bot platform, which uses state of the art techniques, such as natural language processing and sentiment analysis, to provide a more sophisticated customer support.

Acknowledgements

The longest ten months that I have ever been through have now come to an end, and it is time to show my appreciation to those who helped me come this far.

First, I would like to thank the people at WIT Software with whom I shared most of my days in the last year. A special mention has to go to my supervisor, Jorge Sousa, for always making sure that the internship was following the right direction and for taking time to check on my own well being. You made it a lot easier to wake up early in the morning and face another work day. Another special mention goes to Pedro Andrade, who I can call my co-supervisor, for trying to make me less afraid of public presentation (your efforts will not be in vain) and for teaching me that we should constantly strive to exceed ourselves. To all the others that I had to bore with questions and explanations, know that I am also very appreciative of that time you dismissed in my honour.
I would also like to thank my DEI supervisor, professor Ernesto Costa, for always offering his availability and for helping me handle my constant crisis and doubts throughout the year.

Secondly, I would to like to thank my closest friends for still wanting to hang out with me even after we've spent months without a word. Years and years have passed, and with each new one I become more convinced that quantity means nothing when I have the three best friends in the world. As long as we are together in heart, kilometers won't bother me a bit. Especially when those kilometers allow me to visit you in Paris!

Thirdly, it would not be fair to leave my parents out of this page. Even though we share a hard time showing our feelings and thoughts, it would not be possible to get here without your help, and for that I am eternally grateful.

Finally, and without the intent of overshadowing those who I have mentioned above, I have to address a very big and special thanks to the most precious person that I have had the pleasure to meet. Thank you for showing me that life is much more, that I am much more. It startles me how easy it is for anyone to push us down and keep us there, and how hard it can be to find a person who is capable of pulling us out of that black hole. Thank you for being that person and thank you for showing me, everyday, that the future is much brighter that what is seems yesterday. I love you, João Ricardo.

Contents

List of Figures

List of Tables

Abbreviations

Acronym	Description
AI	**A**rtificial **I**ntelligence
AIML	**A**rtificial **I**ntelligence **M**arkup **L**anguage
ALICE	**A**rtificial **L**inguistic **I**nternet **C**omputer **E**ntity
API	**A**pplication **P**rogramming **I**nterface
BPMN	**B**usiness **P**rocess **M**odel and **N**otation
FAQ	**F**requently **A**sked **Q**uestion(s)
GUI	**G**raphical **U**ser **I**nterface
HCD	**H**uman **C**entered **D**esign
HTTP	**H**ypertext **T**ranfer **P**rotocol
IM	**I**nstant **M**essaging
IVR	**I**nteractive **V**oice **R**esponse
JSON	**J**avaScript **O**bject **N**otation
MVC	**M**odel-**V**iew-**C**ontroller
NLP	**N**atural **L**anguage **P**orcessing
NLU	**N**atural **L**anguage **U**nderstanding
OTT	**O**ver-**T**he-**T**op content
RCS	**R**ich **C**ommunication **S**ervices
SDK	**S**oftware **D**evelopment **K**it
SMS	**S**hort **M**essage **S**ervice
UI	**U**ser **I**nterface
WIMP	**W**indows, **I**cons, **M**enus, **P**ointer
XML	**E**xtensible **M**arkup **L**anguage
XPDL	**X**ML **P**rocess **D**efinition **L**anguage

Chapter 1

Introduction

This document reflects the work done by the author during the first semester of the one-year internship included in the Master's degree in Informatics Engineering in the Department of Informatics Engineering of the University of Coimbra. The internship takes place at Wit Software, S.A. and it is supervised by Ernesto Costa PhD, cathedratic professor at the University of Coimbra, and by Eng. Jorge Sousa, team leader at WIT Software.

This chapter provides an overall perspective of the report. Section 1.1 identifies the company where the one-year internship will take place. Section 1.2 describes the motivation for the project and Section 1.3 details the goals. In the last section the structure of the remainder of the document is presented.

1.1 The Institution

WIT Software, S.A. [13] is a company specialized in developing software products for mobile devices and telecommunications companies. Founded in March of 2001 as a spin-off of the University of Coimbra, WIT-Software has a strong position on the world market, having clients in more than fifteen countries including the major operators: Vodafone, Telefonica, Teliasonera and Orange. The company has offices in Coimbra, Lisbon, Porto, Leiria, San Jose in California (USA) and Reading (UK).

1.2 Motivation

As customers become more aware of the technology that surrounds them and start to accommodate the new trends into their lives, they expect their products and service

providers to do the same. Acknowledging the big explosion that occured in the past year related to conversational interfaces' popularity and chatbots as the new assistants, WIT Software is working on a new strategy that includes the expansion of its product set with a new bot platform. This platform will allow future customers to build and deploy their own personalized bots in a broad set of messaging services, get access to updated data analysis and follow the conversations in real time, among other possible features.

More important than just implementing the platform, WIT Software is interested in gathering a viable group of customers that is actually interested in the product. In order to offer an interesting value proposition, some bot prototypes must be developed and presented as a solution to the market they belong to. One of those bots is a Customer Support Bot (also known as 'Telco Bot'), and it is a big part of this internship. Telecommunication operators are a big business sector, and consumer satisfaction is one of the main keys to mark a company as successful. However, telecom companies do not provide tangible products most of the time, and the quality of the service is mostly determined by how good the relationship with the customers is [14]. Knowing that, there are three big points that make the company believe that customer support in telecom operators is a good market target for the Bot Platform project. Those points are:

- **Current customer support model is not satisfactory** - As aforementioned, the relationship between the company and the customers is what grants the major part of a quality service. Overtime, general solutions for customer support have been settled in the industry, which includes human assistance by telephone or e-mail and multi-tree IVR systems. Despite being very popular among the operators, these IVR systems are useless most of the time due to the confusing call trees and the lack of natural dialog that occurs throughout the conversation (since most of the systems are built to understand simple voice controls such as 'yes'/'no' and/or to trigger an event according to a keyboard key pressed).

  Statistics on the subject point out that about 83% of respondees viewed IVRs do not have any benefit to the customer [15]. If customers are not satisfied with the IVR system, they are left with the option of directly contacting the operators. The big disadvantage of this solution, for the customer, is the limited schedule of the operator's assistants and the charges of the calls. The customers are able to write an e-mail or use an online form, but these channels are asynchronous, since they also depend on a human assistant to be held. This chain of events leads to customers unnecessarily asking for live assistants support over self-service, which will raise the inbound call volume of the operator's call center and, consequently, prolong the average waiting time.

Possible improvements to the actual systems have been discussed throughout the years. One of the them is the application of a visual IVR, where customers can actually see the options instead of listening to them [16]. A chatbot dedicated to customer service easily provides a visual interface where the user can have quick access to the desired options, with less frustration and time lost. Even when compared with human operators, chatbots provide multiple advantages [17], such as 24x7 availability and scalability. When using a chatbot, a customer is never left with no answer.

- **Saturation of the App Market** - It is true that smartphone apps ruled the technology world starting in the second half of the 2000s. The fact that they were cheaper and relatively easier to create than computer apps, lead to a growing industry which produces big amount of products each year. The latest statistics state that more than 2 million apps are available for purchase in each one of the biggest app stores today, Google Play and App Store [18]. The problem is that, despite the big variety of apps to choose from, users are not downloading or using them as they did in the past. In the US, for example, studies report that about 65% of users download an average of zero apps per month [19]. It is not that apps are not useful or expensive. What happens is that customers just don't need that many apps, and stick to those that are more than suitable for what they need. And those apps are, precisely, the messaging apps, such as Whatsapp, WeChat or Messenger (see Section 2.1.1 for more details).

  If customers are transitioning to private messaging, then the telecom operators will have to engage on those platforms. It is obviously not cost effective to have human assistants chatting online all day long. This is where the chatbots assume their position as a viable alternative. Gartner Inc [20], the world's leading information technology research and advisory company, published in its last report [21] predictions for the next years. The report stated that, by 2019, about 20% of brands will abandon their mobile apps. The main factors for such decline include the low level of customer engagement that was expected and the costs to maintain the apps. Bots for customer service purposes are more than ready to replace their counterpart apps, bringing with them another set of advantages for both customers and companies:

  - Bots are **easier to install**. Instead of searching for a given app in an app store, waiting for it to install and creating an account so we can finally start what we want, we can simply open our favourite instant messaging application, search a bot from within the contacts and start chatting. The whole process is faster and simpler.

  - Bots are **cheaper**. A good mobile app requires a skilled development team that includes Android/iOS developers and perhaps some user interface designer. Even after it is developed, it must be refined and submited to the approval of an app store. The maintenance also requires higher costs since the app must be prepared to run on different operating systems and even different versions of the same operating system.
  - Bots are **where the users are**. As aforementioned, bots can be deployed in almost every instant messaging application, and those are the ones that the users are most engaged to. There is no need for a hard market penetration, since it already exists.
  - Bots are **easier to interact** to. Although humans have been trained in the last years to interact with a series of standard graphical interfaces, in the end, they are only humans, and as so, they are innately hardwired for language and conversation, using text or speech. No matter how similar the graphical interfaces are, they will always have their design differences, leading the users to being constantly learning and adapting. Chatbots powered by natural processing techniques and some basic artificial intelligence are able to react to that most underlying human trait. And even for more complex tasks (like booking an hotel or a flight), bots can easily use simple graphical options like quick-reply buttons to conduct the conversation.
  - Bots are **most suited** for some use cases. It is not likely that users will install apps for minor tasks such as scheduling an hairdresser session or checking what tariff he/she is using at the moment. Therefore, building apps for those purposes would be a loss of time, even knowing that the use cases are real and viable. Bots make it easier for customers, given their low-friction distribution and development cycle.

- **Economical benefits for the operators** - No doubt that customer satisfaction enhances the value of a company, but the industry needs actual money and profit in order to keep running. The good news is that a new solution based on virtual and human assistants cooperation might bring long-term economical benefits for the telecom operators. Reasons for that are closely related to the previous topics: if users find most of their needs in the chatbots, they won't go to the call centers with questions that they can ask the virtual assistants. Knowing that each inbound call has a certain average cost for the company, this call avoidance can potentially save some ammount of money [15]. As also mentioned, quality apps are expensive to build, maintain and deploy. With bots, the majority of the cost will be at the core backend implementation and not much from there on. Messaging apps (the new

platforms) are built by someone else and new features can be continuously integrated to the backend alone without much effort. In conclusion, the bot economy is growing fast. When comparing the growth of apps and chatbots in their respective early days, both saw substantial increases, but bots overpassed the former in about 70% [22], according to Citigroup Inc. In the same study, comparisons were also made according to the number of developers in the field: in the sixth month, there were already nearly three times the number of active developers working on the bots than the number of developers working on apps by their 14th month after the launch. It is more than time that telecom operators start thinking about the next step and redefine a customer support strategy that leads to a win-win situation.

Besides all these advantages, there is an issue that must be taken into account: even with the latest improvements in the artificial intelligence field of natural language processing, chatbots currently lack the necessary faculties to replace human expertise entirely, and their limitations are just as important to understand.[17].

That is why human cooperation is one of the most important features of the Bot Platform. Bots are great for computation tasks, but not so good at handling conversations for themselves. They are mostly designed for question-answering scenarios, and are not quite able to hold longer conversations as humans do. Even with a good natural language training set, they will have difficulties in spotting ambiguities that are normal for humans. Their knowledge base will always have to be updated manually by humans. However, in customer support, humans should have an even bigger role. A chatbot should provide a good answer or no answer at all. If it is not sure of what to reply, then it should let someone more experienced - the human assistants - handle it and learn from that experience for future similar conversations.

In summary, with a Telco bot integrated with WIT's Bot Platform, the telecom operators can expect an improvement in the customer experience and at the same time reduce operational costs and increase revenue. By combining the power of virtual assistants and the expertise of human operators, companies can expect an enhancement in their quality service and, consequently, in their costumer satisfaction.

1.3 Goals

Two main goal sections are attached to this internship. One corresponds to the enterprise experience that is acquired when working in a enterprise environment - the internship goals, and the other is related to the project itself and its outcome - the project goals.

1.3.1 Internship Goals

In this section, the main goal is to gather experience in an enterprise environment, learning how to develop software with a project team and project managers and where the final products are almost always expected to serve a broad number of customers. It is also expected that I not only apply the academic knowledge that I gathered in the last four years in a vast set of areas but also enhance it, whether it is by applying different methodologies and software processes (such as the SCRUM process) or by learning new concepts and technologies.

1.3.2 Project Goals

The goal of this project is to develop a functional prototype of a platform that supports chat assistants in a customer support context for telecommunications operators. Those assistants will be be deployed in a set of instant messaging applications where they will be available for costumers to ask for support in a finite set of tasks. The platform itself will be held by the human assistants, who will be able to read the conversations in real-time and to intervene when a chat assistance is having troubles (for example, when the chatbot is failing to understand what the user is asking).

In summary, there are two main outcomes of this internship:

- **Telco Bot for a telecommunications operator** - a functional chatbot that can provide customer support in two specific fields: service and question-answering. The first one can inform the user about his tariff and his current consumptions, for example (for detailed features please refer to Section 5). As for the second, the bot shall be able to answer to most frequently asked questions on a set of products or services provided by the operator. The users must be able to use natural text language when requiring support, as so, the server must implement a module of natural language processing that interprets inputs as actions. The prototype to develop will focus on the applications Message+, Backup+ and Call+, all owned by Vodafone operator[23].
- **Human Assistants' Platform Integration** - human operators shall be able to follow conversations with the Telco Bot and perform a set of actions (for example, disable a bot or set the flow of the conversation) using the platform dashboard intended for the human assistance. The platform shall also present real time information about the level of frustration of the conversations that are occurring (information such as 'There are nine conversations where the customer is experiencing anger feelings'. This helps the human assistants know which conversations

they should follow since there is a high probability that the chatbot is not able to provide the right answers in those cases). In order to provide this information, another module must be implemented in the application server, aimed at performing sentiment analysis for every conversation that occurs.

At the end of this project it is not expected to have a product ready to deliver to a customer. However, it is expected to have a fully functional proof of concept platform that meets WIT Software's high standards and expectations and helps the company define the next steps to be taken in the future concerning the Bot Platform proposal.

1.4 Structure

The structure of the remainder of the document is as follows:

- **Chapter 1 - Introduction**: this section presents an overview of the work, including the motivation and the goals
- **Chapter 2 - State of the Art**: this section presents a brief overview of the main concepts applied to the project and the analysis of the alternatives technologies that could be used in order to achieve the results
- **Chapter 3 - Methodology and Work Plan**: this section explains the used methodology, the development plan during the internship and the risk analysis
- **Chapter 4 - The WIT Bot Platform**: this sections gives an overall perspective of the Bot Platform and the architectural design of the major components
- **Chapter 5 - Requirements Elicitation**: this sections describes the process of requirements elicitation and presents the user stories and the requirements
- **Chapter 6 - Implementation and Work Done**: this section describes the work that was done throughout the year, the final result and some implementation details
- **Chapter 7 - Verification and Validation**: this section describes how the processes of verification and validation were handler and presents the functional tests that were performed.
- **Chapter 8 - Conclusion and Future Work**: this section presents the overall conclusions of the project and presents the future work-

Chapter 2

State of the Art

This chapter presents the current state of the art. A primal section provides background on the main subjects that compose the scope of the project and a second section presents the available technologies and the comparison that was made in order to choose the best tools.

2.1 Scope

The scope of the project comprises a set of concepts that are worth to be mentioned and slightly introduced. This section provides a short review of the evolution of instant messaging applications, conversational interfaces and chatbots in general.

2.1.1 Instant Messaging

The definition of instant messaging (IM) has slightly varied over time but, in its immutable core, by the term 'instant messaging' we refer to the synchronous real-time exchange of personal messages (speech or text-based) between two or more computer users [24]. In its early days, an Internet connection was not a requirement since the communications could happen by having all the involved users hooked up to the same computer. Nowadays, and given the technology evolution that has happened since, the instant messaging definition has become coupled with a transmission over the Internet [25] [26]. Instant conversations are also associated to another pair of features: presence information and availability awareness [25]. The former refers to data that describes the current status of each user (online and offline, for example) and the second denotes the willingness of a user to communicate with a certain user that started the chat request (for example in order to avoid intrusion by undesired content). Besides being globally defined

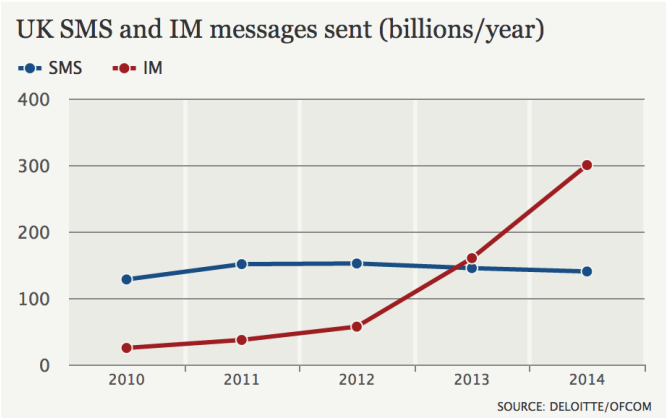

FIGURE 2.1: Volume of SMS and IM messages sent in the UK from 2010 to 2014 [1]

as a synchronous transmission, some authors prefer to use the term "quasi-synchronized" [27] to describe the nature of instant messaging, considering that the messages are not sent until the user presses the "Enter" key or another similar trigger.

The instant messaging applications might have been popularized in the early 1990s but its primal forms can be tracked starting in the early 1970s. [25]. The UNIX "talk" command became a type of standard at that time, despite not being very sophisticated and despite the fact that it interrupted whatever task the user was currently working on [24]. Major steps were taken in the following years, such as CompuServe's CB Simulator in 1980, which was the first service dedicated to online chat, and ICQ in 1996, that was the first text-based messenger to really reach a widespread market of users [28]. Between 1997 and 1999 big companies that include Microsoft and Yahoo! also released their instant messenger apps [29], and in the early 2000s Google and MySpace would join them [28]. Through time, those messaging apps started supporting richer media for conversations instead of only text, such as emoticons, photos, videos and mini applications like games and even bots.

With the rise of social networking, as well as the shift to mobile devices such as smartphones and tablets, instant messaging has endured and evolved. The beginning of the current decade was when the messaging applications entered a highway whose near exit is hard to predict. The revenues of SMS (short message services) in some countries began to fall for the first time in years and, over the same period, WhatsApp [30] installations rose from an estimated 5% to 85% [31]. Finally, in 2013, mobile instant messaging overtook for the first time the SMS texting [32]. A graphical representation of that surpassing in the United Kingdom is shown in Figure 2.1, based on a report produced by Deloitte [33].

More recently, in September of 2016, the market company "Business Insider" released a report [2] stating that the number of active users that used messaging apps had already

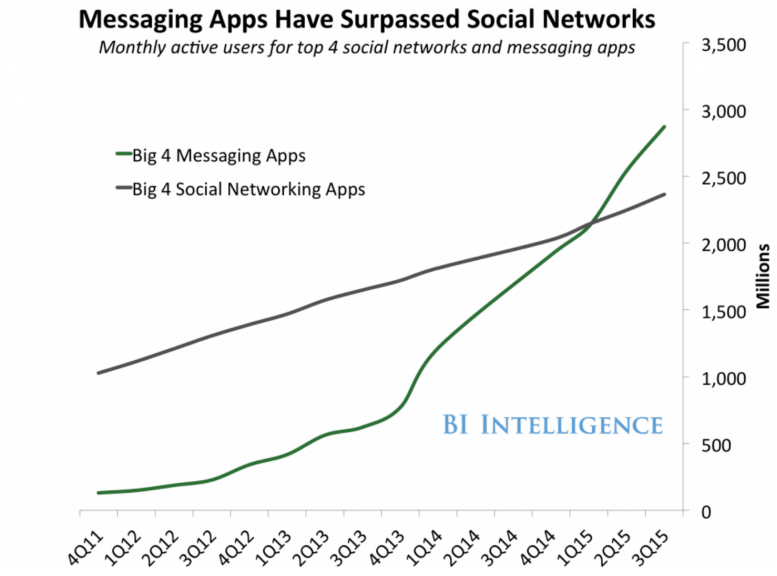

FIGURE 2.2: Monthly active users for top four social networks and messaging apps from 2011 to 2015 [2]

surpassed the number that used the top four social networks in just four short years. That comparison is represented in Figure 2.2.

If a potential future customer is looking for a product or service, a company needs to know where they are likely to start their search. And by looking at the latest results, that journey will definitely start online, on instant messaging applications.

The customer support bot that will be developed shall be available in at least one of the currently messaging apps in activity. Combining two metrics - active users [34] and geographic spread [35], the best three platforms that arise are Whatsapp, Facebook Messenger [36] and Viber [37]. Since at this stage Whatsapp is against the production of bots on its platform, other names may be considered, namely Line [38] and Telegram [39]. Besides all the aforementioned popular messaging services, it is expected that in the future the customer support bot will also be deployed in the RCSs (Rich communication services) application. What matters the most is that the bot must be implemented in a sufficiently generic way in order to allow its integration with any messaging service.

2.1.2 Conversational Interfaces

To the gap where the interaction between user and machine takes place, whether it is cognitively, perceptually or physically, we give the name of user interface [40].

Throughout the years, there have been various phases in how we have interacted with machines [41]. In the early days of computing, processors were very limited and the user interfaces were considered an overhead that could be avoided. The inputs were mainly

punched cards or paper tape, and this became known as the 'batch era' of user interfaces. In the following years, the textual interaction was introduced with the command-line phase, where requests were expressed as textual commands with specialized keywords and expressions.

A third phase emerged around the mid 80s, bringing a game changing user interface paradigm: the graphical user interface (GUI). By using visual representations of programs, files and actions that matched the people's mental models of the real world, this new era made it much easier for users to interact with a machine, through pointing and clicking directly on the desired action. Some GUI styles such as WIMP (graphical interface based on windows, icons, menus, and a pointing device, typically a mouse) have matured enough to be considered standard across almost every GUI-based interface and thouroughly entreched in the way we use computers [42]. With the commercialization of the personal computers and the rise of the user experience field, new design paradigms appeared concerning the learnability of an interface [43]. What was a technology-driven design, where the interface designers relied on training and practice to lead users to interact in a way that matches the system capabilities, evolved into a human-centered design (HCD) [44]. By using this approach, the main focus became the design of interfaces that are more intuitive and easier to learn, both for novice and experienced users.

However, whether it is friendly or not, an interface is still an intermediary between the user and the execution of its intent. Besides, computing has been growing into something that fills our daily lives, so it is expected that interfaces become more natural, intuitive, adaptive and unobtrusive [45]. For users, as human beings, the most intuitive way to communicate is through natural speaking, gesturing and facial expressions [43].

That is why a fourth wave has emerged, which is called the conversational interface. Instead of communicating with a computer on its own artificial terms (icons, commands, etc) we interact with it on our terms, by just telling it what to do. Deeply linked to the conversational interfaces are the conversational agents, i.e, computer programs that provide some form of service by interacting with the user through natural language dialogue [46]. Using conversational interfaces for purposes of business is not a recent idea. In the early 2000s, and even before, there were already a set of proposals where these type of agents could be applied, such as routing telephone calls [47] in a customer support environment and online sales [48]. The renewed interest in conversational interfaces that just emerged is highly related to the explosion in popularity of IM applications aforementioned in Section 2.1.1, among other factors [49]. With people spending most of their time in that type of applications, it is only expected that business companies go meet the potential costumers where they are.

Currently there are two basic types of conversational agents: the voice assistants, whom we interact with by speaking, and the chatbots, whom we interact with by typing. Every major tech company now owns a voice assistant: Apple has Siri [50], Google has Google Assistant [51] and Microsoft has Cortana [52]. These agents are mainly open domain and their goal is helping the user with quick tasks that range from setting a new calendar event to performing an Internet search. As for chatbots, there are thousands of use cases in which they can and are being applying. Not only the big names are betting on them, such as Facebook with M [53] and Slack with Slackbot [54], but also numerous brands have already started to provide their services through messaging platforms [55] and almost every day new names are following the example.

There is another concept that is worth mentioning, which is the hybrid interface. Although it is not recognized as a standardized term yet, it has been used [56] to refer to the fact that some messaging bots mostly rely not only on pure text but also on rich graphical UI elements such as quick replies and carousels. This hybrid interface can be particularly useful in scenarios of e-commerce and customer support. An example for the former is the user asking something like 'what t-shirts can I buy?' and instead of replying with text, the bot shows some preview photos attached to the respective t-shirt links. As for customer support, which is the domain of this work, a hybrid interface can help, for example, when we need the user to provide an objective answer; instead of the classical "type 1 for data consumptions or 2 for SMS consumptions" we present the possible answers in form of quick replies. Another example is the scenario where the user is asking for help. Instead of replying with a long text describing what the bot can do, it can present a carousel of options that the user can directly choose in order to perform the action. Some instant messaging platforms have already started to see the potential of this hybrid interface and already began to give developers the tools to manage these kinds of graphical UI elements [57].

2.1.3 Chatbots

A chatbot, also called chatterbot, can be defined as a computer program that interacts with users using natural language, often designed to convincingly simulate human behavior [58]. The interest in this type of program started to appear in 1950, when Alan Turing theorized that a truly intelligent machine would be indistinguishable from a human during a text conversation [59]. In the same article, which has been considered to represent the 'beginning' of AI, Turing also proposed the now called Turing test as a criterion of intelligence of machines. This test has been used until the current days and, according to the chatbots' primal definition [58], they must have the goal of passing that test.

The very first known chatbot was Eliza [60] and it came to public in 1966 under the appearance of a psychotherapist. Its conversational model was based in the rephrasing of the user inputs when they matched a set of pre-defined patterns. Although its conversational ability was not great, it was enough to lead some people into thinking that they were actually talking with a human in the first minutes of dialog. Six years later came Parry, a chatbot that was designed to simulate a paranoid mental patient [61]. It was a more advanced program than Eliza since it had its own state of mind (its beliefs and desires). Some other new features that Parry implemented include the ability to admit ignorance, change the current topic of the conversation and the introduction of small stories.

Not many improvements in the field occured in the following years, but the interest never really disappeared. In 1991, for the first time the Loebner Prize Competition was held, an annual competition where the chatbots are challenged with a simplified version of the Turing test [62]. The most human-like computer is distinguished with a prize, however, until today, no one has ever won the competition. Some participants that deserve a mention are Jabberwacky [63], the predecessor of Cleverbot [64], a bot that learns the behaviour and words of users, and A.L.I.C.E [65], a chatbot that used AIML [65] for specifying the conversation rules. In 2010, 2011, 2014 and 2015 Bruce Wilcox won the best chatbot prize with his creations Suzette, Rosette and Rose, chatbots built with Chatscript [66], a recent scripting language created in 2010 that has been known to be the AIML sucessor. Interestingly enough, the 2016 winner for most humanlike A.I. as a conversational program was Mitsuku [67], an AIML-based chatbot created by Steve Worswick. Worswick is still until today one of the greatest defensors of AIML as the best language to write chatbots.

There have been, however, some controversies [68] [69] about whether these types of competitions are contributing to the development of intelligent machines or not. The doubt is because most of the chatbots simply pretend that they are thinking without having real intelligence behind them (being deceptive might be interesting but does not show real cognitive features). Putting that aside, these type of competitions strongly contribute to the advances in the field since they encourage comparison of chatbot technologies, among others, helping old and new developers understand what is being used and how well they can perform [70].

It is clear, looking at the recent news about natural language processing tools and even at the new scripting languages that appeared, that there is a trend towards semantic approaches instead of basic pattern recognition. Regardless of current trends, there is no general better approach for creating a chatbot that is able to understand language. In the end it is always going to come down to the purpose of the chatbot. Whether it

has highly tailored tasks or it is just dedicated to maintaining a more or less general conversation, it is crucial to understand that the approaches to development may vary. A **rule-based pattern recognition approach** might be the best choice if we want an entertainment bot that can recognize some keywords and provide a general answer related to the topic. On the other hand, if we want our bot to provide specific tasks within its category (marketing help, customer service, retail, among others), we can think of an **AI-approach**, having a natural language classifier that is trained to map some utterances to the activation of a new workflow related to a determined task.

The apparent intelligence of a chatbot is measured not only in the way it understands the language but also in how it uses the language to reply back. In this realm we could identify two categories: the **retrieval based models** and the generative models. The former model is the easiest, and basically consists in using a repository of predefined answers and some kind of heuristic to pick the most appropriate answer. The challenging part in this model relies in that heuristic: it can range from a regular expression match to a machine learning classifier with some kind of confidence value, for example. Systems that use this model do not generate new text but rely on formats that already exist and, sometimes, use slight variations of it. This is the most used approach in current chatbots, since it is easy to implement and is enough to cover the domain of the bots. On the other hand, a **generative model** can generate new responses from scratch. Although with these models we can create the impression of a more human chatbot, they are very hard to implement. Natural language generation requires significant amounts of training data, not only about common natural language but also about the application domain [71]. Besides, they are likely to make grammatical mistakes (especially on longer sentences), an issue that does not exist in the retrieved-based models.

These models can be related with the chatbot domain according to Figure 2.3. Although generative models are more difficult to implement, they are the only possible choice when we enter the realm of open domains, where the user can set the topic to whatever he wants and the bot is expected to reply accordingly.

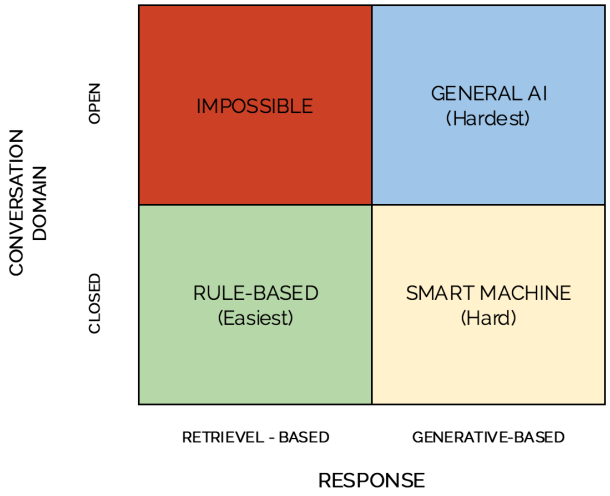

FIGURE 2.3: Chatbot Framework classification according to its conversation domain and response fetching process [3]

A customer support bot, for example, works on a closed domain. It is expected that it can answer a restrict set of questions but not to be able to talk about any random topic, from music to sports. It is almost certain that some users will try to start some small talk interaction, and it is also good to train the bot with some general topics in order to make it not so robotic [72]. However, in practice, no one expects a customer support assistant to be able to talk about anything.

The important thing to understand is that giving human traits to a chatbot is not a problem, it is even desirable in order to engage the users to the conversation. However, trying to pass off a chatbot as a human should not be a major concern. In fact, according to a study conducted by LivePerson on October,2016, 80% of consumers want to be told right away when they are interacting with a bot and not with a real person [73].

If a company can provide a Telco bot that understands natural language well enough, that triggers the right answers at the right time, that can chitchat a bit with the user about a few topics, that does not annoy the customers with constant notifications and that can adapt the future conversations by learning from the past conversations, it is almost certain that users will be more than pleased with the service.

2.1.4 Natural Language Processing

In this section we describe the evolution of the field of natural language processing and the main steps it involves. The second part of the section is solely dedicated to

sentiment analysis, namely the main approaches and the tools that were studied for the later implementation in the project.

2.1.4.1 Definition and Evolution

Natural language processing (NLP) is a field of computer science and linguistics that studies how machines can be used to understand and manipulate natural language for a range of tasks and applications. The term 'natural language' is used to distinguish the common human languages (english, portuguese, japanese, etc) from the formal languages such as C or Pascal, which are specifically developed to communicate and be understood by machines.

Given the fact that humans communicate with each other using natural language, a logical conclusion is that the easiest way for humans and computers to interact would be with natural language as well. Providing machines with linguistic capabilities has been a desire since the conception of the first computers themselves, and it is considered a measure of artificial intelligence [59].

The first major steps in natural language processing started shortly after World War II, mainly in the fields of automation and formal language theory. Turing's model of algorithmic computation [74] led to the development of computing elements that could be described in terms of propositional logic and to the work of Kleene on regular expressions [75]. Those search patterns became very popular in the late 60s and they have been used until the current days, mostly for pattern matching in data sources and lexical analysis in compilers. One of its first applications in programs happened when Ken Thompson built Kleene's notation into the UNIX popular search tool *grep* [76].

Machine translation - the field of AI that deals with translating text from one natural language to another - was the first linguistic computer-based application. Not surprisingly, the first results were disappointing, mostly due to the simplistic view that was followed: that the main difference between natural languages reside in their vocabulary and word orders [77]. As so, the first machine translation programs were no more than word-by-word substitution algorithms that did not take into account the lexical ambiguities inherent to each natural language. A well known urban legend in the field states that in the first public presentation of machine translation, the Georgetown-IBM system in 1954, the sentence 'the spirit is willing, but the flesh is weak' was translated to Russian and, when translated back to English, it was read 'the vodka is agreeable, but the meat is spoiled.'[78].

In 1957, the linguist Noam Chomsky took a big step in the field of language theory and defined for the first time, among others, the concept of context-free-grammar: a set of rules that described a vast set of sentences in a formal or natural language [79]. Sentences are not unordered random words, they are, in fact, grouped into functional constituents such as the subject or the predicate. Taking english as our natural language, we could build a simple grammar such as the one represented in Figure 2.1, where a sentence (S) can consist of a noun phrase (NP) followed by a verb phrase (VP). We would be able to generate english sentences, such as "The girl can drive a car" or "A man will hit the dog".

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow Art\ N \\
VP &\rightarrow Aux\ V\ NP \\
Aux &\rightarrow (can, may, will, must) \\
V &\rightarrow (drive, hit, eat, sleep) \\
Art &\rightarrow (a, the, an) \\
N &\rightarrow (boy, man, car, girl, dog, doll)
\end{aligned}
$$

Listing 2.1: "Example of a context-free grammar for generation of sentences in english"

Structuralist grammars such as the above one carry a lot of handicaps that Chomsky himself pointed out, stating that they were not adequate to account for all the syntactical facts of natural languages. He claimed that in addition to these structure grammars a second kind of rule is required, the "transformational" rules, which transform phrase markers into other phrase markers by moving, adding and deleting elements. Grammars that include those types of rules were defined by Chomsky as Transformational Grammars (TG) [79]. With this new type of rules, sentences can be easily related in a way that the primal structuralist grammars could not allow. For example, given the sentences "The boy will be hit by the car" and "The car will hit the boy" and a structuralist grammar, we would have no way to picture the similarity of the phrases (when they actually mean the same thing, despite the different modes) because it would give us two unrelated descriptions of them. By using a transformational rule to turn passive voice mode sentences to active voice mode ones, that type of co-relation is possible [80]. Another example of a common transformation rule in the English language is the subject-auxiliary inversion, that can allow a grammar to derive a question from a statement ('The boy has read the book' is transformed to 'Has the boy read the book?').

Although CFGs provide a simple mechanism for describing how sentences in natural language are built from smaller blocks (noun phrase or verb phrase, for example) they

are not entirely suitable to represent the entire grammar of any natural language, as originally thought. Some natural linguistic features, such as agreement and reference, are not part of the context-free grammars. In addition, in the mid-80s, several independent studies were published [81] [82] proving that context-free grammars could not represent certain languages, namely Swiss German and Bambara.

While CFGs are theoretically inadequate for natural language, they are still employed in practice for natural language processing (and for the formal description of the syntax of programming languages). Even if they can not be totally accurate about a language's grammar, they can exactly describe the basic recursive structure of sentences (what is the subject?, what is the action?).

Alongside the aforementioned theoretical developments, a few isolated NLP systems were being developed at the time, and those which were had little theoretical basis to the field of NLP [78]. Popular chatbots such as Eliza [60] and Parry [70] were performing a simple keyword based analysis and returning pre-defined phrases. Those type of systems were amusingly fluent and entertaining, but they lacked understanding. It would be in the 70s that the first implementations of the linguistics theories would start to appear, not solely at the syntactic level (which is the major goal of the linguistic grammars) but also at the semantic level. The LUNAR system [83], one of the earliest question-answering systems, had a semantic interpreter that would turn the original sentence into a database query and answer questions about the rocks returned from the Apollo moon missions. Another popular program was the SHRDLU system [84], a conversational system that allowed the users to move objects and ask questions about a simplified visual "blocks world".

In the late 1980s, a revolution started in the field with the introduction of machine learning algorithms applied to language processing. This was mostly due to the increase of computational power (resulting from Moore's Law), the lessening of the popularity of Chomsky's theories and a general push toward applications that worked with language in a real life context [85]. This reorientation resulted in the so called 'statistical NLP', an approach that was shown to be complementary to the old symbolic approaches. For example, the classic CFGs started to have a statistical parsing process, where individual rules have associated probabilities to help to disambiguate. Another recurrent solution was to build decision-trees from feature-vector data, a set of probabilistic rules built from annotated data. By this time, the role of the data sources to be used by the machine learning algorithms had become important [86]. Several attempts were made to turn commercial dictionaries in machine-readable form, which led to work using test corpora to validate or enhance existing lexical data. It was obvious that a 'most-likely' rule is

context-dependent; for example, a parser trained with annotated Wall Street articles may be unreliable for a clinical text.

The key advantage of statistical models is their ability to solve some of the ambiguity problems that are inherent to natural language processing. After all, almost any language problem can be recast as "Given N possible choices of reply to an ambiguous input, choose the most probable one". Not only that, they are also crucial for almost every NLP step, from part-of-speech tagging to reference resolution [78]. The most common machine learning tools, classifiers and sequence models, attempt to assign a simple object to a single class or to assign a sequence of objects into a sequence of classes, respectively. For example, for a spell-checking system, classifiers such as decision trees or support vector machines (SVMs) could be used to make a binary decision for one word at a time (correct or incorrect). On the other hand, sequence models, such as hidden Markov models, could be used to assign correct/incorrect labels to all the words in a sentence at once. By the beginning of the second millennium, statistical and data-driven models had already become quite standard throughout natural language processing, not as substitutes of symbolic models but as a complement of the early approaches.

In the last years, unsupervised and semi-supervised approaches began to receive a new attention [87]. Such algorithms are able to learn from data that has not been hand-labeled by inferring properties about the distribution of the objects in the dataset. The cost and difficulty of producing annotated corpora became a limiting factor in the use of supervised algorithms, leading to studies in a different trend. Although some research showed an effective application of systems trained on unannotated data in the areas of machine translation [88] and topic modeling [89], unsupervised algorithms typically produce less accurate results and need a wide set of data to train. It is expected that research and application of unsupervised techniques (such as clustering) in natural language processing will continue to increase in the next years.

Lastly, we should mention the role of knowledge in natural language processing. What distinguishes language processing applications from other data processing systems is the use of knowledge of language [78]. The tripartite distinction into syntax, semantics and pragmatics is only a starting point when we consider the processing of real data. A finer-grained decomposition of the steps of the process is demanded when we need to deal with real scenarios. The most common kinds of knowledge of language [78] [90], also designated levels of language, are represented in Figure 2.4, followed by a brief description of each level.

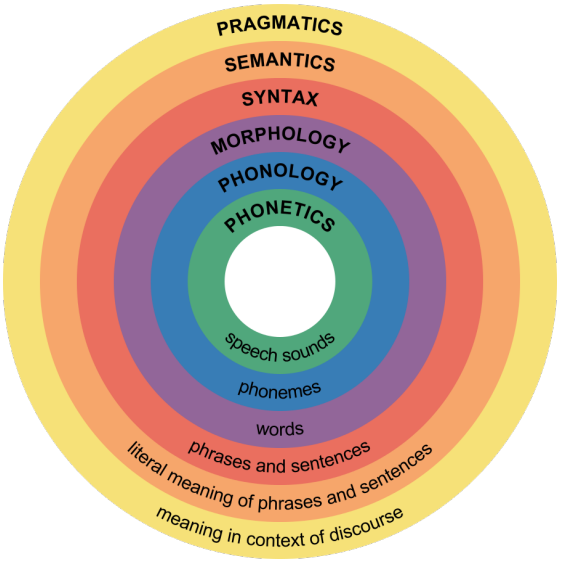

FIGURE 2.4: Major levels of analysis in processing natural language [4]

- **Phonetics/Phonology** - knowledge about linguistic sounds
  Almost always paired together, the level of phonology deals with the interpretation of speech sounds (phonemes) within and across words. A phoneme is the smallest unit of sound that despite having no meaning by itself can cause a change of meaning within a language ('bake' and 'brake' differ by one phoneme and are two different things).

  These levels of language only concern systems with speech inputs and are studied in the field of computational phonology. In an NLP system that accepts spoken input, the sound waves are analyzed and encoded into a corresponding digital signal for interpretation (using phonetic or prosodic rules or comparing to a particular language model).

- **Morphology** - knowledge of the meaningful components of words
  In this level, individual words are analyzed according to their components called morphemes, the smallest meaningful unit of language. Morphology is concerned with recognizing how base words have been modified to form other words, for example, by adding prefixes or postfixes. While the word 'dog' consists of a single morpheme, 'dogs' consists of two morphemes, where the 's' indicates plurality. A more complex word such as 'unhappily' would be broken into three morphemes: 'un', 'happy' and 'ly'.

  The most popular processes in this level are described in Table 2.1, namely the tokenization, stemming and lemmatization and POS-tagging. An example of the outputs produced by each process in represented in Figure 2.5. Note that this is just an example of pipeline between processes; it is not required to follow that order or even to use all three of them.

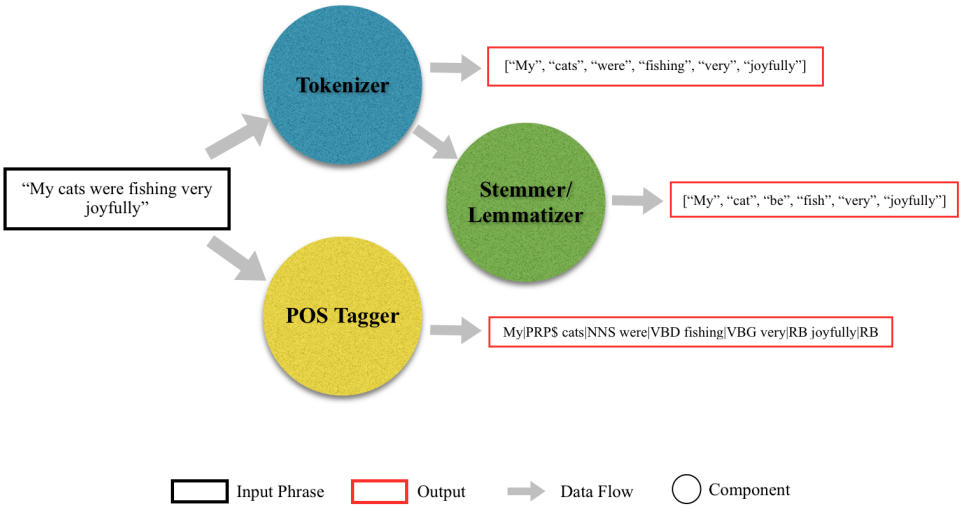

FIGURE 2.5: Practical Example of the three most used processes in the morphological level of language analysis. The POS tags presented are accordingly to the Penn Treebank Project [5]

Process	Description	Common Questions	Solutions
Tokenization	process of breaking a stream of text up into tokens (words and phrases, for example)	Tokens are commonly separated by whitespaces or punctuation, but this is not always accurate (for example, "aren't" should be seen as a single token, as well as "Los Angeles"); There are languages that are *scriptio continua*, such as Greek and Chinese, where tokenization can't use the whitespaces as boundary; Hyphenation should not be always treated the same away (for example, "co-educatio++ and "Hewlet-Packard" should be seen as single token but not the french "donne-moi votre chat")	Querying a table of common special-cases; Develop more complex heuristics (for example, allowing short hyphenated prefixes on words, but not longer hyphenated forms); Word Segmentation techniques for compound nouns or *scriptio continua* languages (having a large vocabulary and taking the longest vocabulary match, for example).
Stemming and lemmatization	processes of reducing inflected and derived words to their root form	Under-stemming and over-stemming (for example, remove 'able' from 'probable' is a case of over-stemming because probable doesn't mean 'able to be probed', they are words with different meaning); Mid-stemming, the act of taking off what looks like an ending, but is really part of the stem. While we could take the suffix 'ly' out of 'cheaply', we can't do the same with 'reply'; Languages with simple morphology, like English, have modest look-up table sizes, but highly inflected languages may have hundreds of potential inflected forms for each root.	Use dictionaries to avoid over and mid-stemming (e.g. reply does not derive from rep and that the meanings of probe and probable are well separated in modern English); A look-up approach may use preliminary part-of-speech tagging to avoid over-stemming (if the stemmer is able to grasp more information about the word, then it can apply more accurate normalization rules);
POS Tagging	process of classifying words into their parts of speech (noun, verb, adjective, etc) based on both its definition and its context	Natural Language is ambiguous 'Book' can either be a noun or a verb, depending on the syntactic form of the sentence. Even so, there are times where there is no syntactic way to decide ('The Duchess was entertaining last night.').	Use stochastic taggers, either HMM based, choosing the tag sequence which maximizes the product of word likelihood and tag sequence probability, or cue-based, using decision trees; HMMs can learn the probabilities of larger sequences, for example, if there is a noun followed by a verb, the next item may be very likely a preposition or an article but much less likely to be another verb. This type of tagging is called 'N-Gram Tagging'.

TABLE 2.1: Main processes used at the morphology level of language

Sometimes, the tokens obtained from the morphological analysis can be used for lexicon lookup on a dictionary. Not only is the POS tag checked, but a word that has only one possible meaning can be replaced by a semantic representation of that same meaning. For example, the words 'dog', 'cat' and 'fox' could all be replaced by the noun 'animal'.

- **Syntax** - knowledge of the structural relationships between words

  In the syntactic phase, the validity of a sentence according to a grammar rules is checked. The most common grammars used for analysing natural language are, as already explained, context free grammars (CFGs). By using a dictionary of word definitions (or lexicon) and a set of syntax rules, the syntactic analyser (or parser) checks if a sentence is well-formed and breaks it up into a structure that shows the syntactic relationships between the different tokens.

  The normal output of a parser is a syntactic structure, like a parse tree, which corresponds to meaning units when semantic analysis is performed. Taking as an example the grammar represented in Figure 2.1 and the sentence 'The girl can drive a car', a parser would produce the structure represented in Figure 2.6.

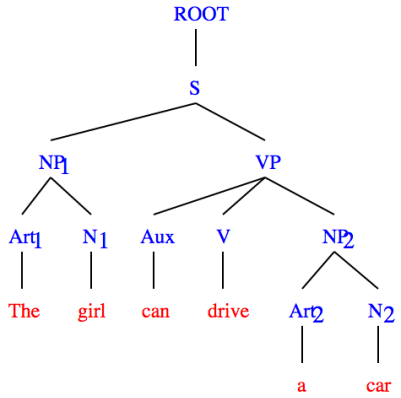

FIGURE 2.6: Parse Tree for the sentence 'The girl can drive a car' following the rules presented in 2.1

  Current research in this area includes alternative parsing strategies since pure bottom-up or top-down parsing is inefficient in many cases because they generate and explore too many structures which turn out invalid [90]. Other techniques include applying more than one strategy to the same input and integration with morphological analyzers to help with syntactic disambiguation.

- **Semantics** - knowledge of meaning

  This level is about the actual knowledge contained in the linguistic structures, including definitions (association of words with well-defined concepts), assertions about the relationship of concepts and implications within concepts.

  The meaning of a word can be directly retrieved from human-made knowledge bases (called ontologies), such as WordNet [91] but that is not always a trivial task. For instance, 'file' can mean either a folder for storing documents of a line of people in a queue. A semantic disambiguation must be performed by looking to the context of the sentence. This is commonly performed with word vectors and

algorithms of word embeddings that compute vectors representing the distributed representation for every word. Two popular services that offer large datasets and algorithms to calculate word vectors are Glove [92], an unsupervised learning algorithm for obtaining vector representations for words created by Stanford University, and Word2Vec [93] a neural network language model to produce word embedding released by Google.

A major part of semantic analysis is the representation of the meaning of a sentence in a way that allows a range of manipulations (inference, for example). Logic-based representation is an example that is easily derived from the syntactic rules. For example, using the verbs as center of the expression and a scenario like the world built for the SHRDLU system, when someone said "Move the pyramid 3 steps to the left" we could produce the correspondent semantic rule, in simplified form, 'apply Move(object=pyramid,steps=3,dir=left)'. These logic-based representations are associated with a semantic type of grammar named Definitive Clause Grammars [94], very popular because of their declarative style but often associated with inefficient parsing strategies. Other approaches to describe a sentence's meaning include the Case Grammar theory [95] which attempts to capture information about any activities in a phrase by describing it in terms of a fixed frame of slots (or cases). These slots can be something like 'action', 'actor', 'destination', 'time' or any other one that we think we need. For example, a sentence like "Mary eats the banana in the bedroom on Sunday" would generate the following case frame:

- Action - eats
- Actor - Mary
- Object - banana
- Location - bedroom
- Time - Sunday

It is obvious that. to implement a case grammar, the notation used for the rules must allow the production of case frames.

A final approach that is worth mentioning is that of semantic grammars, or domain specific grammars [96], that use rules built around domain specific sentences rather than the usual general language grammar rules. In a medical domain, a sentence like "He has a bleeding hole on his head near his left ear" the chunk 'on his head' could be classified by the grammar as a locator-phrase and "near his left ear" as a location-qualifier phrase. A major disadvantage of these kind of grammars is that they do not scale up well, meaning that they are more difficult to extend to deal with more complex or general language.

- **Pragmatics** - knowledge of relation between meaning and context
  Often connected to the level of Discourse, the pragmatic level is concerned with language use beyond the boundaries of a single isolated sentence. It thematizes the relationships between the language use and the language user in a given context and tries to explain how extra meaning can derive from text without actually being encoded in it. Obviously, for some sentences whose intended effect is clearly declarative, the pragmatic level can be skipped.

  However, sentences are not always declarative and obvious. A very usual task performed in the pragmatic level is handling pronouns. When one receives the two sentences "Mary eats bananas. She likes them", we must use the first sentence in order to assign the nouns 'Mary' and 'bananas' to the pronouns 'She' and 'them', respectively. In other cases, pragmatic analysis can disambiguate sentences which could not be disambiguated in the previous levels. Given the utterance "Put the apple in the basket on the shelf", two possible semantic interpretations arise: put the apple that is in the basket on the shelf or put the apple in the basket that is on the shelf. Based on the current positions of the mentioned objects in the world scenario, the pragmatic analyzer should provide the best choice.

On the other end of natural language processing, besides natural language analysis, is natural language generation, where we are concerned with mapping an internal language representation to a surface text. In the history of the field, there has been a lot more work on the first than there has been on natural language generation. Taking this into account, and also considering that the generation of text is out of the scope of the internship, techniques and approaches associated with the process will be left out.

2.1.4.2 Sentiment Analysis

In this section, an overall description of the state of the art approaches to sentiment analysis is presented. In a first subsection, the sentiment analysis field is introduced and the more popular approaches and algorithms that have been used recently are depicted with varying levels of detail. In the second and last section, we present the sentiment analysis tools that have been looked at and evaluated in order to choose the one that would perform best in the context of the internship. The algorithms that are more detailed in the first section correspond to the ones present in some of those same tools.

2.1.4.2.1 Definition and Approaches

Sentiment analysis, also referred to as opinion mining, is a process that combines techniques of natural language processing, text analysis, computational linguistics and biometrics in order to extract, classify, understand and determine an opinion expressed typically in unstructured text. Although there has been a low undercurrent of interest in the area throughout the years that followed natural language processing advances, in the beginning of the second millennium the awareness of opportunities that sentiment analysis could bring started to spread [97] [98].
Two main reasons might have been behind that sudden attention: one, the rise of machine learning methods applied to natural language processing, as already discussed, and the blossoming of the World Wide Web with reviews websites. It was also by that time that the field gained the designation of 'sentiment analysis' [97].

Although they commonly appear paired together, sentiment analysis and opinion mining have slightly different notions [99]. While the former identifies the sentiment expressed in any text, opinion mining extracts and analyzes people's opinion about a certain entity (a product, a movie, or anything else). The scope of the internship falls in the first category, since the intention is to use sentiment analysis techniques to analyze the user interactions with the telco bot. It is not important to know what the user is saying about a certain subject, but how well the conversation is going with the bot, because when it is not, the human operator must intervene and solve the situation.

Word of mouth (WOM), the passing of information from person to person, plays a major role in customer decision making regarding products and services [100]. WOM involves consumers sharing their experiences and opinions with other people and, when positive, it is considered a powerful marketing medium for companies to influence their customers. With this knowledge, companies started to see the potential role of sentiment-analysis and opinion mining as enabling technologies for their other systems. Some examples of applications are augmmentation to recommendation systems [101] (maybe the system should not recommend products with a lot of negative feedback), detection of frustration or antagonistic language in emails or other types of communication and detection of sensitive content in web pages where the placement of certain ads is not desirable [102].

In the past few years, there has been significant growth in the use of microblogging platforms, with millions of messages appearing daily in popular web-sites such as Twitter, Tumblr or Facebook [103]. Because of the free format of messages and the easy access to those platforms, more and more users have been shifting from the traditional blogs and mailing lists to posting about the products and services they use in their personal profiles.

While there has been a fair amount of research on sentiment analysis on online reviews and news articles, there has also been a lack of studying on how sentiments are expressed given the informal language and message-length constraints of microblogging. Currently, the area is under constant attention with dozens of papers published about how to extract sentiment from tweets and new open-source datasets such as the AFINN Lexicon [104], a list of English words rated with an integer between -5 (negative) and 5 (positive), or SentiWordNet [105], an extension of WordNet where each word is annotated in a range of [0,1] based on positively, negativity and neutrality.

Subjectivity is another relevant property of language that must be addressed when talking about sentiment analysis [106]. It refers to the linguistic expression of someone's opinion, beliefs and speculations. The binary classification task of labeling an expression as overall positive or overall negative opinion is called polarity classification and only makes sense in situations where subjectivity is present. However, the task of analyzing text in terms of subjectivity/objectivity is not trivial [107], and sometimes the difference between them is very subtle. Consider, for example, the tenuous difference between "The battery lasts two hours" and "The battery *only* lasts two hours". In addition, objective sources may contain inner subjective sentences (like quotes on a news article), making the subjectivity of sentences depend on their context. In cases like the latter, it has been shown [108] that removing objective sentences before classifying its positivity/negativity improved the sentiment detection performance. The relation between sentiment and subjectivity classifications is represented in Table 2.2.

Sentiment Analysis	**Subjectivity Analysis**
Positive	Subjective
Negative	
Neutral	Objective

TABLE 2.2: Relation between sentiment analysis and subjectivity analysis

Note that the label 'neutral' is used as a label for the objective class, when there is a lack of opinion. Some tools of sentiment analysis use this definition when labeling sentences as 'neutral' while others prefer to see the term as a sentiment that lies between positive and negative. Given the application that we intend to give to a sentiment analysis tool – the sentiment level of a conversation between a user and a bot – both interpretations are well suited. However, the same couldn't be said if we were talking about a rating system, since there are have been experiments [109] showing that the information contained in a neutral rating is often perceived by users to be much closer to negative feedback.

There are three main classification levels to consider in sentiment analysis:

- **Document-level** - aims to classify an opinion documented as expressing a positive, negative or neutral opinion or sentiment, considering a single topic.
- **Sentence-level** - aims to classify the sentiment expressed in each phrase. The first step is to identify whether the sentence is subjective or objective and the second is to perform the polarity on the subjective portion of the sentence [108].
- **Aspect-level** - aims to classify the sentiment with respect to the specific aspects of entities, knowing that the user can give different opinions about different aspects of the same entity (for instance, "The camera quality of the iPhone 7 is great, but the battery life is not that good").

As already mentioned, the sentiment analysis we intend to perform is abstract, not tackling any specific subject or entity. Therefore, the remainder of this section only targets sentence-level aspects, since this is the one that matters to the internship scope.

Contrary to simple text classification, using simple lexical-based classifications does not produce satisfactory results in sentiment analysis [6]. It is possible that sentences convey a strong opinion without using many strongly subjective words and that users express a positive opinion by using slang words and other pejorative words. Take, for instance, the phrases "This film should be brilliant, the plot is great and the actors are over the top. However, it can't hold up" and "That was bloody awesome", respectively.

Given these problems, sentiment analysis has to address these hurdles with new methods of classification. Existing approaches can be grouped into three main categories: knowledge-based or lexicon-based techniques, machine learning algorithms and hybrid approaches [7]. The former relies on sentiment lexicons, precompiled and hand-made collections of sentiment words or expressions, while the machine learning approach applies state of the art machine learning algorithms to the sentiment analysis context. Hybrid approaches are quite common, with sentiment lexicons playing a key role in the majority of the machine learning methods. This distinction between classification techniques of sentiment analysis is depicted in Figure 2.7, alongside other already mentioned categorizations.

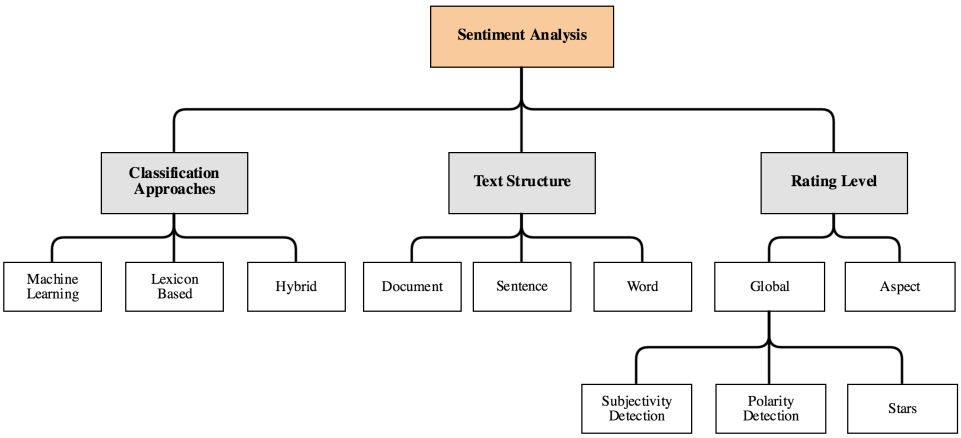

FIGURE 2.7: Possible Sentiment Analysis categorizations according to classification approaches, text structure and rating level [6] [7]

The state of the art machine learning algorithms that are most used in sentiment analysis tasks are represented in Figure 2.8. Afterwards, some of the most relevant algorithms in sentiment analysis are described with more detail.

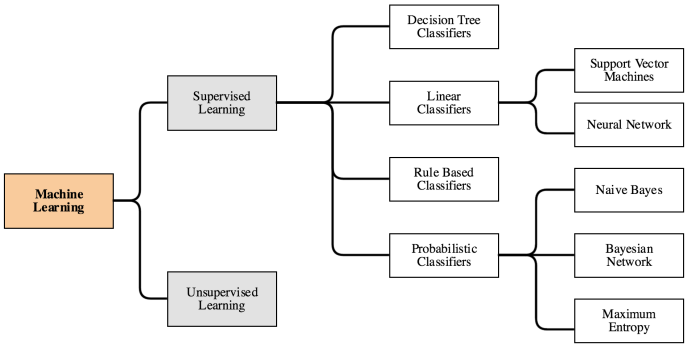

FIGURE 2.8: Machine learning algorithms applied in sentiment analysis [8]

- **Decision Tree Classifiers** A decision tree is a flowchart-like structure in which the nodes represent a condition on an attribute, the branches represent the outcomes of those conditions and the leafs represent a class label. These classifiers provide a hierarchical decomposition of the training data space in which a condition on the attributes values is used to divide the data [110]. Those conditions can be the presence or absence of one or more words ("does the word 'bad' or any variant of it appear in the sentence?"), the frequency of certain word clusters or any other feature that is discriminatory enough.

  Decision trees are simple to use and easy to understand, but they carry some drawbacks with them. One of them is the instability, because even a small change in the input data can, sometimes, cause big adjustments in the tree. This can

undercut the confidence and hurt the ability to learn from it. Additionally, decision trees try to find interactions between variables. If we happen to have a dataset that produces a lot of uncorrelated variables, the performance of the algorithm is affected, making linear methods a much more reliable choice.

- **Linear Classifiers** Linear classifiers are the algorithms whose output (or score or probability) of a particular classification is based on a linear combination between an object's feature vector and their respective weights.
  Formally, being $\overline{X} = \{x_1, x_2, ..., x_n\}$ the feature vector (for example, with some terms' occurrence frequency) and $\overline{W} = \{w_1, w_2, ..., w_n\}$ the vector of linear coefficients, a linear classifier method would construct the following decision function:

$$d(x) = \overline{WX} + b$$

  where b is a scalar [111]. The outcome of $d(x)$ is a separating hyperplane between different classes.
  Linear classifiers can be categorized as generative or discriminative models. The first ones model the distribution of individual classes, providing a model of how the data is actually generated, while the last ones learn the boundary between classes and provide classification splits [112]. In other words, generative models learn a model of the joint probability $p(x, y)$ of the input x and the label y and use Bayes rules to calculate $p(y|x)$, picking the most likely label y. Discriminative classifiers model $p(y|x)$ directly. Empirically, discriminative models often result in better classification performance, although it has been shown that generative models can outperform them when small datasets are used [113]. Support vector machine classifiers (SVMs) and neural networks are two of the most popular linear classifiers, and the two of them use discriminative models. A brief description of each algorithm is given next.

  - **Support Vector Machines** SVMs are supervised learning models whose main principle is to determine linear separators in the search space that can best separate the different classes. A SVM constructs one or more hyperplanes in a high dimensional space, which can be used for classification and regression tasks. In this context, data points are viewed as a p-dimensional vector that SVMs will try to separate with a $p-$-dimensional hyperplane [114].

    As a supervised method, SVMs seek to maximize the distance to the closest training point from either class - called the support vectors - in order to achieve better generalization performance on test data [115]. This good generalization characteristic is due to the implementation of the Structural

Risk Minimization principle, which entails finding an optimal separating hyperplane and guarantee the lowest true error [116]. The goal is to choose an hyperplane with the greatest possible margin (the distance between the hyperplane and the support vectors) giving a greater chance of new data being classified correctly.

For example, considering the Figure 2.9, the hyperplane C would be the best choice to distinguish between healthy people and osteoporosis patients. However, if we choose to work with hard margins, when confronted with the decision of choosing between the maximum-margin hyperplane and an hyperplane that classifies with more accuracy, SVMs will elect the second one. Take the example in Figure 2.10, where the hyperplane B would be the right pick. The problem with this hard margin approach is that it can result in a model that performs perfectly in the training set but that is also possibly less generalized when applied to a test dataset. The allowance of softness in margins allows for errors to be made while fitting the model but also for a small influence of the outliers in the hyperplane choice.

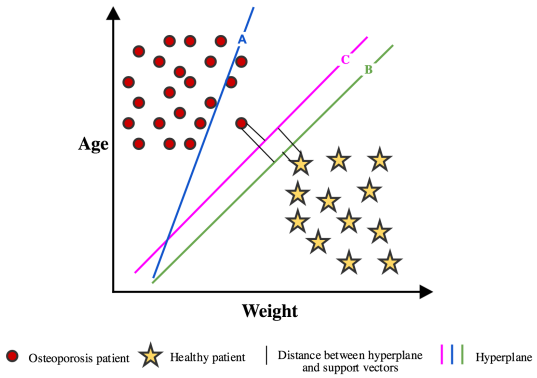

FIGURE 2.9: Maximum-margin Hyperplane

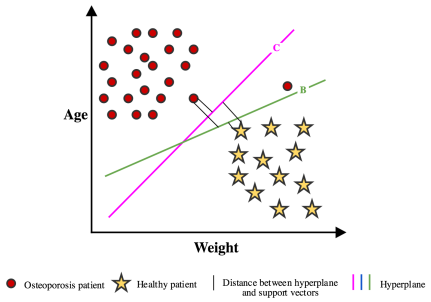

FIGURE 2.10: Classification Accuracy

FIGURE 2.11: Choosing the best hyperplane in support vector machines

Not all classification problems will have a training set that is linearly separable, meaning that the SVM will not always be able to define a linear hyper-plane between two classes. This limitation was overruled in the 90s [117] with the application of a kernel trick to maximum-margin hyperplanes. Kernel functions are functions which take a low dimensional input space and transform it into a higher dimensional space (or feature space). Figure 2.12 shows a graphical representation of that application.

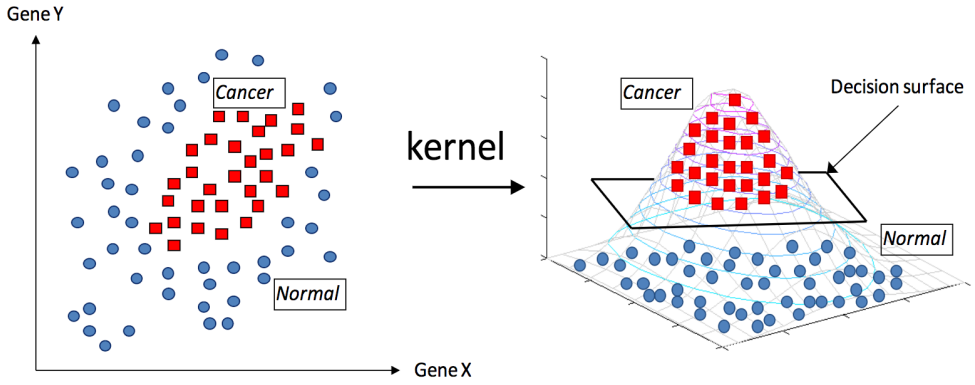

FIGURE 2.12: Representation of the kernel trick in order to allow SVMs to perform a non-linear classification [9]

Support vector machine are binary classifiers, meaning that they only perform a classification between two classes at a time. However, there are tasks where we will need to use more than two labels (take the example of sentiment analysis which is trilabeled - positive, negative and neutral). In order to do this, multiclass SVM should be used. The dominant approach is to reduce the single multiclass problem into multiple binary classification problems [118]. Two of the most common methods are 'one-versus-all', where we build a binary classifier which distinguishes between one of the labels and the rest, and 'one-versus-one', where a binary classification is performed between every possible pair of labels.

SVMs have guaranteed optimality due to the nature of the convex optimization that is used when searching for best hyperplane [117]. Linear SMVs are also very efficient when used with big datasets (in the order of millions of instances) because they only retain a smaller portion of the entire set as its support vectors, leading to reduced training times when compared to other margin classifiers. The same cannot be said about non-linear SVMs, mainly due to the state of the art algorithm Sequential Minimal Optimization, which shows quadratic performance. However, most of the time, the sparse nature of text leads to a large set of relevant features that tend to be correlated with one another and generally organized into linearly separable categories [8].

A common disadvantage of techniques such as SVM is the lack of transparency of results since they do not provide insights on the target event. Another drawback that is pointed to SVMs is the influence of the parameters' definition on the final performance. The key to avoiding overfitting lies in the careful tuning of the regularization parameter C and, in the case of non-linear SVMs, of the choice of kernel and tuning of the kernel parameters. The problem is that there is a lack of criteria over these parameters, and their fine-tuning

sometimes ends up being a trial-and-error process [119]. These are called the hyperparameters.

– **Neural Networks**

A neural network, more properly referred to as an 'artificial' neural network (ANN), consists of an algorithm that is modeled after the neuronal structure of the human cerebral cortex. It is typically represented by a network diagram which is composed of nodes connected by direct weighted connections, arranged in layers: the input layer, the output layers and n hidden layers. The neuron's output is determined by whether the weighted sum $\sum_j w_j x_j$ is less or greater than some threshold value (where w_j is the weight of the input x_j) [115].

By varying the weights and thresholds, we can get different models of decision-making. ANNs contain some form of 'learning rule' which is able to modify the weights of the connections according to the input patterns in the training phase of the algorithm. The most popular learning method for multilayer networks is the backpropagation method. In each training iteration, it uses the error values of each neuron of the network (by comparing the produced output with the desired one) to calculate the gradient of the loss function which is then used to update the weights [120]. A major drawback of this tuning method is that it is not guaranteed that the global minimum of the error function will be found, leading the model to converge only to locally optimal solutions.

The first and most simple type of artificial neural network is the feedforward neural network, where the information moves only in one direction - forward. On the other hand, recurrent neutral networks are models with bi-directional data flows, able to propagate data from later processing stages to earlier stages. A representation of both models is represented in Figure 2.13.

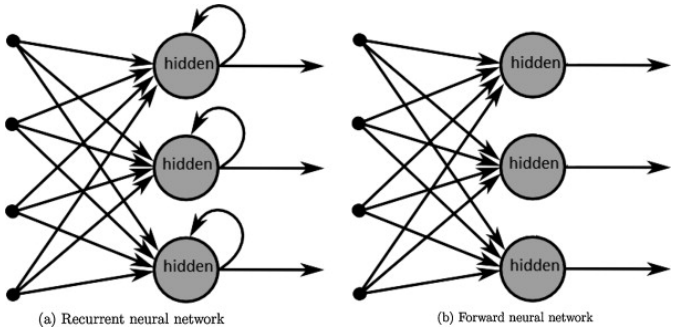

FIGURE 2.13: Recurrent and feedforward neural networks representations [10]

Neural networks can also be used for non-linear problems by adding more hidden layers to the architecture. The multiple layers are used to induce

multiple piecewise linear boundaries that approximate closer regions belonging to a particular class [8]. With a sufficient amount of data and time and the right architecture, one should be able to approximate any function.

A drawback that ANNs share with SVMs is the high influence of parameterization, in this case, the architecture: number of hidden layers to be used, number of hidden units to appear in each layer, activation function, among others [121]. Much like SVMs, ANNs are usually called black box learning approaches because it is hard to interpret relations between input and output. Apart from defining the general architecture of a network and perhaps initially seeding it with a random number, the user has no other role than to feed it input and watch it train and await the output. Without fully understand how neural networks solve a problem, we cannot know how well they will generalize when shifting to the testing process [122]. Contrary to SVMs, we do not hold much power in the avoidance of the overfitting, although there exist additional methods to help reduce the risk [123].

In summary, SVMs and ANNs have both demonstrated good results when applied to sentiment analysis tasks, despite all their differences. The former have evolved from a robust theory to the implementation and experimentation while the second followed an heuristic path, from extensive experimentation to the theory [124].

SVMs have become in the past years a standard tool for text categorization tasks and sentiment analysis, in specific, due to the good results they have shown in previous experiments. Early polarity classifications of movie reviews compared popular supervised algorithms and documented the superiority of SVM classifiers against probabilistic methods [125] [108]. Later research involving semantic distinctions in the feature set based on lexicons lead to increases in the accuracy of SVM classifiers [126]. In the past decade, experiments shifted from classifying larger pieces of text, like documents or big reviews, to microblogging messages. Even in the new context, where new language features can appear (for example, the presence of emoticons) SVMs kept their high accuracy against probabilistic methods [127] [128] [129]. They have been highly used for predicting sentiments on Twitter, both in general contexts [130] [131] or more specific ones, such as predicting the political alignment of the network users [132].

Neural networks have rarely been used as an approach for sentiment analysis and have rarely appeared in the literature. They have been popularized in other classification domains such as medicine [133] [134], image recognition [135] and character recognition [136]. In the field of natural language, neural networks have been successfully applied in tasks of text categorization (decide whether a document belongs to a certain class), namely backpropagation neural networks [137] [138].

Recursive neural networks, taking advantage of the recursive structure of the text, have achieved state of the art sentiment analysis results in movie reviews [139] [140]. Other experiments used simple feedforward networks and were able to reach around 70% accuracy on a Twitter dataset [141]. Another type of sentiment analysis was also conducted in movie domains, using self-organizing maps - an ANN that is trained used unsupervised learning - to cluster microblog posts according to subjective aspects like "funny" and "predictable" [142].

A recent empirical comparison [143] between SVM and ANN algorithms draws some interesting conclusions. Using three benchmark datasets of products from Amazon, they showed that ANN outperformed SVM by a statistically significant difference except for some unbalanced data contexts. Limitations of both methods were also confirmed in the study, namely the computational cost of non-linear SVMs at the running time and ANN at the training time. However, that cost can be reduced in both cases when applying certain feature selection methods.
Both methods have been successfully applied in sentiment analysis tasks but, as usual, the no-free lunch theorem [144] applies, meaning that no classifier is *a priori* superior to the other; the choice will depend on the nature of the particular dataset.

- **Probabilistic Classifiers**

  These are also known as generative classifiers, a second kind of models already discussed, which models the distribution of individual classes. The two most common algorithms used in text-related tasks are Naive Bayes and Maximum Entropy, and the first is briefly explained next.

  - **Naive Bayes** A simple and popular algorithm, the Naive Bayes classifier has been used for a while in text categorization tasks such as spam detection and document categorization, using word frequencies as the features [8] (also known as 'bag of words').

    Naive Bayes extends the Bayes' theorem by assuming that each data point is independent; a document or a sequence is treated as a sequence of words and it is assumed that each word is generated independently of every other. Using Bayes's theorem, the conditional probability can be decomposed as

    $$p(label_A|X) = \frac{p(label_A) \times p(X|label_A)}{p(X)}$$

    where $label_A$ is one of the possible classes and X is a vector of n features $(x_1, ..., x_n)$. Given the Naive assumption which states that all features are independent, the equation can be rewritten as follows:

    $$p(label_A|X) = \frac{p(label_A) \times p(x_1|label_A) \times ... \times p(x_n|label_A)}{p(X)}$$

The Naive Bayes algorithm is popular mainly because it is easy to implement, is very fast to train and test and usually works reasonably well [145]. Additionally, the fact that each distribution can be independently estimated as a one-dimensional distribution helps alleviate problems related with the curse of dimensionality (various phenomena that might compromise the optimization of a classifier when too many features are used) and allowing the Naive Bayes to not require as much training data as some other methods.

Despite its unrealistic independence assumption, the Naive Bayes classifier has been shown to outperform other methods when there are completely independent features or functionally independent features [146]. It has also been hypothesized that the Bayesian classifier may often perform better when the sample size is small [147].

The algorithm has been used for text categorization tasks, with and without modifications, and its performance results seem to diverge. It has been shown that SVM performs significantly better than Naive Bayes with unigram features, while the outcome is reversed with bigram features [148]. Some heuristic solutions to the problems with Naive Bayes classifiers have been proposed in order to enhance the quality of text classification results, such as adding a complement class to the training when dealing with unbalanced data or normalizing he classification weights [145].

In sentiment analysis, Naive Bayes classifiers have been successfully applied for classifying chinese documents [149] and online reviews of travel destinations [150]. In this last study, the experiments showed that when the training sets had a large number of reviews, the Naive Bayes approach would reach accuracies of 80%, pairing with the SVM approach. Other experiments combined a Naive Bayes classifier with a forward feature selection technique [151] and with background lexical information in terms of word-class associations [152], and in both cases the performance was improved. More recently, an improved Naive Bayes classifier was proposed [153] to solve the problem of the tendency towards a very positive classification accuracy. When using the algorithm with restaurant reviews, the accuracy was improved in both recall and precision.

In summary, machine learning algorithms are usually preferred to solve sentiment classification problems for its simplicity and ability to use the training data which gives them the privilege of domain adaptability. Throughout the years, the number of articles that use machine learning and lexicon based algorithms have changed [8]. The overall work for the recent years shows a growing interest on lexicon based and hybrid approaches. However, machine learning techniques are still an open field of search.

Other conclusions include the interest in languages other than English, namely chinese and arabic languages. The most common lexicon source is WordNet [91], which exists in more languages, and researchers are now working on building resources of European languages. There are some benchmark data sets, especially in movie reviews, but it is clear that there is a lack of data in the field. With micro blogs and forums playing a new role in sentiments expression, it is important to gather other sets that represent well these new forms of information. More than that, researchers and developers should work on building more domain-independent corpus, since it has been shown that working on domain-specific problems gives better results than more general sets [8].

2.1.4.2.2 Tools

One of the platforms to be prototyped during the internship is the Human Assistance Dashboard, a web interface where the operator's assistants will be able to follow the multiple conversations and intervene when one of the bots is having trouble with a costumer. Knowing that hundreds of conversations can be happening at the same time, and that it is not possible to assign one human to each conversation, it is import to define what is the threshold that a bot conversation has to meet before calling for help.

This is where sentiment analysis can play an important role. On one hand, it is possible to identify when a bot is not being capable of giving answers to a user, simply because it can't find any patterns that match the input. However, another problem that is harder to detect is having the bot matching requests to the wrong intents, and giving wrong answers. Unexpected replies may lead the user to overreact and to adopt a more aggressive or annoyed posture. If the Human Assistance Dashboard is able to keep track of the overall sentiment of a conversation, it can detect when that value reaches a given negative threshold, and send a warning to the assigned human operator.

It was not in the scope of the internship to implement a sentiment analysis classifier from scratch but to analyze the current state of the art of the field and possible available solutions. The former has been presented in the previous section, and the solutions found are presented next.

The sentiment analysis tool should be integrated with WIT's own platforms, and as so, desktop solutions and other kinds of isolated applications do not suit the purpose. In addition, paid-only solutions should be avoided. Other properties should also be taken into account, such as the classification time and the performance on the type of data that is common in messaging contexts.

A set of seven classifiers was chosen, taking into account the aforementioned constraints, including open-source and proprietary solutions. In order to choose the best solution for the problem, additional comparisons were performed. These regarded the performance of each classifier when applied to datasets that represent the type of data that the bots would expect to receive in common messaging interactions. Those datasets and the experimental procedure is described with more detail in Section 6.3.1.

A general comparison between the tools assets is represented in Figure 2.3. A more detailed description of the non-proprietary classifiers is given next.

	Free	Licence	Frequently Updated	Techniques	API/SDK	Labels	Languages Supported
Twitter Hybrid Classifier	Yes	Open Source (GNU GPL)	No (last in 2014)	Emoji detection, sentiment lexicons and Linear Kernel SVM	Python 2/3 module	(Positive, Neutral, Negative)	English
Tweetment	Yes	Open Source (GNU GLP)	No (last in 2015)	Sentiment lexicons and Linear SVM	Python 2/3 module	(Positive, Neutral, Negative)	English
aueb twitter sentiment	Yes	Open Source (GNU GPL)	No (last in May 2016)	Linear SVM, Sentiment lexicon and word embeddings	Python 2 module	(Positive, Neutral, Negative)	English
Vader Sentiment	Yes	Open Source (MIT)	Yes	Rule-based model combining lexical features and grammatical conventions for expressing and emphasizing sentiment intensity	Python 2/3 module	(Positive, Neutral, Negative). Can also return a compound score normalized between -1 and 1	English
IBM Natural Language Understanding	Free for 1000 items per day	IBM Property	Yes	Not specified	Rest API (demands creation of service in IBM Bluemix)	(Positive, Neutral, Negative)	English, French, German, Italian, Arabic, Portuguese, Russian and Spanish
Stanford CoreNLP	Yes	GPL+Apache / Proprietary	Yes	Stanford Sentiment Treebank corpus and Recursive Neural Networks	Rest API	Very Negative, Negative, Neutral, Positive, Very Positive	Arabic, Chinese, English, French, German, Spanish
Microsoft Text Analysis	Free for 5000 items per month	Microsoft Property	Yes	Microsoft Office's Natural Language Processing toolkit and features such as n-grams and POS tags	Rest API (demands creation of service in Microsoft Azure)	Score between 0 and 1	English, Portuguese, Spanish and other in beta.

TABLE 2.3: Comparison of the sentiment analysis tools analyzed

- **TwitterHybridClassifier** [154] - a project developed in the University of São Paulo, Brazil, which is now available after being present in the task 'Sentiment Analysis in Twitter' promoted by SemEval in 2014. SemEval (semantic evaluation) is an ongoing series of computational semantic analysis systems that have recently expanded the set of challenges with a task dedicated to analyze sentiment in micro blogging environments, namely Twitter.

  As the name indicates, the system adopts an hybrid classification process that uses three classification approaches: rule-based, lexicon-based and machine learning. The former only includes rules that detect the presence of emoticons in the text - the number of positive and negative emoticons determine its positive or negative classification. If there are no emoticons, the text is passed to the lexicon-based classifier where the polarity of a text is given by the sum of the individual polarity values of each word. Those polarity values are calculated based on a sentiment

lexicon [155] of positive and negative words, a list of sentiment hashtags [156] and a handcrafted list of negations.

When there is no clear difference between the positive and negative scores, the machine learning classifier is called, making the final decision. This approach includes a Linear Kernel SVM classifier trained with features such as unigram, bigram and trigam counters, the presence of negation, number of words in uppercase and the number of positive and negative words computed by the lexicon-based method.
Before the pipeline of classifiers starts, the text is submited to a normalization step where emoticons are grouped into representative categories (such as 'happy' or 'angry') and part-of-speech tagging is performed.

- **Tweetment** [157] - a Canadian project which participated in the task 'Sentiment Analysis in Twitter' promoted by SemEval in 2013. The team generated two large word-sentiment association lexicons, namely a list of sentiment hashtags and an emoticon lexicon that was based on an existing emoticon corpus of tweets [158]. This last one was generated by counting the instances where a given unigram or bigram was associated to a positive or a negative marker in the original corpus and assigning a final sentiment value between -5 and 5.

  The input text is tokenized and is part-of-speech tagged. It is then transformed into a feature vector made of groups of features such as word n-grams, number of words with all characters in upper case, number of occurrence of each POS tag, number of contiguous sequences of exclamation marks, question marks, and both exclamation and question marks, polarity of emoticons, polarity score according to the sentiment lexicons and number of elongated words (for example, 'soooo' instead of 'so'). The features were used to train a Linear Kernel SVM classifier. According to other experiments conducted by the authors, the sentiment lexicon features and ngrams counting were the most influential features.

- **aueb.twitter.sentiment** [159] - a project developed in the university of Athens that participated in the task 'Sentiment Analysis in Twitter' promoted by SemEval in 2016. The system consists of a weighted ensemble of two sentiment polarity classifiers, SP1 and SP2, each influenced by a subjectivity detection classifier, SD1 and SD2, respectively. The final label would be determined by the results obtained by the two classifiers (SD1,SP1) and (SD2,SP2). The former pair is a combination of two Linear Kernel SVMs: the SD1 is trained on data of two labels - neutral and subjective - and SP1 is trained is trained on the normal three labels. They used the same features, apart from the score of SD1, which is used as an extra feature to SP1. A total of 40 features is extracted from the original text, including

the number of elongated tokens, the number of upper-case words, punctuation marks, existence of slang, number of adjectives, number of interjections, existence of emoticons and polarity scores obtained from the sentiment lexicons.

The second classifier (SD2,SP2) uses the centroid of the word embeddings of the text as the feature vector. Word embedding is a technique used in NLP where words from a vocabulary are mapped to vectors of real numbers. In this case, the 200-dimensional word vectors for Twitter produced by Glove [92] were used, which follow a matrix factorization where a cell X_{ij} is a 'strength' that represents how often the word i appears in the context of the word j.
Because SP2 also uses the output of the SD2, it ends up using a 201-dimension feature vector. Both of them also consist of a Linear Kernel SVM, the first trained with two labels - neutral and subjective - and the second trained with the three normal labels.

- **Vader Sentiment** [160] - an acronym for 'Valence Aware Dictionary and sEntiment Reasoner', Vader is lexicon and rule-based tool that is specifically attuned to sentiments expressed in social media. It started by constructing and empirically validating a gold standard list of lexical features and their associated sentiment intensity measures. Those lexical features include a full list of Western-style emoticons (for example, ':-)' denotes a happy face), sentiment-related acronyms and initialisms (such as 'LOL' or 'WTF') and common slang with sentiment value (tokens such as 'meh', 'nah', 'bah'). Those features were then rated on a scale from -4 (extremely negative) to 4 (extremely positive).

  Those lexical features were combined with five general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity. Those rules are: punctuation (namely the exclamation point, which increases the magnitude of the intensity), capitalization (specifically using ALL-CAPS to emphasize a sentiment-relevant word in the presence of other non-capitalized words), degree modifiers (words such as 'extremely' or 'marginally' that impact sentiment intensity by either increasing or decreasing the intensity), contrastive conjunctions (tokens such as 'but' indicate a shift in the sentiment polarity) and examination of the trigram preceding a lexical feature in order to catch a wider range of negation flips that are not always direct (for example, 'The food here isn't really all that great').

- **Stanford CoreNLP** [140] - a set of human language technology analysis tools developed in Stanford University, where a sentiment analysis module is included. Instead of using semantic vector spaces for single words, the authors followed the trend of compositionality in semantic vector spaces to capture the meaning of

longer phrases. With this method, the order of the words in a sentence is not ignored and important information is kept.

Giving the lack of available large and labeled compositionality resources and models, the authors introduced themselves the Stanford Sentiment Treebank [161], the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. Instead of just labeling sentences as a whole, each sentence is split into multiple inputs, and each snippet is then classified from 0 (very negative) to 5 (very positive). For example, from a single movie review, they created the following corpus:

- "This quiet , introspective and entertaining independent is worth seeking" - 4
- "This quiet , introspective and entertaining independent" - 3
- "This" - 2
- "quiet , introspective and entertaining independent" - 4
- "quiet" - 2
- "introspective and entertaining" - 3
- "entertaining" - 4
- (...)

These representations are then used as features to classify each phrase, using recursive neural Tensor Networks.

2.2 Technologies

The year of 2016 was the year of the "chatbots hype", not only because of the aforementioned instant messaging rise but also because of the great improvement that happened to artificial intelligence software, mostly due to the new interest from big names of Silicon Valley. All popular messaging services started to open their platforms for 3rd party developers and, in the month of July, around 11 000 bots had already been created only for the Facebook Messenger platform [162].

In addition, service providers such as IBM [163] and Google also started to create and share their "AI as a service" platforms focused on NLP technologies. The main focus of these platforms is the same: providing public APIs that understand the intent of a user input and extract the necessary entities needed to provide a service. Due to the machine learning algorithms implemented by each platform, a new model is trained from a set of

example inputs that can be associated to a certain intent and therefore it should be able to provide significant results on inputs that are similar to the trained ones.

Before this new kind of platforms, in the early days, chatbots were developed using a combination of pattern matching and more or less complicated natural language processing and ontologies techniques [70]. Later on, with the creation of AIML, scripting languages started to become one of the most popular tools for building chatbots. The fact that these kind of languages are quite easy to use and yet powerful enough in some conversational scenarios is the reason why they have been used so broadly until nowadays.

However, a lot of companies have been releasing new solutions that offer the possibility of designing conversational flows without the need of a very demanding programming background. Companies such as Chatfuel [164] and Botsify [165] constantly advertise their propositions of "Build your bot in less than 5 minutes without any coding background", offering drag and drop graphical interfaces where question and answer scenarios can be easily set. This might by appealing for some small businesses who are just looking for a fast and simple way of deploying their services in a new interface, but for those who want to take it to the next level, these platforms may not be the right answer (at least not yet).

One of the major problems with those drag and drop interfaces is that there is no decoupling of competences: dialog flow designers should work on dialog scripts and usual programmers should work on back-end services. With those platforms, everything ends being mixed up. Figure 2.14 represents an example of a conversation flow designed for giving recipes to users according to some parameters like the ingredients.

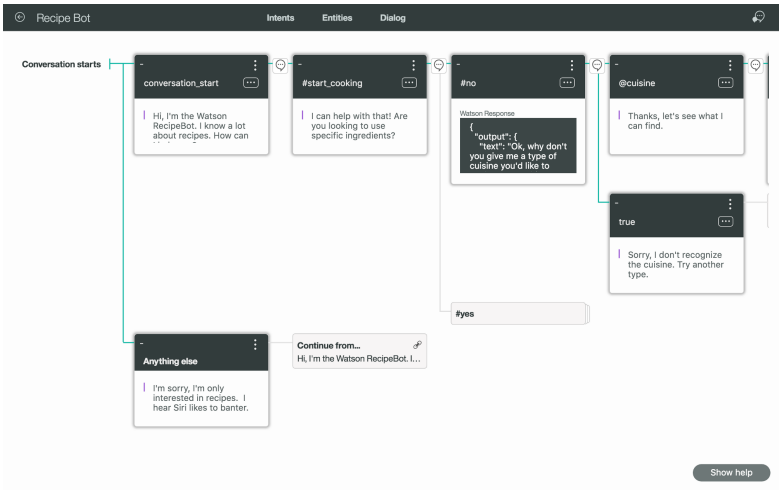

FIGURE 2.14: Example of a conversation flow for recipes in the drag and drop interface of IBM Watson Conversation [11]

Besides that loss of task division, companies become strictly coupled with those third party tools, and may have to upgrade to a paid plan soon enough (as more and more people are attracted to those kind of services). Not to mention that having the dialog flows and rules as flat source files - as it happens with scripting languages - can have an enormous value in the long term since they are easily editable, maintainable and reusable.

Taking all of these variables into account, the proposed solution for the WIT Bot Platform comprises three major concepts: a scripting language to handle the input-output matching for general conversation topics, a flow tool for building the conversational design of the most important requirements of the bots and a natural language processing API integration to map user utterances to specific intents that trigger the different flows. The chosen flow tool design was BPMN [166] since it has been the global standard for business process modeling for a while and it is taught in worldwide universities and business environments. As for the scripting language and the NLP platform, some comparisons had to be made first. The next sections are dedicated to those comparisons and to the final conclusions.

2.2.1 Scripting Languages

Along with the evolution of the chatbots in the 20th century, researchers and developers noticed that the basic pattern-matching techniques applied so far were not scalable or suited enough for chatbot development. Some basic NLP experiments were being done in the field [58], but it was also noticed that modeling bots by using natural language processing was revealing itself as a complex task. Natural language processing goes deep into problems of linguistics that were not very well understood at the time, and as so, the work that had been done was not quite useful to be applied in conversational agents [167].

As an attempt to enhance the pattern-matching method without getting much into the complex linguistics field, some developer communities started to work on dedicated technologies for chatbot making. The biggest outcome of this new research era was AIML, created by Richard Wallace between 1995 and 2000. These kind of languages are applied in dialogs that follow the stimulus-response approach. A set of possible user stimulus is modeled and then, for each one, the answers that can be returned are pre-programmed. Characteristics that differentiate these scripting languages from the first generation techniques include the capability of having a memory, contextualizing, reevaluating sentences and doing procedure calls to redirect different input patterns to

a same answer template [168]. Its structure was not complicated, but the performance was vastly superior when compared to the former bot generations.

AIML arose as a very powerful tool at the time, and it continues until today to be used and to produce good results in what concerns conversation design. For a long time, no more scripting languages were developed, until ChatScript appeared a few years ago. Later on, based on both of these precedents, RiveScript appeared, which can be considered an hybrid of the aforementioned languages.

A brief description of each of the three languages is provided next.

- **AIML (Artificial Intelligence Markup Language:** is a XML (eXtensible Markup Language) based markup language created by Richard Wallace in 1995. It is the basis for numerous chatbots, including Wallace's ALICE and Steve Worswick's Mitsuku, both Loebner Prize winners. The most important units of AIML are:

  - <**aiml**>: the tag that begins and end an AIML document.
  - <**category**>: the tag that marks a basic unit of an AIML dialog. The set of all categories makes the chatterbot knowledge base.
  - <**pattern**>: the tag that contains a possible user input.
  - <**template**>: the tag that contains the possible responses to a user input.

  There are also about 20 more tags that can be used, and it is possible to create personalized "custom predicates". The AIML vocabulary consists of words, spaces and special characters known as wildcards (_ and)) which are used to replace a string (words or sentences). Categories with patterns that have those wildcards are given a higher priority and, therefore, are analyzed first. Figure 2.2 shows an example of an AIML file.

```
<aiml version = "1.0.1" encoding = "UTF-8"?>

   <category>
      <pattern> WHO IS Isaac NEWTON </pattern>
      <template>Isaac Newton was a English physicist and mathematician.</template>
   </category>

   <category>
      <pattern>DO YOU KNOW WHO * IS</pattern>
      <template>
         <srai>WHO IS <star/></srai>
      </template>
   </category>
```

```
    <category>
     <pattern>WHAT ARE YOU</pattern>
     <template>
         <think><set name="topic">Me</set></think>
         I am the latest result in artificial intelligence.
     </template>
</category>

</aiml>
```

Listing 2.2: Example of AIML language

The <think> tag is used to perform an action but hide the result from the user. Using this tag along with the "topic" variable will allow any categories elsewhere with the same "topic" value to match better than categories with the same patterns that are not given an explicit topic. This is one example mechanism that shows how a developer can exercise control over a bot conversational flow.

Because AIML has existed for so many years, it has a lot of available resources. The A.L.I.C.E. A.I. Foundation makes three separate products freely available: the technical specification of the AIML language itself, a set of software for interpreting AIML and the contents of A.L.I.C.E. and other free bot personalities [169].

Since early 2013, the ALICE foundation has been working on a new specification, AIML 2.0 [170]. Among the new features are a new tag that allows the bot to access responses from remote web services and other AIML bots, new wildcards, conditional loops and local variables. AIML 2.0 also integrates Pandorabots extensions into the language, including formatted date and time and access to previous bot response history. Pandorabots [171] is one of the oldest chatbot hosting services in the world that implements and supports development of AIML bots. It is considered until today one of the world's leading chatbot platform, with more than 285,000 created chatbots [172].

Aside from the multiple development platforms, AIML also has a set of editors available for use. They serve as test environments for AIML commands and conversation sequence, helping to debug and decrease the development time. Editors worth citing include GaitoBot [173] and Simple AIML Editor [174].

- **ChatScript:** is a scripting language/chatbot engine developed by Bruce Wilcox and his wife, released as open source in 2011. Although being a recent language, it has already won three editions of the Loebner Prize and placed second in 2012 with the chatbot "Angela".

  ChatScript is organized into rules, which are gathered under topics. While AIML finds the best pattern match for an input, ChatScript first identifies the best topic

match and then executes a rule contained in that topic. As it happens with the AIML categories, developers can write topics on anything and just dump them into the system and have them automatically hooked up and working. An example of a e-book commerce conversation is represented in Figure 2.3. In this example, another ChatScript feature is visible – concepts, which are sets of synonyms. Once a concept has been created, it will match user input containing words within the concept. This is a concept that does not exist in AIML; in that language, synonyms are handled with the recursive calls tag, meaning we would need a category for each synonym.

```
concept: ~ebook NOUN_
     SINGULAR(ebook epub kindle_book
     nook_book Overdrive)

  topic: ~policies_ebook keep repeat
     (~ebook)

  t: We have many eBooks for you
     to download and enjoy. Follow
     this link for more information:
     http://www.ourlibrary.org/
     overdrive.html

  u: (need * card * checkout) You
     need to have a library card to
     check out eBooks. Do you have a
     card?

  a: (~yes) Great! Follow this link
     to our eBooks: http://www.our
     library.org/overdrive.html
```

Listing 2.3: Example of ChatScript language

Another feature that is implicit in the ChatScript engine is the ability to match both the original and canonical forms of a word (for example, 'book' or 'books' are regarded as the same). In AIML and RiveScript this kind of canonization has to explicitly done.

The only current ChatScript interpreter is written in C and C++, which makes the engine very fast, but it has the drawback of being the only existing engine. Since it is a recent trend, the ChatScript community is not very active, although during the internship, contacts were made with a bot enthusiast who stated that he had plans to develop a ChatScript interpreter in Ruby as a future project.

Since ChatScript is both a scripting language and an engine, implementation is also somewhat more difficult than AIML, and there are currently no hosting services available for ChatScript, although this may change in the near future. Additionally, it introduces many more symbols and 'text-noise' standards that make the learning curve bigger than the one in the other languages.

- **RiveScript:** is a plain text, line-based scripting language developed with the goals of being simple to learn and easy to read and maintain. The major intent of RiveSript's creator was to create a new language that would contrast with the other languages by not requiring the use of an "ugly" XML-based structure code (like AIML) and by not memorizing a lot of new random symbols and terminologies to write and read code (like ChatScript). Although it was as idea that emerged in the first half of the 2000's, it was only in 2010 that it started to be built as a serious project.

  RiveScript's intention is not be an all-in-one chatbot. It is only concerned with fetching a response to a user input message, leaving everything else to be set in the developers program. It is designed to be a self-contained software library with a simple API that developers can easily plug into an existing codebase.

  Its features do not differ much from the already existent, but its commands are made to be simple and easier to understand. Figure 2.4 shows an example of the language.

```
 + i hate you
- You're really mean! I'm not talking again until you apologize.{topic=sorry}

>topic sorry

  // This will match if the word "sorry" exists ANYWHERE in their message
  + [*] sorry [*]
  - It's OK, I'll forgive you!{topic=random}

  + *
  - Nope, not until you apologize.
  - Say you're sorry!
  - Apologize!

<topic
```

Listing 2.4: Example of RiveScript language

As described in their website [175], "RiveScript was designed to do everything that AIML can do and then some, and then some more and more". Some of the add-ons include definition commands and object macros. The former relates to

definitions such as word substitutions ('youre' is pre-processed to 'you are'), bot variables, global variables and person substitutions ('I am' generates an answer of the type 'you are'). These definitions are built right into RiveScript itself while the AIML bots need external configuration files. Object macros consist of programming functions written in another programming language (only works for languages for which RiveScript has interpreters). These functions allow to do extra things with the bot's responses, like fetching the Internet for information to present to the user.

The community around RiveScript is also very active when compared to languages like ChatScript. Third-party authors have already developed some first versions of RiveScript interpreters for C# and PHP. Those join the official interpreters ready to be used for the following languages: Go, Java, Javascript, Perl and Python.

Even though the new scripting languages are bringing interesting features and capabilities, they are still in a very mature stage. AIML has the advantage of being more mature, stable and easier to find resources for, given its years of development and contributions.

There are currently over 120 000 AIML rules available for a free integration in the knowledge base of any chatbot. More than that, there are extensive community contributions with categories in other languages than English, including German, French and Brazilian Portuguese. These contributions include not only AIML categories but also synonym lists and other natural language mappings that can be use to pre-process a user input before searching for an AIML pattern.

Additionally, software AIML's resources are also vast. The ALICE foundation offers for download in its website a selection of AIML interpreters for 10 different languages, including Java, C, Python and Ruby. Another advantage of using AIML in this project is the existence of libraries for Node.js, the language used in the platform. The amount of tutorials and forums dedicated to help AIML developers is also vast and another favouring point. With ChatScript and RiveScript it is almost impossible to find documentation and information outside their own projects.

In conclusion, there are many reasons why AIML is the state of the art scripting language for chatbot development. There are great perspectives for ChatScript and RiveScript in the future, but in the meantime, while they still lack important resources, AIML is the safest choice to use as scripting language in the Bot Platform.

2.2.2 Natural Language Understanding Platforms

The 'AI as a service' platforms were one of the big technology news in 2016. By combining natural language concepts such as intents and entities with machine learning techniques, these platforms boasted the potential of chatbots with conversation and artificial intelligence capabilities.

When thinking about developing a Telco bot aimed at providing the best possible support to the customers, it may not be sufficient to offer simple searches by regular expressions or keywords, like the scripting languages described in section 2.2.1. If in a chit-chat situation the user can be easily satisfied with some general answer provided by a scripting language, the same cannot happen when the user is asking for a specific task. It is not easy to write all the rules of regular expressions to match a user input to the intended intent. For example, these are some ways a user can ask the bot to tell them their current data consumptions:

- "What is my data consumption at the moment?"
- "Hows my dataplan?"
- "Tell me my data consumptions please"
- "How many mb do i still have"
- "check my data plan pls"
- "do i have mobile internet"
- "do i have enough data"
- "dataplan"

The intent mapping goes as far as simple cases like 'Yes' and 'No', where sentences like "Ok", "yeah!", "let's do it" should be rapidly perceived as a "Yes" intent. Chatbots with a clear set of tasks need to understand the user inputs in a way that is more sophisticated than simple text searches.

Some companies work with their own in-the-house solution, using popular NLP libraries such as NLTK [176] or Stanford CoreNLP [177] to perform the natural language processing and then add some kind of machine learning algorithm to classify the utterances. This alternative is very demanding and requires a good team of developers, and the result cannot always be expected to be the best. But given the advantages of this alternative (privacy of data and transparency of what is actually being done), another internship

is occurring at the same time at WIT, where another intern will be performing some experiments with one of those tools and evaluate the utility it can provide to the Bot Platform. Taking this into account, a comparative analysis was performed on the biggest five platforms in order to choose the best one use in the NLP module of the platform.

2.2.2.1 Comparative Analysis

In this section we present the results of the comparative analysis performed on the NLU platforms. First, each one of the platforms is identified and is given a brief description. That is followed by the explanation of the features used for comparison and a table with the results. The conclusions are presented in the end of the section.

Wit.ai was originally an AI start-up created on 2012 by two friends that had worked with natural language processing projects in the previous years. In September of 2013 they shared the first alpha version of the API and about 1100 developers signed up and started using it in the first day. From there on the start-up did not stop growing. On January 5th, 2015 the company was acquired by Facebook and it is until now the only platform that is totally open and entirely free for everyone.

Api.ai was first created by a team who had built a personal assistant app for major mobile platforms with speech and text-enabled conversations. It is one of the most advanced platforms when it comes to provided features. It supports the normal intents and entities and also the definition of action with optional or mandatory parameters, contexts and a set of pre-defined trained domains (such as small talk) ready to be used. According to the CEO, their big NLP and machine learning differentiators are based on the volume of data processed, which reached the million unit daily. It was purchased by Google on September 19th, 2016.

IBM Watson Conversation is part of a set of services provided by Watson Developer Cloud, maintained by IBM. In the last years, IBM Watson has been developing smart services that fall into the categories of language, speech, vision, data insights and embodied cognition. In the section of language, they originally created a service called IBM Watson Dialog that has been replaced by IBM Watson Conversation, a better version with enhanced machine learning methods and an intuitive user interface for conversation design.

Amazon Lex is the newest platform in the current state of the art: it only arrived on December of 2016, and it is under the control of Amazon Web Services. Unlike the other platforms, Amazon Lex restricts the access to its services, so it was not possible yet to get in contact with all of the features. Even being the latest to join the group, it has already brought with it some innovative actions, like bot version control and integration with some Amazon Web Services.

LUIS (Language Understanding Intelligent Service) was introduced during this year's Microsoft Build 2016 event in San Francisco, together with the Microsoft Bot Framework. It is part of Microsoft Cognitive Services, a set of APIs for language, speech, vision and knowledge analysis. It is currently in beta version and imposes a limit of ten entities for each application.

The list of features used for the comparison are listed below:

1. **Speech Recognition**: does the platform provide a speech recognition feature bundled with the NLU service?
2. **Free Plan**: does the platform have a free plan? What are the constraints?
3. **Paid Plan**: what are the price and benefits of the platform's paid plan?
4. **Chat Logs**: does the platform log the requests that are made to the API? Fow how long?
5. **Required Accounts and Services**: do we have to create an account in order to use the service?
6. **Creation of Intents(GUI)**: does the platform provide a graphical interface dedicated to the creation and definition of intents?
7. **Creation of Intents(File import)**: does the platform allow the creation and definition of intents by importing files? What formats are allowed?
8. **Creation of Intents(API)**: does the platform allow the creation and definition of intents by sending API requests?
9. **Built-in Intents**: does the platform already provide some type of trained intents?
10. **Confidence Level for Intent matching**: does the platform provide a confidence value for the matching that is performed between the queried input and the intent?

11. **Default Fallback Intent provided**: does the platform provide a default fallback intent that is matched when no other intent is matched with enough confidence?
12. **Creation of Entities(GUI)**: does the platform provide a graphical interface dedicated to the creation and definition of entities?
13. **Creation of Entities(File import)**: does the platform allow the creation and definition of entities by importing files? What formats are allowed?
14. **Creation of Entities(API)**: does the platform allow the creation and definition of entities by sending API requests?
15. **Built-in Entities**: does the platform already provide some type of created entities?
16. **Private Data**: does the platform assure the privacy of our data?
17. **SDKs**: does the platform provide any SDKs to ease the use of its API?
18. **Supported Languages**: how many languages does the platform currently support?
19. **Version Control**: does the platform offer some mechanism of version control?
20. **Integration Modules**: does the platform offer some mechanism of integration with other services or platforms?

2.2.2.2 Conclusions

During the first semester of the internship there was always some new feature that was important to evaluate or some updates in the platforms propositions that required the creation of multiple versions of the comparison. The version represented in Table 2.4 is the final compilation after three months of a dynamic state of the art that finally started to show some signs of stabilization.

We can see that some features are shared by the majority of the platforms. All of them provide a graphical interface that allows the creation of intents and entities and a predefined set of entities ready to use. The version control is clearly a lacking feature: aside from Amazon Lex, with the other platforms the developers are responsible for handling themselves the different bot versions. As it was already expected, all of the platforms require the creation of an account in one or more of their services and provide a free plan whose benefits are limited (excluding Wit.ai). In addition, four of the platforms offer an official SDK for Node.js, the language which the Bot Platform will be built upon.

	Wit.ai	Api.ai	IBM Watson Conversation	Amazon Lex	LUIS
Speech Recognition	Yes	Yes in paid plans	Yes when combined with IBM Text to Speech	Yes	Yes when combined with Microsoft Cognitive Services Speech API
Free Plan	All features	No limit of queries and bandwidth of 1 RPS	1000 queries per month, limit of 25 intents and 25 entities	-	10 000 queries per month
Paid Plan	-	Unspecified price for extra support and 99% uptime.	0.0025 USD/api call. Unlimited monthly queries. 2000 intents.	0.004 USD/voice request. 0.00075 USD/text request	0.75 USD/1000 queries (max 10 RPS)
Chat Logs	Yes, with no limit of time	Yes (for a period of a month)	Yes (last 90 days). Kept for 7 days in free plan.	No information	Yes (the most recent 100K utterances)
Required Accounts and Services	Wit.ai account	Api.ai account	IBM Bluemix account	Amazon Web Services account	Microsoft and Microsoft Azure Portal accounts
Creation of intents (GUI)	Yes	Yes	Yes	Yes	Yes
Creation of intents (File import)	No	Yes (json file)	Yes (csv file)	No	No
Creation of intents (API)	No	Yes	No	No information	No
Built-in intents	No	Yes (37 domains, one of which is free). Not all languages supported.	No	No	No
Confidence Level	Yes	Yes	Yes	No	Yes
Default Fallback Intent provided	No	Yes	Yes	No information	No
Creation of Entities (GUI)	Yes	Yes	Yes	Yes	Yes
Creation of Entities (File import)	No	Yes (json/csv file)	Yes (csv file)	No	No
Creation of Entities (API)	Yes	Yes	No	No information	Yes
Built-in entities	25 entities	44 system entities	5 system entities	Several slot types (numbers, dates, times and lists of items)	10 pre-built entities
Private Data	Yes	Yes (for paid plans)	Yes (when a proper header is set in the API calls)	No (Amazon can use data for enchancements. Can request deletion of voice logs)	No (Microsoft will use data to enhance its services)
SDKs	Node.js, Python, Ruby, Android and iOS	Android, iOS, Mac OSX, HTML, Cordova, Javascript, Node.js, C#, Unity, Xamarin, C++, Python, Ruby, PHP and Java	Java, iOS, Node.js, Android, Unity and Python. Community SDKs for Go and scala.	iOS, Android	Python, Android, Node.js and C#.
Supported Languages	11 (other 39 in Beta)	13	9	1 (US English)	5
Version Control	Manual	Manual	Manual	Versioning support for bots, intents and entities.	Manual
Integration Modules	No (only community examples)	One-click deployment on 15 channels	Multiple channels including Facebook and Slack (Botkit plugin)	Facebook Messenger (only text) and Amazon services.	Slack and Microsoft Bot Framework

Table 2.4: Comparative analysis of natural language understanding APIs

The platform that was chosen for the Telco Bot was **Api.ai**. The decision was not based on the shared features but, obviously, in the key features in which it differed from the others. First of all, it has the most complete set of techniques to create intents and entities, including GUI, file import and API calls. The former might be appealing for people with no programming background, but it very time consuming, which is not desirable. It is also the only platform so far that offers some kind of pre-built intents. Some of these are actually very helpful, like the small talk domain, which shifts away from the developers the task of creating scripts or workflows dedicated to extra chitchat talks that are not the main purpose of the Telco Bot. It is also the platform with more pre-defined entities and the second one with more supported languages, including English, Portuguese and Spanish. Lastly, it seems that Api.ai is the easiest platform to integrate, not only because of its one-click deployment on fifteen channels (that range from Facebook Messenger and Slack to Amazon Alexa and Google Assistant) but also because of its official SDKs for fifteen different languages.

Chapter 3

Methodology and Work Plan

In this chapter are documented the overall decisions related to project management. First, we explain the software methodology followed during the internship. In a second section, the work plan is presented and discussed. Lastly, in the final section, we present the risk analysis of the project.

3.1 Methodology

In software development, project management plays a huge role, allowing the planning of tasks and deadlines and the control and better guidance of what is being done. There is no silver bullet when it comes to the best methodology, meaning that a team must choose what methodology is more suited for a given project.
Agile development is the most popular software methodology at the company and it was also the chosen one to be applied to this internship. In particular, the SCRUM [178] agile process was used. Following this development framework, the team can make constant adaptations and the priority of requirements can change regularly. Since the scope and the requirements of the project were not totally immutable, SCRUM arose as the best methodology to use.

3.1.1 The SCRUM Framework

In the beginning of the project, a set of features that the product should implement is defined, which is called the **product backlog**. As previously discussed, this is not a static set, and items can be changed, added or deleted at any time of the project duration. A priority is then assigned to each item of the backlog, where the features with high-priority are completed first.

The development is structured in cycles of work called **sprints**, whose duration should not exceed the four weeks and should be more or less constant throughout the project. Sprints should be set in a way that at the end of each one a potentially releasable product increment is created. At the beginning of each sprint, in the so-called sprint planning, the development team selects the prioritary items of the product backlog that will be completed during that sprint. When the items are selected, minor **tasks** might be created for each one of them, in order to help developers estimate the time and effort that will take to complete each item. In the end, a backlog is associated with the sprint, the **sprint backlog**.

Besides the aforementioned sprint planning, three more formal events are described in the SCRUM process: **daily scrum**, **sprint review** and **sprint retrospective**. The former consists in a daily 10-15 minute time-boxed discussion for the development team to synchronize its activities and point out possible obstacles. The sprint review and the sprint retrospective are commonly scheduled to the same day, at the end of every sprint. The first gathers an informal presentation of the product increment and a primal definition of the product backlog items for the next sprint. Due to project environment changes or other variants, the product backlog can also be changed in these meetings. Lastly, in the sprint retrospective, an inspection with regards to people, relationships, process and tools is conducted.

In Figure 3.1, we present a simplified diagram of the SCRUM process, showing the timeline of the discussed concepts.

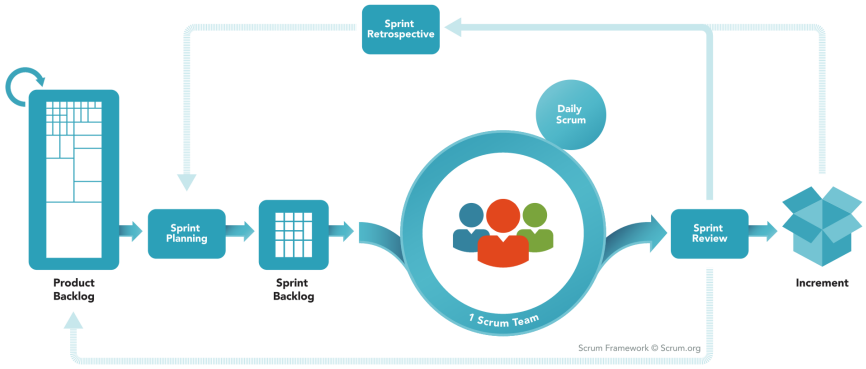

FIGURE 3.1: The SCRUM methodology [12]

In the first semester, daily scrum meetings with the supervisor took place, where he would be updated about the status of the project. In practice, these did not occur every day of the week because of university related deadlines and classes.
As for the sprint reviews and sprint retrospectives, they happened at the same time. The sprint duration was one week. In each of the meetings, the following members were

present: the intern, the product owner, the scrum master and the other intern who worked in natural language processing applied to a question-answer project.

In the second semester, the daily scrum meetings with the supervisor were changed to weekly meetings, which was sufficient. The product owner had a more representative role in this semester, and the intern also attended weekly meetings with him and with other members of the development team.

3.1.2 SCRUM Roles

A SCRUM team is composed by three main actors. The definition of each role and its attribution in the project scope is as follows:

- **Product Owner**: responsible for managing the product backlog, which includes the definition and prioritization of the desired features. In the big picture, they seek to maximize the value of the product and the work of the team. This role was played by Pedro Andrade, Head of New Product Development at WIT Software.
- **Development Team**: responsible for building the product required by the Product Owner. The team should include all the necessary skills to deliver the product and have a high degree of autonomy. Since this is an internship, these two requirements are already expected. There are, however, some variants that are also expected to happen, such as the lack of expertise for some specific tasks. In those cases, extra time and help from someone with more experience can be provided, so as to overcome the skill gap and any difficulties found. Another difference between the standard Scrum team and the internship team is the size. The team in Scrum is, usually, seven plus or minus two people, while in this project the development team consisted in the intern and one or two other developers that were assigned to the Bot Platform Project.
- **Scrum Master**: responsible for monitoring all of the Scrum process and protecting the team from outside interferences, ensuring the success of the project. This role was played by Jorge Sousa, the supervisor at WIT Software.

3.2 Work Plan

In this section we present the workplan of both semesters of the internship. A timeframe of the major tasks is presented and challenges that may have appeared are also explained.

3.2.1 First Semester

In the beginning of the first semester, some changes were made to the initial submited proposal. In the first version planning it was stated that four different bot prototypes would be developed, including news, weather and retail bots. In the early beginning of the internship the scope changed and only one prototype should be implemented: a bot for customer support assistance.

Since the first month of the internship was dedicated to state of the art and technologies exploration, that change did not have any impact in the planning. The Gantt diagram with the major tasks throughout the first semester can be seen in Figure 3.2.

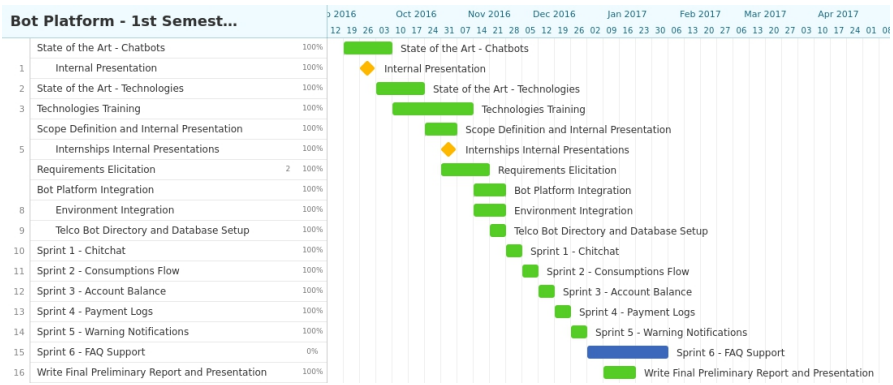

FIGURE 3.2: Gantt Diagram for the first semester

The planning always took in consideration the fact that during the first semester this was not a full-time internship due to the concurrent university courses that are mandatory. No major challenges happened in the first semester, and every task deadline was met. It should only be pointed that the integration with the platform was delayed a few days because the lead developer responsible for the platform went on vacation in the scheduled period. However, by taking some extra hours to study the code and the (scarce) documentation, things got balanced and the original plan was not altered.

3.2.2 Second Semester

The second semester was more focused on the human assistance interface, including the sentiment analysis module, and on the integration with the knowledge base that was being developed in another internship. The initial plan management is represented in Figure 3.3.

Although all the tasks that were proposed in the beginning of the semester were finished, some major changes occured in the timeline. Figure 3.4 represents the actual workplan that was followed.

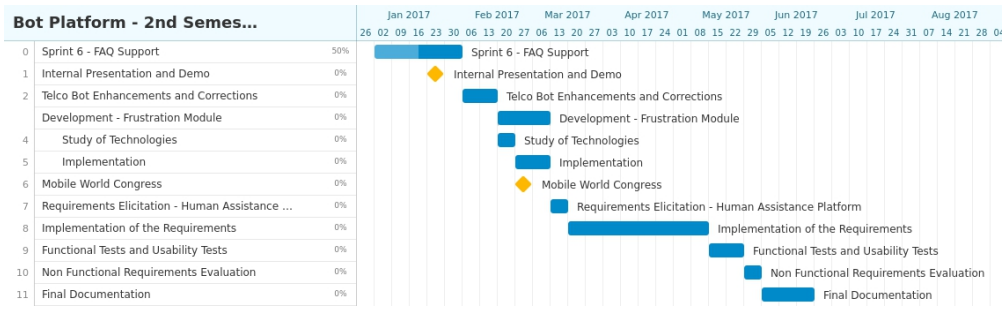

FIGURE 3.3: Planned Gantt Diagram for the second semester

The major difference was the shift between the sentiment analysis study and the implementation of the human assistance dashboard. After the World Mobile Congress 2017, the product owner informed that he would be involved in a series of demonstrations of the Bot platform, including the human assistance interface.

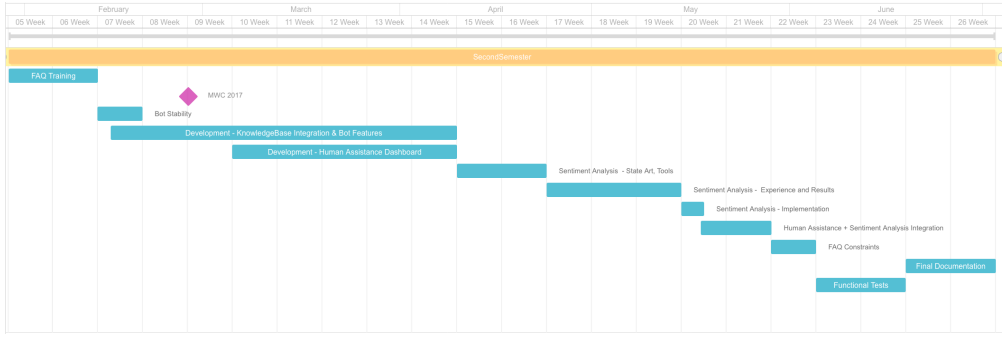

FIGURE 3.4: Real Gantt Diagram for the second semester

Additionally, in order to start testing the performance of the knowledge base being developed in the other internship, an integration process had to take part in the middle of February, which remained until the end of the year. The task of sentiment analysis, including the study of the state of the art, available tools and design of the experiment to choose the best one, took part mostly in the months of April and May. The task of functional testing, which had two weeks dedicated solely to it, ended up being a continuous process throughout the semester, given the necessity of having the system fully functional before each of the many demonstrations.

3.3 Risk Management

This section is dedicated to an important process that takes part in every software development project: the risk management. First we describe how that process is done and what is its importance and, in a second section, we describe the risks for the project.

3.3.1 Purpose of Risk Management

Risk management is a very important task in a project's development because it allows monitoring risks and minimizing their impact, if they occur.

Every software project has attached a set of risks which, when not correctly handled, might lead to its failure. A risk can be defined as an event or condition, related or not to software development, that, when it occurs, will probably result in a negative effect on the project's objectives.

Risk management is the process of identifying, analyzing, planning and monitoring risks. When done correctly, we can expect a minimization in their impact in the project, if they occur. The four stages of the risk management are described as follows:

- **Identify** - initial stage where the intern tries to understand if there are any factors that can spoil the success of the project. This process of risk identification happens periodically, since the project is not immutable and changes in the plan are allowed.
- **Analyze** - After identified, a risk has to be analyzed according to some attributes. The most common are the impact of the risk and the probability of its occurrence. To each one of the attributes, metrics have to be defined. The impact of the risk (representative of the level of threat it presents to the project) has the following measures:
  - **Maximum** - Risk that has the potential to greatly impact project schedule or performance. Success goals won't probably be achieved and failure is imminent.
  - **Medium** - Risk that has the potential to slightly impact project schedule or performance. Success goals can still be achieved but extra work will be required.
  - **Minimum** - Risk that has relatively little impact on schedule or performance. Success goals may still be achieved easily.

  As for the probability of the occurrence of a risk, it can have the following values:

  - **High** - probability of happening is greater than 70%.
  - **Medium** - probability of happening is between 30% and 70%.
  - **Low** - probability of happening is below 30%.

  It is also in this analysis phase that it is concluded if the risk can be mitigated or even eliminated.

- **Plan** - When a risk can be mitigated or eliminated, it must be defined a plan that explains how that can be achieved. This plan is elaborated in this third phase of the risk management.
- **Monitor** - The final step is to monitor the risks and updating them according to their evolution throughout the project development. For example, if a mitigation plan is not working, a new one must be formulated.

After having the risks identified and evaluated, it is common to display the results in form of a risks matrix. Although there exists standard risk matrices in different contexts [179], it is practical for individual projects to create their own matrices according to the measures that will be used.

The risk matrix for this project is represented in Figure 3.5. It is formed by nine categories, each one defining a level of danger of the risk. The color green marks the zone with lower danger and red marks the opposite. In the middle categories, yellow and orange are used to identify the risks that are neither catastrophic or negligible, but are in the middle.

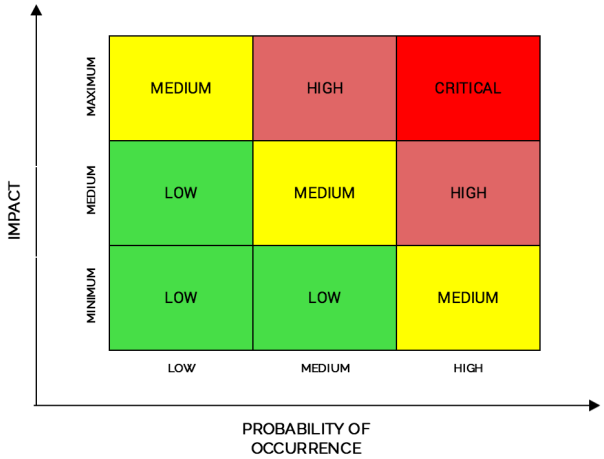

FIGURE 3.5: Risks Matrix - Level of danger for the project

3.3.2 Risks Description

In this section we describe the risks encountered for the project. As mentioned in the previous section, the process of risk management should occur periodically, with the existing risks being re-evaluated and new ones being added to the list. We now present all the risks identified throughout the project, and the re-evaluation of the risks can be consulted in Appendix B - Risks.

ID	**R01**
Date of Identification	November, 2016
Title	Maturity of the project
Description	The bot platform is at an early stage of development. Being a recent product, some people are currently working in a first stable version. Besides, there is not much documentation regarding the project. Getting to understand how the platform is organized and how to use it is a time consuming task.
Impact	Maximum
Probability	High
Mitigation Plan	Keep in contact with the team that is currently working on the project by scheduling meetings or chat intervals in order to receive helpfull information and feedback on how the platform works.

Table 3.1: Risk 01 - Maturity of the Project

ID	**R02**
Date of Identification	November, 2016
Title	New Technologies Learning
Description	Besides having to learn how to work with the platform, the intern has to deal with new technologies she is not familiarized with (NodeJS and NLU api's). Difficulties in using new technologies can cause delays in the tasks execution.
Impact	Maximum
Probability	Medium
Mitigation Plan	Train doing some tutorials and seeing some examples online. The intern will have the opportunity of doing experiments with the technologies before starting in the platform itself. Besides, more experienced colleagues at WIT Software are available to help.

Table 3.2: Risk 02 - New Technologies Learning

At the beginning of the project, when the first risk analysis was performed (November, 2016), the resultant risks matrix was the one represented in Figure 3.6. This initial analysis was the most important analysis of the project, and hence, it was the one for which the most formal and carefully planned process was taken into account. This is because at the beginning of a project there are many unexplored possibilities and

ID	**R03**
Date of Identification	November, 2016
Title	Ambitious Schedules
Description	It is expected that the prototype of the Telco Bot will be presented at the Mobile World Congress 2017, which will occur at the end of February. The deadline may be too ambitious for an intern not working on full-time, resulting in a lack of time to complete the goals and the need to work over budget.
Impact	Medium
Probability	Medium
Mitigation Plan	Ensure that tasks assigned to the intern are realistic and viable. Divide tasks into smaller tasks and reconsider the time given to the execution of each.

Table 3.3: Risk 03 - Ambitious Schedules

ID	**R04**
Date of Identification	November, 2016
Title	Dynamic State of the Art
Description	The hype on chatbots has been growing with no signs of stopping. The current state of the art will probably be in constant change, with new companies developing more natural language and artificial intelligence services that can make the one that was chosen obsolete.
Impact	Medium
Probability	High
Mitigation Plan	Being aware of new solutions available and evaluate if a best one has appeared and if it is viable to make a change. Change the requirements priority when needed. Develop modular code in a way that it can be easy to adapt to different APIs.

Table 3.4: Risk 04 - Dynamic State of the Art

ID	**R05**
Date of Identification	November, 2016
Title	Results regarding the QA system are not satisfactory
Description	Another internship is taking part at WIT Software with the intent of developing a reliable Question Answering (QA) system that can be integrated with the Telco Bot to perform better the search for the best answer to a given user question. The system may not be built at all or the results are far from satisfactory.
Impact	Minimum
Probability	High
Mitigation Plan	Have frequent meetings with the other intern in order to understand how the work is evolving. Besides, a backup solution must be implemented using the normal platform methods (NLU APIs and BPMN flows).

Table 3.5: Risk 05 - Results regarding the QA system are not satisfactory

ID	**R06**
Date of Identification	November, 2016
Title	Results regarding the NLU libraries are not satisfactory
Description	Because the NLU libraries are very recent (some of them are a few months old) there are not much guarantees that the results will be good enough.
Impact	Maximum
Probability	Medium
Mitigation Plan	Perform some prior experiments in order to choose the NLU API that is best suited for the project. If even the best one does not full fill the requirements, new strategies must be thought with the supervisor and the rest of the team.

TABLE 3.6: Risk 05 - Results regarding the NLU libraries are not satisfactory

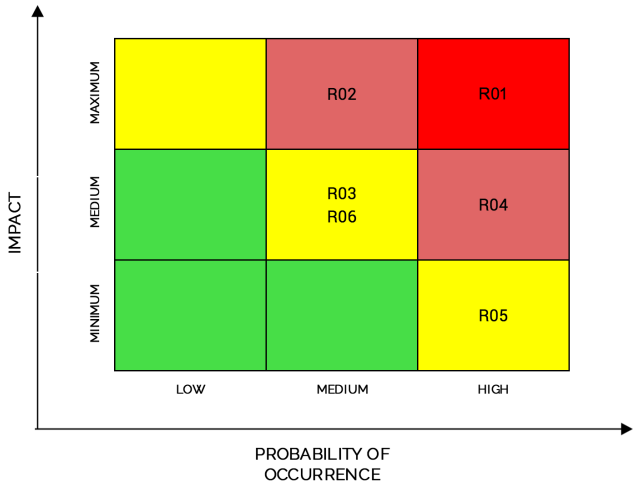

FIGURE 3.6: Risks Matrix - First Evaluation (November,2016)

incorrect decisions can have high costs in the later stages of software development. It is therefore critical to perform a risk analysis in the beginning of the project.

It is easily perceived that the majority of the risks fall into red or orange categories. This is a great indicator of how dangerous these risks seem at the beginning of the project and, for that reason, why it is so important to have a good set of mitigation plans to deal with them. For more informaton regarding the risk analysis performed during the project, please refer to Appendix B.

Chapter 4

The WIT Bot Platform

So far, we have explored how chatbots are a positive strategy for telecom operators when it comes to customer support. These virtual assistants can be easily reached through the most common messaging applications and provide support to users in need, all this without the additional uneasiness of installing new software on the phone and learning how to interact with it. In this section we described the architecture of the WIT Bot Platform, the product that will make that value proposition come true.

4.1 System Context Diagram

A context diagram is a useful starting point for documenting a software system, since it provides a look at the big picture and is the most abstract view of all. The focus is on people that interact with the main system (and how they interact) and on the services that it uses. The contextual diagram for the WIT Bot Platform is shown in Figure 4.1.

Users that want to interact with the chatbots built with the Bot Platform must find them and talk with them through one of the platforms' supported languages (Facebook Messenger or RCS app, for example). There are bots that can be used without registration, and others which require it. In the second semester, the Telco Bot started to require a previous registration before engaging conversation with the user, whatever the question was.

The Bot Platform itself does not hold all the information needed to answer all kinds of requests. In order to attend service requests, the Telco Bot must be integrated with the operator's service infrastructure, in order to have access to private user's information. Other external services can also include 3rd party bots whose APIs are free to use. Currently, this is not a common practice, but it is expected that in the meantime

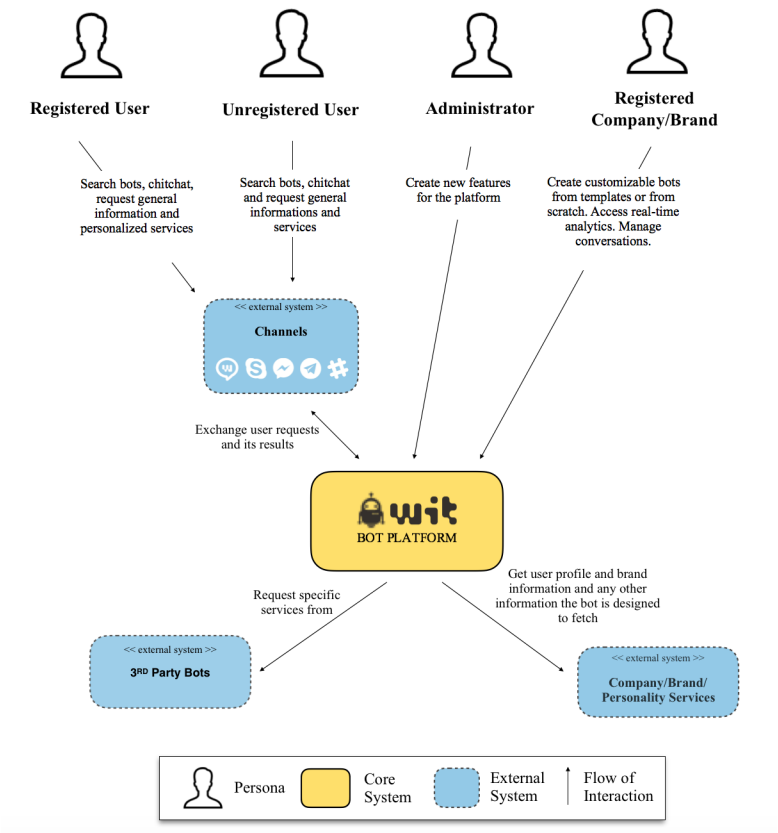

FIGURE 4.1: Context Diagram of WIT Bot Platform

more and more bots will start to open their endpoints and even their source code, especially after big names like Mark Zuckerberg have shared their vision about collaborative programming in chatbots [180].

The personalities and companies interested in building their own bots using the platform interact with it through a Web user interface. They can start by using the pre-built bot templates that the platform will offer (for Customer Support, Enterprise Productivity and Retail) or start one from scratch, using the interface to build new workflows. After having one or more deployed bot, they can follow its progression through the real-time analytics and provide human assistance to the conversations that need it.

4.1.1 Physical Diagram

The nomenclature 'WIT Bot Platform' actually refers to a set of three big components that connect between each other and some external services. Those components are the Bot Platform core, the artificial intelligence service's and the dashboard that allows the manipulation of the former modules, besides other features. In Figure 4.2 one can see represented those components and the most important modules of each one for the context of the internship.

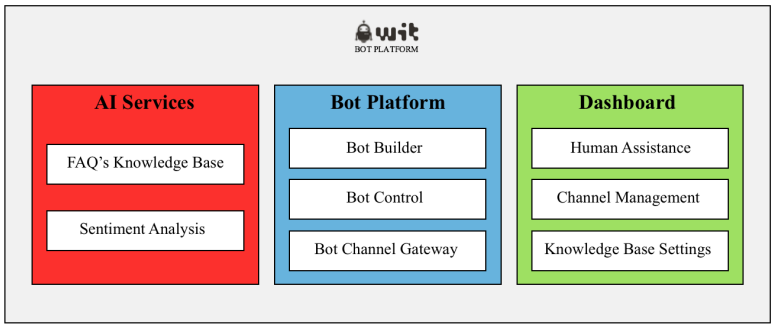

FIGURE 4.2: Major entities involved in the WIT Bot Platform

- **AI Services** - all the modules that concern machine learning algorithms or other artificial intelligence approaches are stored in the same machine. It is where the sentiment analysis web server is implemented and where the knowledge base web server, developed by another intern, is hosted.
- **Bot Platform** - this is the core component of the project, where all the conversation handling and business logic happens. The attributes that define each bot are stored in this component in individual folders. It is also in this component that the Bot gateways are defined.
- **Dashboard** - this component comprises the web interface that allows some manipulations in the bot platform such as the availability of each bot and channel management. Other features related with the internship are the knowledge base definitions (what FAQS to include, manipulation of those FAQS, creation of constraints and refinement of the machine learning algorithm) and the human assistance interface, where the operator's can follow the conversations and perform actions such as engaging bot flows and choose the right answers.

The three modules interact between them and other services as represented in Figure 4.3. Two external services are also identified: one is the MyVodafone gateway where the platform posts requests about the user's account (current balance, tariff, and others) and the other is API.ai, an NLP service that is trained to recognize intents related with the service component of the bot (ask for invoices and ask for the last debts, for example). For reasons of confidentiality, more detailed information about the architecture is described in Appendix A.

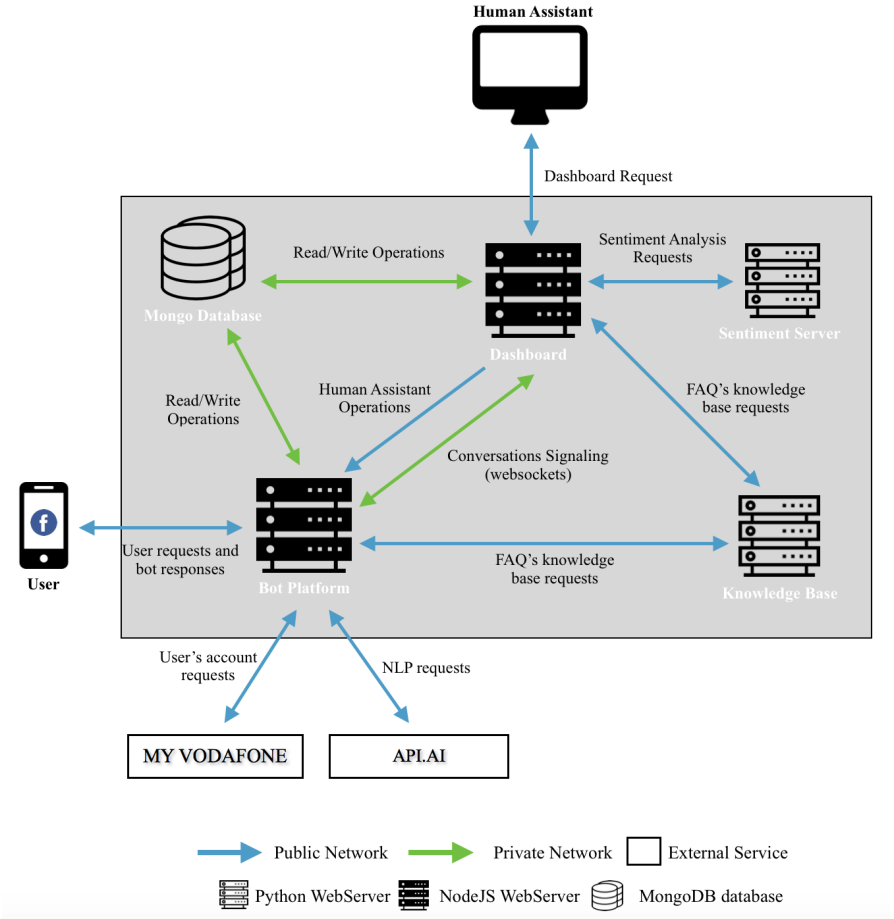

FIGURE 4.3: Physical diagram of WIT Bot Platform

There are multiple possible flows of actions, one example being:

1. User asks a question to the Telco Bot on Messenger
2. Bot platform receives the text and sends it to the dashboard through a websocket
3. Dashboard requests sentiment analysis of the input and updates the conversation's information
4. Bot platform sends the input to API.AI
5. No service intent is identified so the platform sends the input to the FAQ's knowledge base
6. Knowledge Base does not find a perfect match returning instead a set of possible questions to figure in the quick replies
7. Bot platform searches the AIML files in order to check if the input matches any chitchat pattern

8. No matching is found so the platform returns a 'non-understood' message followed by quick replies with the answers returned from the knowledge base
9. Bot platform sends the reply to the dashboard via websocket
10. Dashboard updates the conversation's information

These actions, along with the other possible flows of interaction, will be explained with more detail in the remainder of this chapter and in Chapter 6

4.2 Bot Platform

The contextual view allows the understanding of the interactions between personas, external services and the system. In this subsection we perform a zoom into the system context and present the high-level technology and the main containers of the Bot Platform container. Its components are shown in the diagram represented in Figure 4.4.

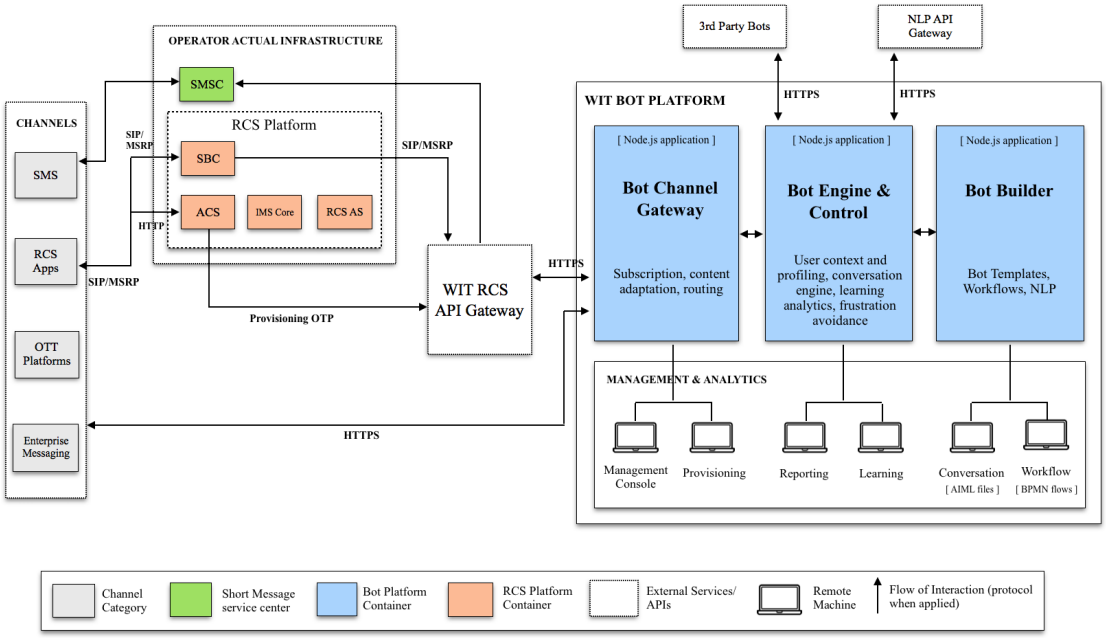

FIGURE 4.4: Components Diagram of the Bot Platform

As is visible in the diagram, the intention is that the bot platform is able to deploy its bots in four categories of channels, including the SMS, OTT platforms and the RCS apps. OTT platforms refer to the over-the-top applications such as Facebook Messenger, Telegram and Slack that allow the sharing of instant messages (text and media) via the Internet. With the increase of popularity of this kind of applications (see Section 2.1.1),

telecom operators started to lose a lot of the revenue that came from SMS and calls exchange. In order to mitigate that problem, a new solution was implemented among some operators, called the RCS (rich communication services) apps. Created by the GSM Association [181], in 2008, the RCS allows communications via SMS or voice over the internet (using Wi-Fi or the operator network) and more extra functionality such as chat, file sharing and group messaging. The RCS is marketed under different names according to the operator. For example, in the case of Vodafone, the RCS app is called Message+.

AIML files are used in the platform to define how a bot should respond to a question using pairs of <pattern> and <template> elements. The pattern represents the user's question, and the template defines the bot's possible answers. In Figure 4.1, we present an example of two AIML categories intended to reply a greeting to the user.

```
<aiml>
    <category>
        <pattern>HELLO</pattern>
        <template>
            <random>
                <li>Hello to you too</li>
                <li>Hey there!</li>
                <li>Hi, good to see you!</li>
            </random>
        </template>
    </category>

    <category>
        <pattern>HI *</pattern>
        <template>
            <srai>HELLO</srai>
        </template>
    </category>
</aiml>
```

LISTING 4.1: "Example of AIML categories intended to greet a user"

When the user types "hello", the pattern "HELLO" is activated and one of the possible answers is chosen. If, instead, "Hi there bot" is typed, it will fall in the pattern "Hi *" and, therefore, AIML will recursively evaluate the text and replace it with the response (in this example, the next recursion call would be at the pattern 'HELLO').

Another important part to define regarding the chatbots are the conversational workflows to follow when a special task is triggered. The AIML scripts are good for general question-reply chitchat interactions, but not so good with more complicated situations. For example, when a user asks "What is the price of this application?", there is no right answer to give because we don't know what application is being mentioned. Instead of replying something generic like "I did not understand your request, please try to

rephrase it", the bot should understand the primal intention of the request and start asking questions until it has all the data that will allow it to fetch the right answer ("Of what application are you talking about: A, B or C?").

The chosen mechanism for building these flows was the Business Process Model and Notation (BPMN). BPMN has been the global standard for business process modeling for a while now. It provides an intuitive graphical notation for specifying business processes based on a flow charting technique. It is supported by many software products, making developers and businesses less dependent on any particular vendor's products. Basically, BPMN fills the communication gap between the business process design and the process implementation. More detailed information about the definition of the BPMN files will be provided in the following sections.

The interaction with the Bot Platform happens via HTTP requests. They can either come directly from the messaging platforms or indirectly from the operator's RCS gateway (this intermediate step is necessary for the SMS and RCS applications).

The three components are detailed in the next sub-sections.

4.2.1 Bot Channel Gateway

This is the container responsible for receiving HTTP requests in the multiple available endpoints and routing them to the appropriate next step and also sending the HTTP responses back to the same channel of origin. It is also responsible for managing the content adaptation, which means that it is in this container that the formats to deliver based on the destination are decided.

The need for this content adaptation exists because the chatbots' user interface (UI) is dynamic and does not rely solely on written text. As mentioned in Section 2.1.2, chatbot interfaces are moving from just conversational to hybrid, combining both text and graphical elements to drive the conversational flow. In Figure 4.7 we present the two most common graphical elements in messaging apps such as Facebook Messenger and the Android RCS app.

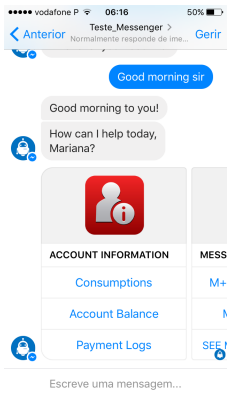

FIGURE 4.5: Carousel Template

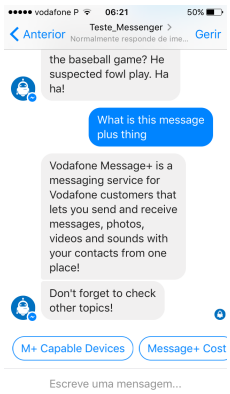

FIGURE 4.6: Quick Replies

FIGURE 4.7: Bot UI Elements

There are channels, like SMS, that do not support this type of rich text. When that is the case, the Bot Gateway has to perform some rearrangements in the built message, like extracting all the textual elements from the graphic elements and create a "super text message" combining all of the them.

4.2.2 Bot Engine And Control

This is the core component of the Bot Platform. It is here that very important features will be hosted such as the integration with 3rd party bots and services, user context and profiling (module that allows bots to have access to user profile information, like active services and subscription info), learning analytics (for providing key indicators for bot analysis, like Engaged Users, Lost Users, among others), bot firewall (for definition of rules of push messages and spamming actions) and, of course, the module of the conversation engine. Alongside with the Bot Builder, this is one of the containers where the the internship tasks were more focused on, namely the user profiling and the integration with the NLP services.

The most important task that has to be performed by this component is the conversation engine. When a new message arrives to the bot, it decides what to do with that message and defines the flow of actions to be taken until the reply message is sent back to the user. The sequence diagram presented in Figure 4.8 illustrates the overall flow for fetching an answer for a given user utterance.

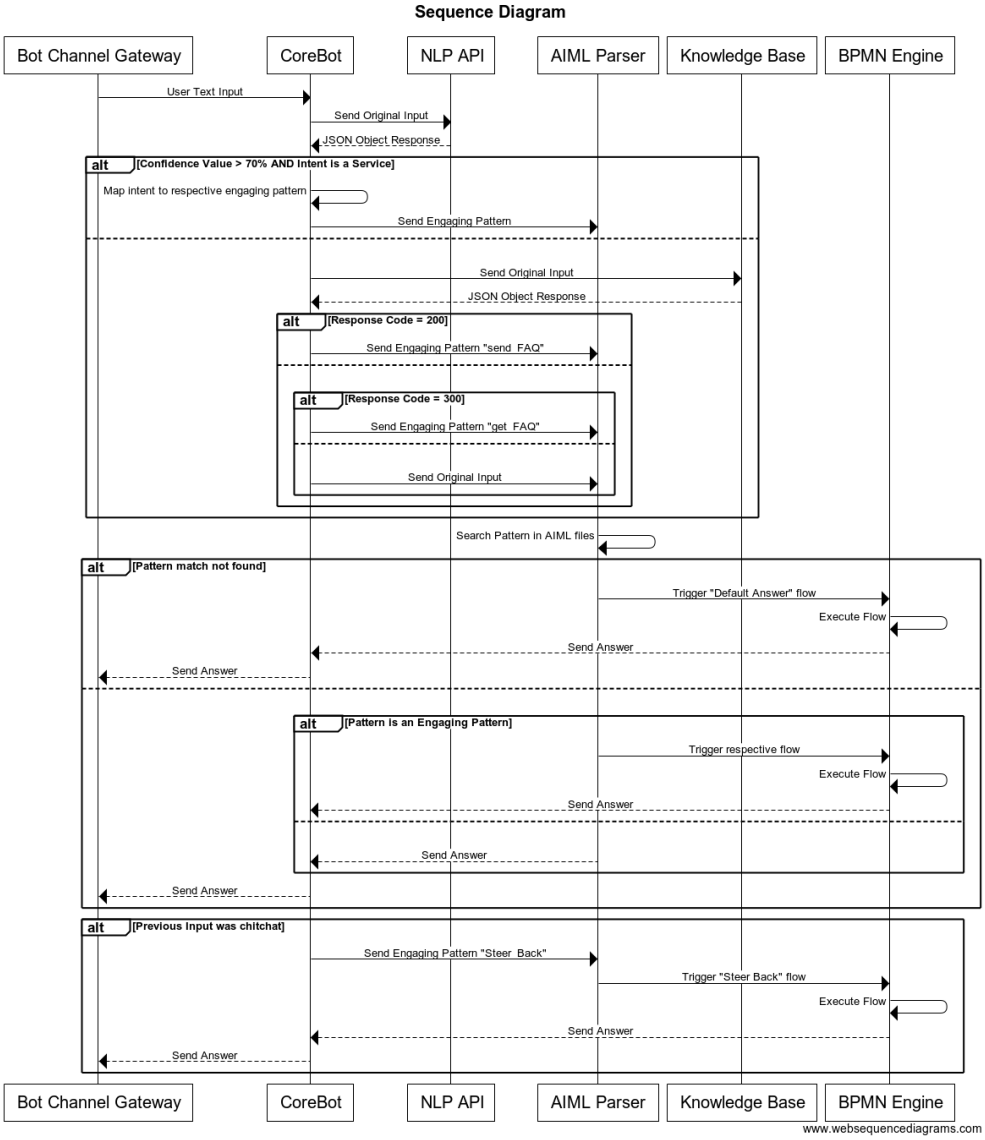

FIGURE 4.8: Sequence Diagram for searching the best answer to a user input

The flow that is current implemented is described as follows:

- **Message arrives from the Bot Channel Gateway**: the CoreBot module is the main core of every chatbot in the platform (each bot has a respective CoreBot). Every message that arrives or leaves the system has to pass in this module. It is also in this module that the integrations with other services - databases, NLP APIs, etc - are done. The sequence diagram starts when a new user message arrives for that bot, from any of the available channels.

- **NLP request**: if an NLP service is currently set, the Core Bot will send an API request for the service with the user utterance. In the case of the Telco Bot, the default NLP service is API.ai. The NLP will then reply with its respective JSON

response object which contains the information about the intent and the entities identified and the level of confidence of that matching. If no NLP service is set at the moment, the CoreBot will send the original input into the AIML Parser module.

- **NLP Response Evaluation**: the NLP response object includes a value of the confidence level with which the intent was mapped to the user utterance (if there was an intent at all). When that confidence level is not high enough (meaning it is below 80%), the CoreBot redirects the original input to the Knowledge Base module. After confirming that the intent identified matches an actual service that the bot is providing at the moment, the CoreBot sends the respective engaging pattern to the AIML parser.

- **Knowledge Base request**: if the user is not asking for a service, the next step is to find whether the input matches a FAQ. As so, a request is sent to the FAQ's knowledge base endpoint and the result is interpreted. If the response code equals 200, it means that a match was done, and the correct answer is returned. If the code equals 300, it means that a matching question was found but, in order to give a specific answer, some questions have to be made to the user. Otherwise, it means that there was no match, and the top ten questions which had the highest matching scores are returned.

- **AIML Parser searches pattern**: this is the module responsible for receiving a pattern and look for the respective template in the AIML files. If it cannot find a match, a default flow is triggered, responsible for sending a 'did not understand' message to the user. Otherwise, the next direction depends on whether the text is a chitchat or an engaging pattern. In the case of the former, the template is replied right away, and the 'steer back' flow is triggered. If it is an engaging pattern, the AIML Parser sends a message to the BPMN Engine to start or continue the flow.

This sequence of steps is specific of the Telco Bot. At the time this internship was being finished, all the other bots trained all of their intents in an NLP provider or used AIML patterns. Because of that, the corespondent sequence flow is much more concise.

4.2.3 Bot Builder

Bot Builder is the component dedicated to the creation and storage of the bot conversational capabilities and other personal characteristics. Each chatbot on the platform must follow a pre-defined architecture in order to be correctly integrated with the platform. That architecture consists of the file system represented in Figure 4.9.

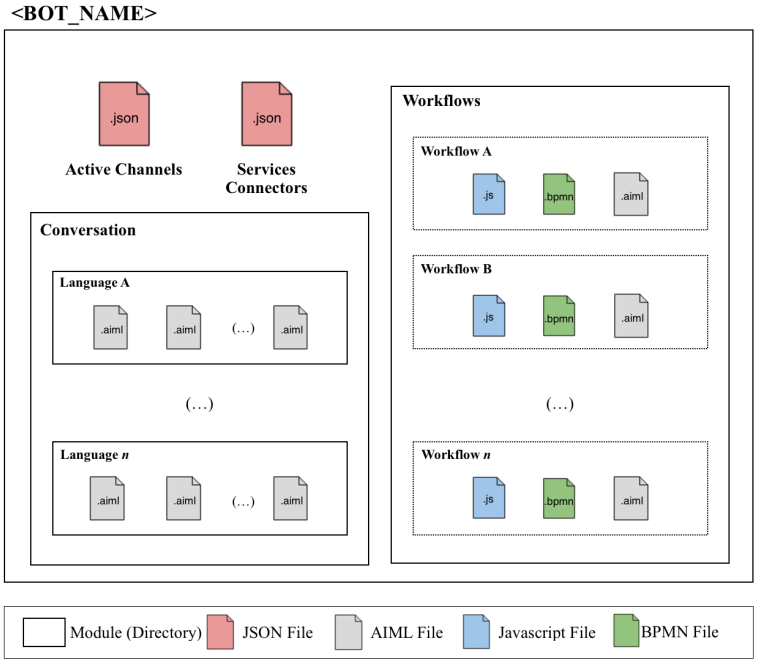

FIGURE 4.9: Required file structure for each bot

A brief description of the three main components of the file structure is as follows:

- **Active Channels:** a JSON file that contains the channels where the bot is currently deployed and the respective configurations.
- **Services Connectors:** a JSON file that contains the external services that the bot needs (a database and a NLP provider, for example) and the respective configurations.
- **Chitchat:** in this directory are stored the files reserved for chitchat interactions, divided by the natural language in which they are written. Those files consist of AIML categories that can be matched to the user input through basic pattern matching, each one with the respective pre-defined answer to return to the user. The developers have the freedom to create AIML rules for any topic they want, in any language. The Bot Platform already provides some ready-to-use chitchat files in English and Portuguese that cover some generic topics. The last count marked 11637 patterns in the English dictionary. Table 4.1 gives a brief description of each of the files.

ID	Topic	Description	Examples of categories
F1	Artificial Intelligence	General answers about artificial intelligence, AIML, famous chatbots and robot features	"Who created AIML", "When will you die", "Are you a * ELIZA", "Who is Agent Ruby"
F2	Astrology	General answers about the horoscope	"AQUARIUS", "What is a Capricorn"
F3	Atomic	General answers about random topics and statements	"Have you been drinking", "Amen", "Bravo", "Do you know Pokemon", "Go to sleep"
F4	Biography	One-phrase biographies of some popular personalities and characters	"Who is Madona", "Who is Fidel Castro", "Who is R2 D2", "Who is Alan Turing"
F5	Bot	General answers about general statements and questions about the bot capabilities and personality	"You are cool", "You are not impressive *", "You must be American", "Do you like Titanic"
F6	User Traits	General answers about topics the user introduce about himself	"I have blue eyes and*", "I have faith", "I am poor", "I am a horrible person"
F7	Drugs	General answers about drugs usage and smoking	"Do you smoke *", "What is weed", "Do you like pot"
F8	Emotions	General answers about emotions	"You are jealous", "Have you ever been in love", "What makes you sad", "Tell me about love"
F9	Food	General answers about food, drinking and electricity	"What does electricity tastes like", "Drink *", "What do you eat", "Do you like lemons"
F10	Geography	General answers about countries, cities and capitals	"What is the capital of Syria", "California is *", "Where is Russia", "What is the population of USA"
F11	History	One-phrase facts about historic events and personalities	"American Civil War *", "Who invented the steam"
F12	Jokes	Compilation of quick jokes	"Jokes", "Tell me a joke", " * a joke"
F13	Knowledge	General answers about a little bit of everything	"What is the lifespan of a dog", "What is CIA", "What month comes after February", "What color is the sea"
F14	Literature	General answers about books and literature	"Have you read books", "Leo Tolstoi", "Who wrote the Hobbit"
F15	Money	General answers about money and economy	"What is a dollar", "How much do you earn", "Do you accept money"
F16	Movies	General answers about movies and cinema in general	"You sound like Hal", "Who is Spider Man", "Lord of the Rings"
F17	Music	General answers about music	"Do you play a musical instrument", "Why is * your favorite band", "Who is Beethoven"
F18	Science	General answers about biology, physics and chemistry	"What is cytology", "How far is the moon", "What are the laws of thermodynamics"

TABLE 4.1: Description of the chitchat files provided by the platform

- **Workflows:** in this directory are the files that define the flows that the conversation should follow when a major task is triggered. As shown in Figure 4.9, each flow is composed by a **BPMN file** (description of the flow), a **Javascript file** (backend logic for each element of the flow) and a **AIML file** (text replies to send to the user). One of the major intents of the Bot Platform is to ease the overall process of creating a chatbot without spoiling its capabilities and power. As so, anything that is related to platform standards should be left as little as possible in the hands of the bot developers, allowing them to focus mostly on the backend logic. With that in mind, a solution was created that, receiving a BPMN file as input, creates and auto-fills the Javascript and AIML scripts of the flow.

  Since the second version of BPMN, released in 2010, it has been possible to serialize BPMN diagrams and provide XML schemes to describe them. In order to correctly generate the files, some extended attributes must be added to the flow definition. The attributes can either be added manually to the file (which can be hard, time consuming and easy to make mistakes) or by using a BPMN software tool that also

implements the XPDL notation (a standardized notation to interchange process definitions between workflows). The second technique is the one that is currently used by the bot developers of the platform. After exporting the BPMN and the XPDL files and recreating the first using the extended attributes in the second one, a simple script can be run in order to generate the templates of the AIML and Javascript files. A visual representation of the process is represented with an example in Figure 4.10.

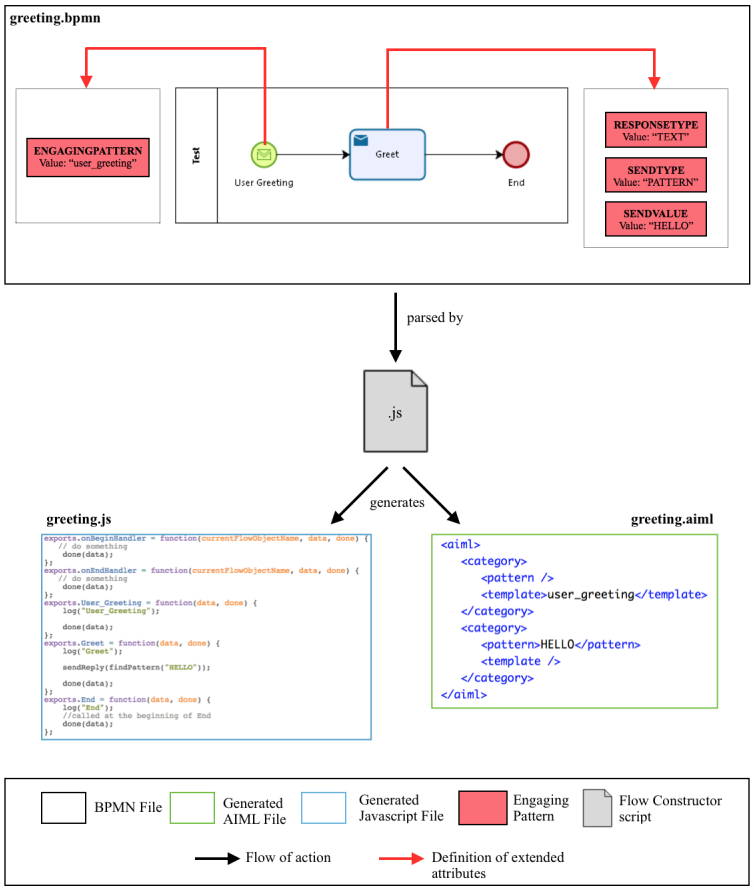

FIGURE 4.10: Generation of AIML and Javascript scripts from BPMN file with extended attributes

This is a very simple flow just shown as an example. When the user input is classified as a "user_greeting" intent, this flow is triggered by the Bot Engine container and it moves on to the next element. "Greet00 is a Send Task that sends to the user a text message whose content is the AIML matching for the "HELLO" pattern.

The generated AIML scripts will have one category for each ENGAGINGPATTERN attribute and each SENDTYPE with the value "Pattern". This last one must be always paired with a SENDVALUE, since it is the attribute that defines the pattern tag. The template tag is left for the developer to fill in with whatever

he wants. As for the Javascript files, they are filled in order to be compatible with the Node.js "bpmn" module [182], the module used in the Bot Engine container to execute BPMN 2.0 processes.

This solution, however, is not ready yet to handle every BPMN event. Currently, it supports start and end events, send, user and service tasks, exclusive gateways and timer events. In the future it is expected that support for parallel gateways and subprocesses will be implemented.

A more in-depth view of the architecture of the Bot Platform is available in the Appendix A.

4.3 Human Assistance Dashboard

The other major entity of the WIT Bot Platform related with the customer support context is the Human Assistance Dashboard, a sub container of the general dashboard that manages the Bot Platform component.

The technologies that are involved in its implementation are mainly four: HTML and CSS, used for structure and style the the webpage, Javascript, for the business logic, and AngularJS for handling dynamic events in a page. Angular [183] is a javascript framework that extends traditional HTML to present dynamic content through two-way data-binding that allows for the automatic synchronization of models and views. It implements the MVC pattern to separate presentation, data, and logic components. The interaction between the browser, the AngularJS and the NodeJS server is represented in Figure 4.11.

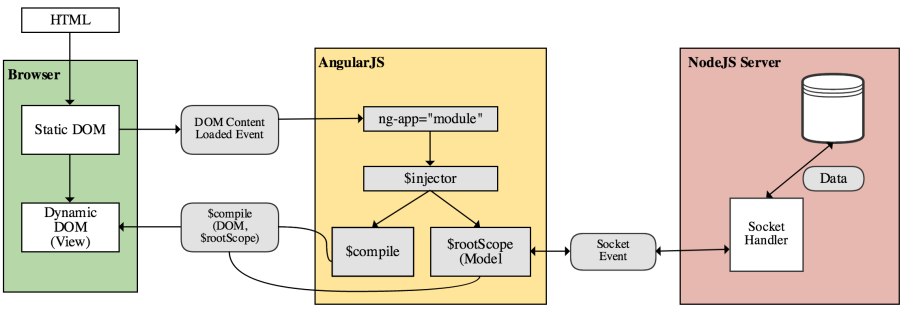

FIGURE 4.11: Interaction between the browser, AngularJS and the Node server

A brief description of the main features of AngularJS is presented next:

- **Data-binding** - the automatic synchronization of data between model and view components. The model detects changes using algorithms of dirty checking and

modified HTML expressions in the view accordingly. Likewise, any alterations to the view are also reflected in the model.

- **Scope** - is a Javascript object that refer to the model where all variables defined in it can be accessed by the View as well as the controller.
- **Controller** - Javascript functions that are bound to a particular scope.
- **Dependency Injection** - a software design pattern that allows the creation of dynamic references to certain components without needing to explicitly declare them. One of the most popular is the service $https: for making HTTP requests.
- **Directives** - markers on DOM elements (such as HTML elements or CSS) that allow the creation of new elements with any intended properties. The most common are the *ng-app*, that initialized an AngularJS application, *ng-init*, that initializes application data, and *ng-model*, that binds the value of HTML controls (input, select, for example) to application data.

The communication between the model and the NodeJS server is established via web sockets, allowing the exchanging of data without having to reload the whole page. Web sockets use a bidirectional channel where both the client and the server can send messages at any time, by using socket events. More details about those events and the overall architecture of the Dashboard can be found in the Appendix A.

Chapter 5

Requirements Elicitation

The requirements elicitation is one of the most, if not *the* most, critical stage in software development. Fred Brooks, a well known software engineer, wrote in [184] that "The hardest single part of building a software system is deciding precisely what to build". More important than knowing how to build the system is knowing what purposes the system is intended to attend. In this section we detail the process of requirements elicitation for the prototype of the Telco Bot and the Human Assistance dashboard, along with the user stories. A detailed description of the requirements is available in Appendix C.

5.1 User Stories

One of the most popular techniques for defining requirements is by creating user stories. A user story is a short and simple description of a functionality that will be valuable to the user/purchaser/software that desires the capability [185]. They are often written using the formula "*As a <type of persona>, I want to <action> so that <outcome>*".

User stories are part of an agile approach that usually precedes the functional requirements definition process. When the user stories are grouped by the different users and points of view, it becomes easier to understand them and to guide the development phase. To find the user stories, some meetings were scheduled with the product owner of the project in order to define what would be the major services that the Telco Bot should provide. Other user stories not directly related to the user were defined according to the intended architecture design, to the analysis of what are the currently most appreciated conversational traits in chatbots [72] and to some personal opinions.

User Stories			
ID	"As a"	"I want to"	"So that I"
US1	**User**	Find the Telco Bot in Facebook Messenger	Start a conversation
US2	**User**	Find the Telco Bot in the RCS app	Start a conversation
US3	**User**	Get an overview of what the Telco Bot can do	Be aware of its capabilities
US4	**User**	Do some basic small talk with the bot	Can have the feeling of having a real conversation
US5	**User**	Get information about my SMS consumptions	Can control them
US6	**User**	Get information about my Calls consumptions	Can control them
US7	**User**	Get information about my Data consumptions	Can control them
US8	**User**	Get my last invoice	Can consult it
US9	**User**	Get information about my current tariff	Can have the information
US10	**User**	Get information about unpaid invoices	Can control them
US11	**User**	Get information about my current balance	Can control it
US12	**User**	Know when was my last charge	Can control my charges
US13	**User**	Know when was the last debt	Can expect the next one
US14	**User**	Register in the bot	Can access my personal information
US15	**User**	Get access to FAQS about phone lost and repair	Can find the answers to my questions
US16	**User**	Get access to FAQS about changing phone number	Can find the answers to my questions
US17	**User**	Get access to FAQS about network coverage and problems	Can find the answers to my questions
US18	**User**	Get access to FAQS about MyVodafone application	Can find the answers to my questions
US19	**User**	Get access to FAQS about bills	Can find the answers to my questions
US20	**User**	Get access to FAQS about SIM cards	Can find the answers to my questions
US21	**User**	Get access to FAQS about SIM cards	Can find the answers to my questions
US22	**User**	Get access to FAQS about account management	Can find the answers to my questions
US23	**User**	Get access to FAQS about data control	Can find the answers to my questions
US24	**Bot**	Search and provide right-ahead the most probable service the user wants when he asks for a general task	Can prevent unnecessary steps
US25	**Bot**	Be trained to understand the user's service requests	Can fulfill the users interests
US26	**Bot**	Be connected to FAQ's knowledge base	Can find the answers to FAQ-related questions
US27	**Operator**	Send text messages to the user in name of the bot	Can answer to the user's requests more directly
US28	**Operator**	Engage one of the bot's flows	Can provide the right answer to the user
US29	**Operator**	Teach the bot what question the user input was referring to	Can teach the bot for future similar interactions
US30	**Operator**	Ask about the user's context	Can have access to the right answers to give
US31	**Operator**	Have access to the current sentiment level of the conversations	Can choose the conversations that need human intervention
US32	**Operator**	Be warned when a user is confused or angry	Can intervene immediately in those conversations

TABLE 5.1: User Stories

5.2 Functional Requirements

As aforementioned, expressing requirements in the form of user stories was introduced by the agile approaches to software development. Because they emphasize verbal communication over written language, it becomes less likely that customers and developers interpret a statement in different ways, as often happens with ambiguous statements. A second advantage of user stories is that they ease the process of project planning: they are written in a way that each team member can quickly estimate how time-consuming

the task will be. Additionally, each story is implemented all in a single iteration of an agile project (or a sprint).

There is no standard template to describe requirements, although use cases have become the most popular method worldwide. When they were first formulated, in 1986, they appeared under textual, structural and visual modeling techniques, and since then other contributions have been made overtime. The most notable was probably by Alistair Cockburn with his templates based on visual symbols for design and goal levels, published in the famous book 'Writing Effective Use Cases' [186]. The final requirements description that was made combined both features from the Cockburn's templates (leaving some of them out, like the design and goal symbols) and WIT's own template for documenting a system's functional behaviour. From the long-term discussions throughout the year with the product owner, each requirement was also assigned a priority in order to state the importance they place on the delivery of the project. The nomenclature used was based in the MoSCoW method [187] which goes as follows:

- **Must Haves** - requirements that are critical to the current delivery timebox in order for it to be successfull. It is the minimum scope for the product to be useful.
- **Should Haves** - requirements that are important but not necessary for delivery in the current delivery timebox.
- **Could Haves** - requirements that are desirable but not necessary. Most of these features could potentially be included without incurring to much effort or cost.
- **Won't Haves** - requirements that have been requested but are explicitly excluded fr the next delivery timebox.

Each requirement comprises, beside the priority label, seven other parts:

- **ID and Name** - unique identifiers of the requirement.
- **Description** - brief description of the requirement
- **Primary Actor** - who is the stakeholder that will trigger the requirement.
- **Pre-Conditions** - conditions that the system will ensure are true before the respective use case starts.
- **Post-condition Success** - the outcome from a successful interaction.
- **Main Success Scenario** - the main steps that will occur if everything goes as expected (the ideal scenario).

- **Extensions** - the alternative steps that the system performs when one or more steps from the main scenario does not go as expected.

Table 5.2 represents a functional requirement described as mentioned. The remainder of the requirements are detailed in the Appendix C in order not to overload the core of the document.

RF 01 - Log in	
Primary Actor	Non Registered User
Description	User enters his Vodafone credentials in order to start using the bot features
Priority	Must Have
Pre-conditions	User has access to a messaging application where the Telco Bot is available. User has access to an internet connection. User has MyVodafone valid credentials.
Post-condition success	User is logged in the system and is able to ask questions
Main Success Scenario	1. User attempts to start a conversation with the bot 2. Bot returns a login message and a URL for subscription 3. User clicks the URL 4. User inserts his Vodafone credentials in the web form that opens 5. Credentials are validated and user is returned to the conversation 6. Bot sends a greeting message
Extensions	3a. User ignores URL and keeps trying to engage conversation 3a1. Bot keeps returning the same message and the same URL 5a. Credentials are wrong 5a1. Error message is shown and the form stills open

Table 5.2: Functional Requirement 01 - Log In

Some requirements did not show up in the final version of this work, namely C.20 and C.21. They were not part of the initial internship proposal and appeared late in the project, making it impossible to include them in the final prototype, given the amount work that needed to be done already.

Chapter 6

Implementation and Work Done

This section is dedicated to presenting the system that was developed during the year, the major implementation tasks and possible challenges that may have occured. The development phase was not a continuous process during the internship because, from time to time, new requirements would appear and new architecture and design patterns had to be determined. Because each major component of the system - bot platform, dashboard and sentiment server - had different development approaches, they are explained in the remainder of the section separately.

In the next section we present the prototype of the Telco Bot implemented in the first semester of the internship. As mentioned in Section 4.2, the important tasks of a bot are defined by BPMN workflows with specific extended attributes. Those workflows and a more detailed description of each one is provided in the Appendix E.

6.1 Telco Bot

The Telco Bot was the first major outcome of the internship: a chatbot that is deployed in popular instant messaging applications and that is able to provide information and support about an operator's services.

The bot - later dubbed as "WIBY" - was integrated in an earlier project that WIT started a few months before the internship, the Bot Platform. At the beginning of the internship, the platform was little more than a NodeJs web server that received messages from Facebook Messenger and returned pre-defined answers written in AIML files. More than implementing a new chatbot, the contributions of this work include the creation of new conversational flows and designs and the study and integration of state-of-the-art NLP tools that take a step beyond the reliance on scripting languages.

The Bot Platform is written in NodeJs, a server side JavaScript environment focused on performance and low memory consumption. It not only is the favorite language to develop chatbots at the moment but it has also become one of the most popular frameworks for implementing webapps, mostly because of its asynchronous I/O event based model and the best performance when compared to other server side technologies such as PHP/Apache and Nginx [188].

Each bot of the platform is assigned to a Contextual Database, a MongoDB database with two main collections: USERS, where is stored all the information about users that have engaged conversations with the bot, and REQUESTS, where is stored the information about users's authentication attempts. Each bot can have additional databases in order to save other information that may be relevant to its context. The Telco Bot has another Mongo database where is saved the information about the available services and their respective engaging pattern.

6.1.1 NLP Engine Integration

In Section 2.2 we presented the current state of the art on technologies and methodologies to add natural language capabilities to chatbots. Scripting languages can only lead us so far, and, thus, different approaches needed to be studied and tested. The API.ai service was chosen as the best external natural language understanding tool and, with time, all the bots that were being developed started to also integrate with it.

A new module was created to allow the CoreBot module to deal with these kind of actions in a transparent way. The interaction is as represented in Figure 6.1.

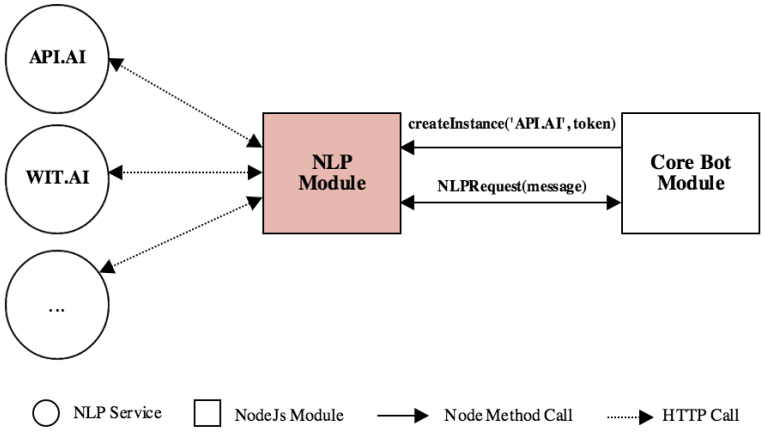

FIGURE 6.1: Module for abstract NLP actions handling

Currently, the module includes methods for connecting to a bot profile, requesting a text analysis, getting all the available entities and getting an entity by name. The first two

are available for both API.ai and Wit.ai services while the last ones are only implemented for API.ai.

6.1.2 Chitchat Interactions

The Telco Bot should be able to provide some basic chitchat interactions with the user. This component exists not only because of the AIML files provided by the platform (see Section 4.2.3), but also because of the small-talk domain that can be activated in the API.ai platform. Whenever API.ai detects an input with the intent of small talk, the platform replies to the user with the recommended answer returned by it.

Figure 6.4 shows some interactions with the Telco Bot. The first one shows a normal conversation flow and the second one shows the usage of some of the bot capabilities for entertainment purposes.

FIGURE 6.2: Conversational small talk

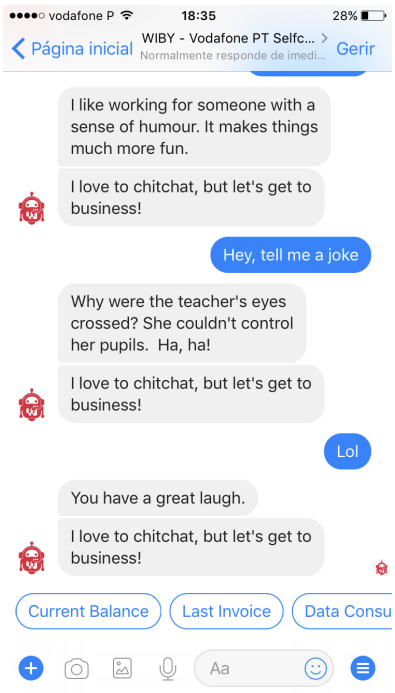

FIGURE 6.3: Bot capabilities

FIGURE 6.4: Telco Bot Chitchat Interactions

In the second semester, a new requirement appeared and was implemented. This requirement stated that after each chitchat interaction, the bot should return a steer back message, calling the user's attention to the bot's purpose and the services it can provide. This was accomplished with the creation of a new BPMN flow that is available in the Appendix E.

Also in the second semester, with the integration of a new module of FAQs, some minor problems started to appear. By default, the platform was accepting all the suggested small-talk answers by API.ai and returning them right away, without losing time with consulting the knowledge base or the AIML files. It was a good solution because until then, all the intents, service and FAQ related, where trained in the API.ai agent. When the FAQ intents were removed, the engine started to map some inputs to undesirable small-talk domains. For example, questions such as "Do you know if Vodafone sends paper bills?" were being replied to with generic templates like "I know a lot of things".

In order to surpass this, the platform had to start filtering the small-talk domains that could be accepted. In the end, all the domains that fell in the categories of 'user', 'agent' or 'person' were ignored, and the original input would be passed to next sequence step.

Another minor occurrence happened in May, when API.ai notified that the small-talk documentation had changed to a newer version with more domains and that the previous format would be dropped until the end of the month. After a quick analysis of the situation with the scrum master, it was concluded that a day of work would be enough to perform the upgrade, and so it was.

6.1.3 Main Menu

The main menu is one of the most import features of any customer support assistant. It works as the main dashboard and interface of web and mobile applications, where the user can easily have access to the functionality of the product.

In the case of the Telco Bot, there are two main capabilities: service and FAQ questions. Later in the project came the idea of adding a troubleshooting feature to the bot that would help the users solve some technical problems in their devices. A Python-based web scrapper tool even started to be implemented in order to automatically grab the information from the website. However, a few days later, this requirement's priority was set to a minimum level, when it was decided that there would be no time for it.

The service tasks are all grouped in a carousel card, which was called "Account Information". This carousel must always appear in the first position, and it is followed by a variable number of other cards, each one for a different category.

The menu appearance can be triggered in two situations: when the user greets the chatbot (Figure 6.5) or when the user explicitly asks for help (Figure 6.6).

While in websites and mobile applications, for example, the content is almost always visible and reachable, in a conversation it is hard to do. Constantly pushing a menu of

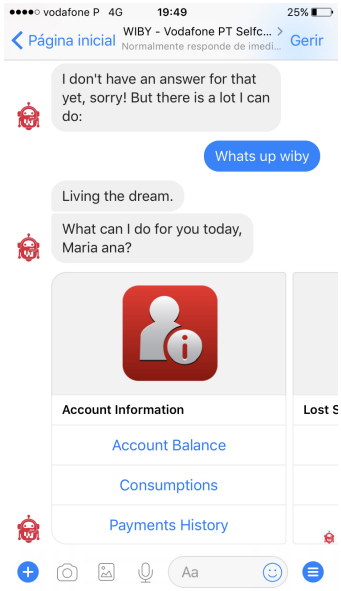

FIGURE 6.5: After user greeting

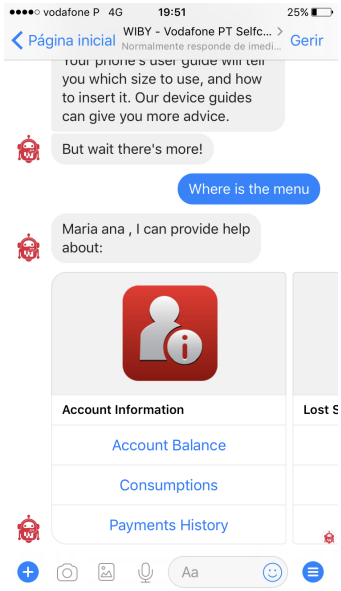

FIGURE 6.6: After help request

FIGURE 6.7: Telco Bot Main Menu

options into the conversation impairs the user experience but, on the other hand, leaving the conversation in an open context may lead to some confusion and a feeling of loss on the user side. Some requirements were defined in order to achieve a middle ground, such as the already mentioned steerback flow, the quick reply's suggestions after a FAQ is answered and, as represented in Figure 6.8, reminders of useful commands that the user can type.

Sometimes, the bot will remind the user that they can ask for help whenever they wants, as shown in Figure 6.8.

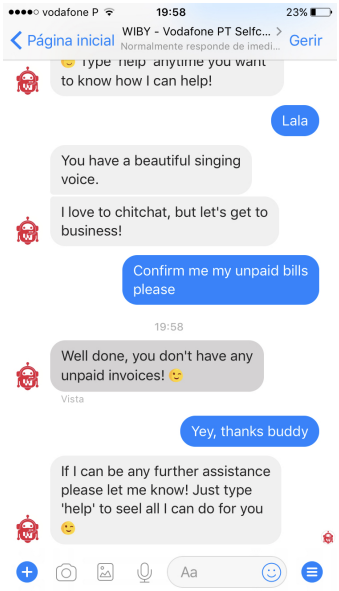

FIGURE 6.8: Telco Bot reminds the user that he can use the 'help' command

6.1.4 Notifications

In the first semester a use case that would allow the bot to push messages to the users, without receiving any input first was implemented. In the Telco Bot context, this could allow the telecom operator's to send notifications about consumptions reaching a given threshold or warnings about the next debt's date, for example.

In order to achieve that, some new modules had to be created, namely:

- **Bot Events**: each bot can be assigned to a list of events that can be triggered. Events are no more than functions that are called when the respective event emitters are triggered.
- **Notifications Endpoint**: each bot has a specific endpoint for receiving notifications events (/notification). The request format expected must be pre-accorded.

Telco Bot currently supports the event 'WarnData', that sends a warning message to a user, warning him about the limit of his data consumptions. For that to happen, an external service must send a request to the notification endpoint with the event id ('warn_consumptions'), the name of the bot and an ID of the target user. Given that we are in the domain of the operators, it makes sense that users are identified by their phone number. With that information, the Telco Bot has to find the remaining information about the user (their ID in the various messaging channels, for example)

and trigger the event. When 'WarnData' is triggered, is starts the workflow 'Warning of Data Consumptions' (refer to section E.2 of the Appendix).

After sending a warning text message, the bot will search for additional upgrade plans to suggest. If it finds nothing, it doesn't say anything else. If it does, it presents the options, as represented in Figure 6.9. The user can then choose an option, or just ignore.

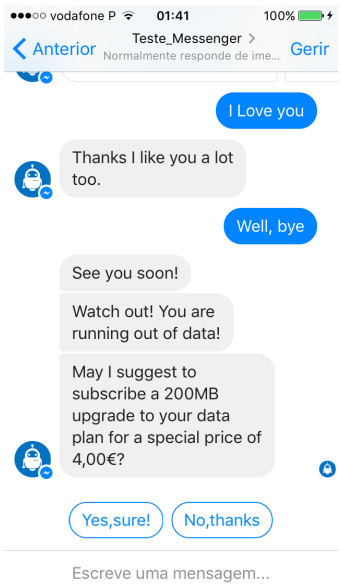

FIGURE 6.9: Warning Notification

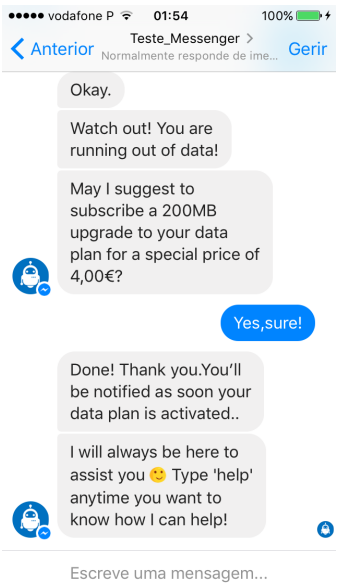

FIGURE 6.10: Warning Notification Reply

FIGURE 6.11: Telco Bot Warning Notification

Developers can add new notifications anytime. The only required steps are the creation of the BPMN flow intended for that service and the creation of the event to be triggered.

Contrary to the remainder of the bot's actions, this use case was not integrated with any operator's real data and it was not updated since its first conception. However, its importance remains the same, and the notifications' module is still available in the Bot Platform for future applications.

6.1.5 Services Support Component

There are three major service categories that the chatbot can currently supply: consumptions, account balance and payments' history. In the next sections, each one will be described with more detail.

There are, however, some similar behaviours in all of the services categories. In order to give an accurate answer, the bot must know what entity the user is interested in.

So, whenever a general request is made, the bot must ask for the additional information until it has all the necessary data.

Another improvement was done later in the project: if for a certain service category (consumptions, for example) the user has made in the past a lot more requests in one of the entities (for example, they almost always request their Data consumptions), then the bot shall present right away that information, and only after present other options. If that was the intended information, it is one less click to be done. If not, the options will still be available for choice. For an entity to be seen as a probable target, it must have been chosen more times than half of the total calls that were made to the services of its category.

6.1.5.1 Vodafone Integration

In the second semester, the possibility of connecting the Telco Bot to an operator's service and switching from dummy values to real data was discussed with the product owner. After reaching a project manager that had been working in the MyVodafone application, the intern was granted access to a set of REST endpoints that could provide information about the following topics:

- **User Profile** - information about PIN number, PUK number, e-mail, name, customer type, and other Vodafone account details.
- **Balance** - information about the consumptions of data, SMS and calls, as well as the next renewal dates, the periodicity of renewal, the next debit date and the current balance.
- **Billing** - information about the user's invoices, namely the content of their last bill and the current payments due ammount and deadline.
- **Supplementary Services** - information about extra services that the user is signed to.
- **Tariff** - information about the user's current tariff.

These endpoints allowed us to cover all of the bot's services except for the last charge information, which had to remain with a default value.

To have access to the user's information, the Telco Bot needs to have access to their credentials in order to authenticate him. As so, a new development phase led to a basic authentication feature where user's are registered with their Vodafone credentials.

Until this registration step is not successfully completed, the user will not be capable of engaging conversation with the Bot. This scenario is represented in Figure 6.12 and in Figure 6.13 is an image of the registration page, a webview that runs within the messaging app, without needing a browser.

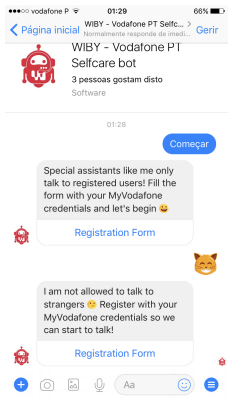

FIGURE 6.12: Registration Obligation

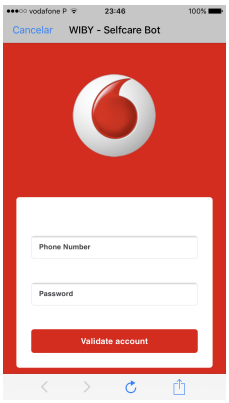

FIGURE 6.13: Registration Webview

The intern used her own Vodafone credentials during the internship duration and also had access to another enterprise account, in order to see where the main differences were and to adopt the bot replies to all the possibilities.

The current authentication method will be probably replaced in the future with a One Time Password verification, where instead of a combination of phone number and password, the authentication relies on a combination of a phone number and an SMS token.

The main challenge in this development phase was the latency of the Vodafone service's requests, especially the balance request that sometimes would take more than 10 seconds to be processed. If the bot starts to take too long to respond, the users will think that some error occurred or that the message was lost, and start typing more messages. This problem was mitigated by providing some kind of feedback indicating that the bot is processing the information, such as the visual tweek of 'user is typing' that most of the instant messaging applications provide.

The biggest problem was with Alexa because the device has a pre-defined request timeout. When the answer could not be found in time, Alexa would close the connection and end the use case. In order to solve this, the Vodafone requests' responses started to be stored in the local user storage. When information from the services was needed, if it has been asked recently it would be available locally. However, in order to keep the information updated, new requests should have to be done periodically. Since the Human Assistance dashboard implements a session system, the same logic was applied here.

When the dashboard detects that a user session has ended, it sends an event message to the Bot Platform, and therefore the local user's storage is cleaned.

6.1.5.2 Consumptions Information

This services category can provide information about the user consumptions on three entities: SMS, mobile data and calls. As mentioned, when the required information is not provided, the bot asks for more, as shown in Figure 6.14, and then replies the answer (Figure 6.15).

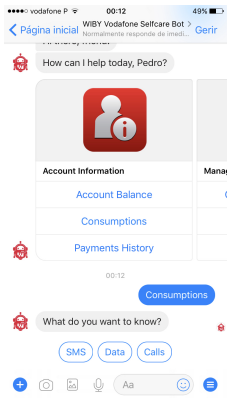

FIGURE 6.14: Consumptions Request

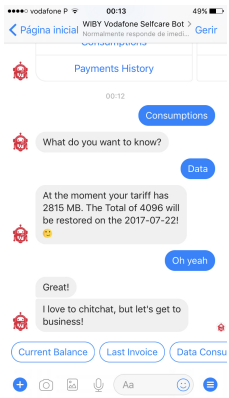

FIGURE 6.15: Consumptions Reply

The already mentioned probable target scenario is represented in Figure 6.16. The user can, also, just ask for a specific topic from the start, and get an answer (Figure 6.17).

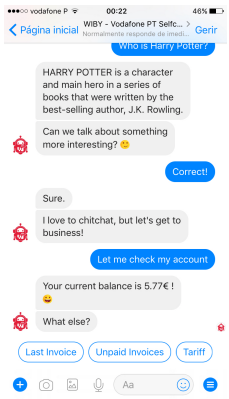

FIGURE 6.16: Most probable answer

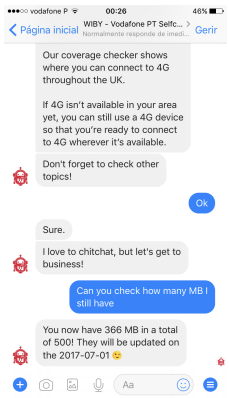

FIGURE 6.17: Direct request

6.1.5.3 Account Balance Information

There are four major service categories that the chatbot can currently supply: last invoice, unpaid invoices, user's tariff and balance. The last one was not required in the original proposal but starting in the second semester it became an essential asset.

The conversational flow is very similar to the consumptions one, without having the warning sub-part. Figure 6.18 shows some interactions.

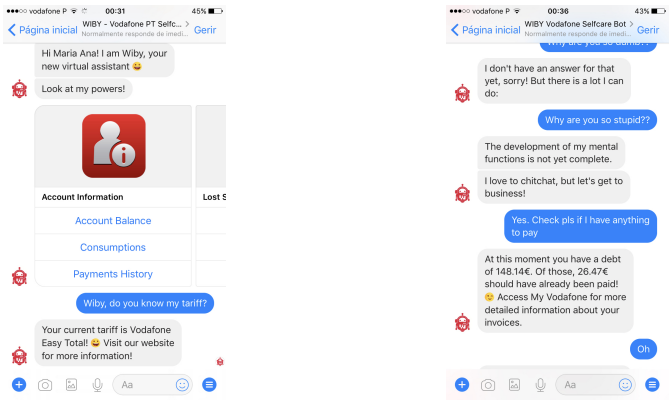

FIGURE 6.18: Telco Bot Account Balance Service

6.1.5.4 Payments History Information

There are two major service categories that the chatbot can currently supply: last debt and last charge information. The conversational flow is very similar to the account balance's one, but only with two engaging patterns, so far.
The last phone charge information is the only one that is still a dummy value because no way of getting it has been made available yet. As for the last tariff's debt request, the bot returns the date and value of the last debt and the date of the next one.

6.1.6 FAQ Support Component

The FAQ support component had to be rethought in the second semester due to integration with the FAQs knowledge base. The domain of the question-answering is much more open than the services' domain. That is why another intern has been working on a different approach for these cases, which includes developing a specific classifier that uses machine learning to calculate question-question similarity.

In the first weeks of the second semester, while the knowledge base wasn't ready for integration, another solution using the same agent in API.AI was implemented. This solution was to train different intents and associate them to certain utterances. For

example, when a user asks "How much is it?", the utterance is classified as an intent of "know the cost of service". In order to retrieve the right answer, we only need to know what service the user is referring to (i.e., the name of the service is a required entity for the intent "know the cost of service").

This involved a lengthy tedious job, not only because every question needed to be individually trained in the agent but also because each intent needed to have an entry in the database with the required entities it should be associated with (if any). With the new integration, all this work was relegated to the knowledge base, which would take care of receiving a user question and find the most probable answer for it. However, all the intents that were trained about Call+, Message+ and Backup+ remain available in the platform, and were the ones that were used in the first bot demonstrations.

The first thing to do before integrating with the knowledge base was to find a new set of questions that the bot would be able to answer. By consulting the original Vodafone FAQ page [189] and the equivalent page in spanish [190] (because Vodafone Spain would be the one to whom the bot would be presented to in the next months), a total of fifty new FAQs were documented and approved by the product owner. In order to use the new corpus as a training set for the knowledge base classifier, the intern was also asked to complete the document with three different ways of asking each of the questions. The final results can be found in Appendix F - Frequently Asked Questions.

In order to communicate with the knowledge base, a new API had to be created. The endpoints and the request and response objects were defined during a with the responsible for the maintenance of the knowledge base. A description of that API can be found in Appendix A - Architecture.

When the user aks for help, the main menu is displayed. In the FAQ section, the users are always presented the two most popular questions and a third button that allows the user to see more questions. When the third button is clicked, all the questions that the bot has access to are displayed in form of quick replies. Both cases are represented in Figure 6.19. Note that the two most popular questions are dynamically updated and are not fixed in place.

Everytime the bot answers a question that corresponds to an existing FAQ, it will always end up showing more related questions about the category, whether the request came from a click or from a text input (Figure 6.20).

When the user asks a question that has no match anywhere - services, knowledge base or AIML files - the bot presents a 'not-understand' message followed by a set of quick replies with the questions that had the highest scores in the knowledge base matching. It can happen that one of those holds the right answer, its score just wasn't high enough to

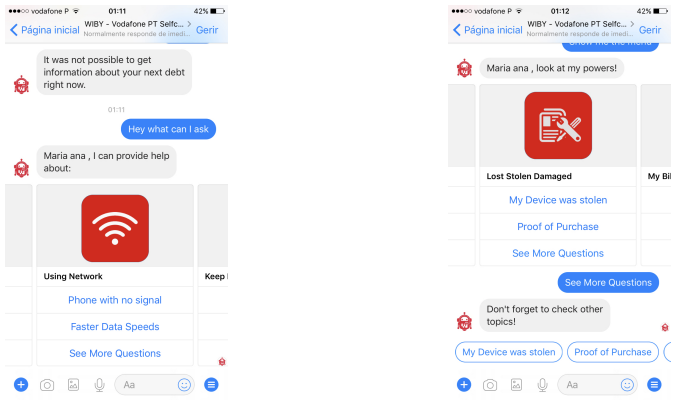

FIGURE 6.19: FAQs in the Bot Main Menu

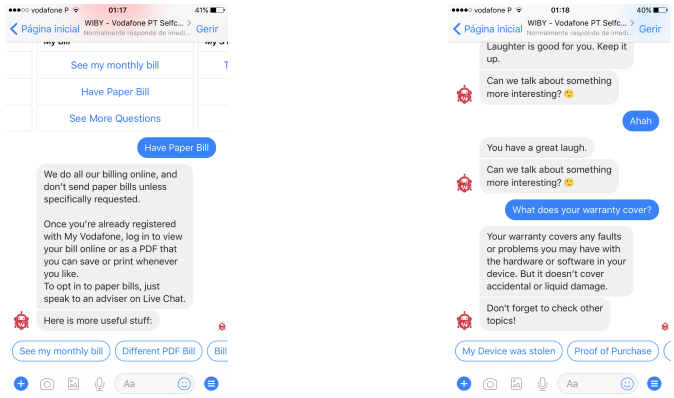

FIGURE 6.20: Telco Bot Questions Service Suggestions

reach the threshold of admittance. This can happen anytime there isn't an input match, independently of the original nature of the question, as represented in Figure 6.21.

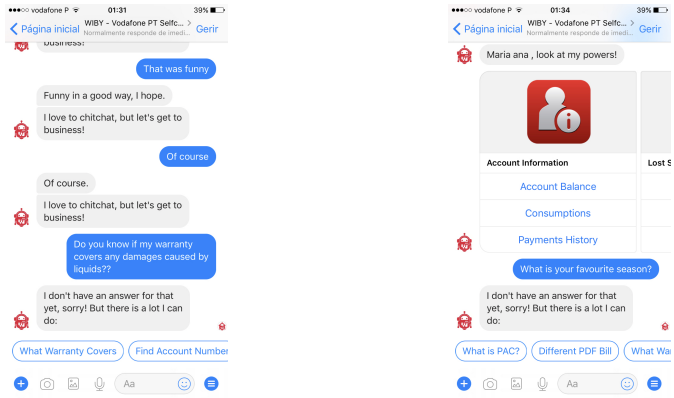

FIGURE 6.21: Telco Bot Questions Service Flow

It was noticed, almost from the start, that some answers were not suited for a messaging context. In a website, it is expected to find long answers to certain questions, especially when they apply to different user contexts. In addition, several utterances were accompanied by helpful media such as links, images and videos, which could not be directly transported to a messaging channel.

To deal with the media problem, the bot was enriched with extra elements to support those features. Additionally, in the dashboard part, an extra component was added where the operators can add image links and URLs to each question. Figure 6.22 represents an example of two enriched answers. The images always follow the question text and the URLs are presented as a button.

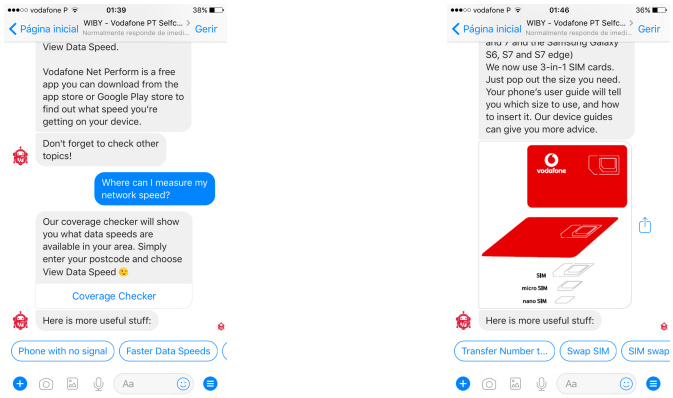

FIGURE 6.22: Enriched FAQ's Answers

So far, there is no support for video media, and it is not a priority at the moment. In a near future, the capability of adding more than one URL for an answer should be added, since currently it is only possible to choose one and certain questions would benefit with more.

Regarding the context problem, a new requirement appeared in the third quarter of the second semester, stating that it should be possible to add constraints to a FAQ and provide different answers to each of the constraints. This lead to the restructuring of the API endpoints of the knowledge base and of the BPMN flow of the FAQS. In addiction, a new attribute had to be created for each user, in order to keep the information of their context. This object would be passed to the knowledge base each time a new answer was requested so that if the question needed a context, it could be retrieved instantly. Figure 6.23 shows an example of an answer whose format is not appropriate for a messaging bot. Figure 6.24 shows how the bot now handles FAQs that need to be contextualized before being answered.

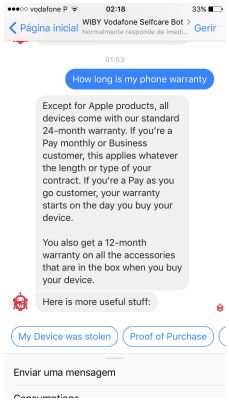

FIGURE 6.23: Inappropriate FAQ answer for the Telco Bot

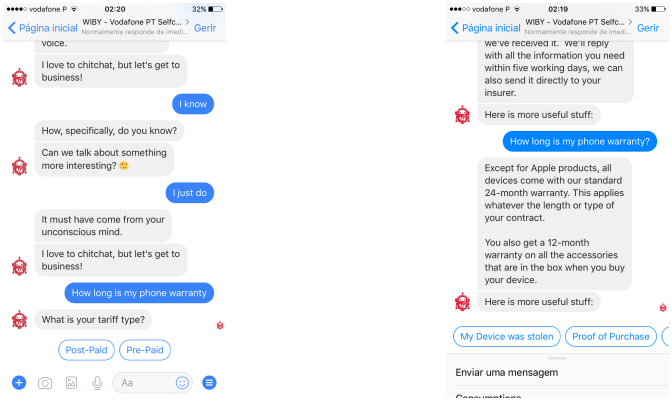

FIGURE 6.24: Telco Bot answers contextualized questions

There are constraints that can be easily fetched from the user's personal account, such as the tariff example showed above. Currently, all the constraints have to be manually created in the Dashboard and assigned to each question in particular. For future work, it is expected that constraints such as the user's tariff are retrieved right away from the operator's account, with no need to be asking for them.

6.2 Channels

Although the internship proposal only addressed the intention of deploying the Telco Bot in one popular messaging app, the work was successful enough to be deployed in Facebook Messenger and in RCS+, an app developed at WIT Software which is part of a new messaging standard designed to greatly improve the messaging and calling functionality that comes installed on phones by default.

The requirements on both channels are exactly the same, except for the persistent menu, which the RCS app doesn't have yet. Figure 6.25 shows some screens of the Telco Bot in RCS+.

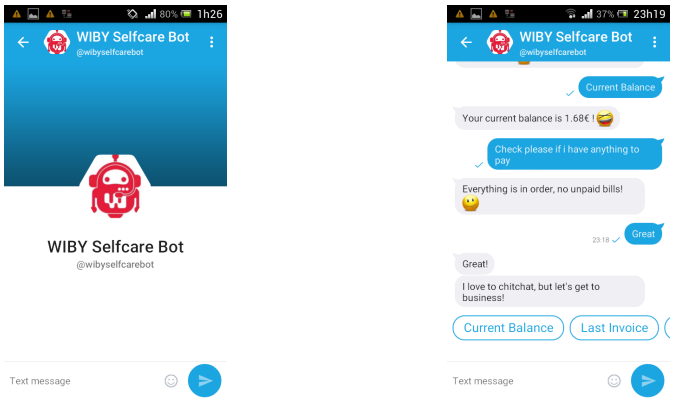

FIGURE 6.25: Telco Bot on RCS+

During the second semester, the company acquired an Amazon Echo device, the smart speaker developed by Amazon that connects to intelligent personal assistant, Alexa. Because it is possible to build personalized skills and make them available through the agent, some days were spent trying to implement some of the bot services in the Alexa. Currently, only the service part of the bot is available and only via direct requests. For example, Alexa can answer questions like "Alexa, ask MyVodafone what is my current balance" or "Alexa, ask MyVodafone if I have bills to pay". In order to communicate with Alexa, a "Skill' had to be implemented in the platform, through the use of a REST interface.

6.3 Human Assistance Dashboard

The Human Assistance dashboard is part of a more general dashboard from where human operators can manage the Bot Platform and their bot's conversations. The dashboard design and core implementation did not happen in the context of the internship; it was implemented by another team starting in the first semester, and some weeks later the project was handed to the intern. The Human Assistance interface is represented in Figure 6.26.

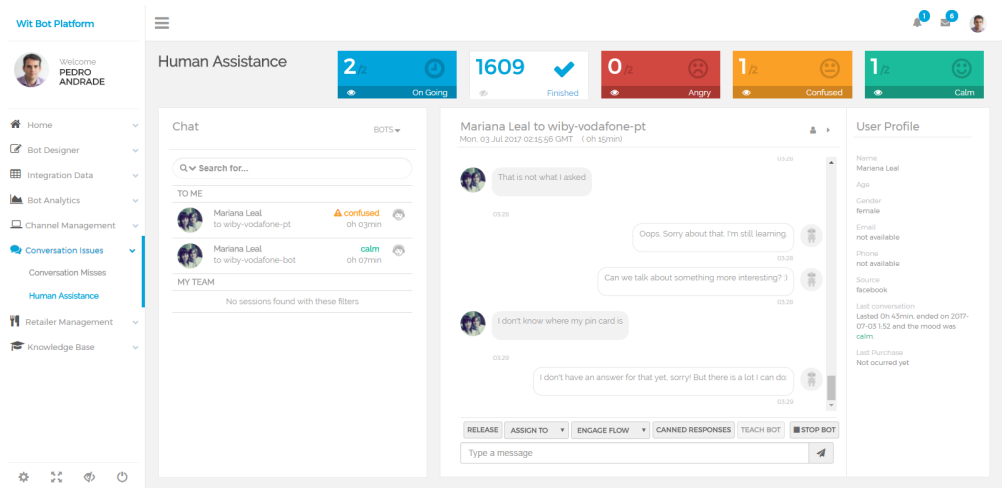

FIGURE 6.26: Human Assistance Dashboard

It was the intern's responsibility to implement the requirements related with the knowledge base and the bot platform interactions as well as the integration between the dashboard and the sentiment analysis server.
The sentiment server is constantly classifying the messages of the conversations and labelling them as positive, neutral or negative, which are then translated into "Angry" or "Calm" values that appear in the dashboard. The sentiment analysis is explained with more detail in the Section 6.3.1.

Each human assistant should be able to perform four main actions on a conversation. One is the possibility of sending a text message to the user from the dashboard, in the name of the bot. The second is a variant of the first one, where instead of typing a whole new sentence, the operator can use a pre-defined example from the available canned responses set.

Another action that the operator should be able to perform is to engage one of the flows of the bot, in order to answer a user's request that the bot is not understanding. Everytime a new conversation is registered, the dashboard requests the bot's available flows and the respective engaging patterns, so that they can be triggered. The flows are always available in a combo box below the chat window, as represented in Figure 6.27.

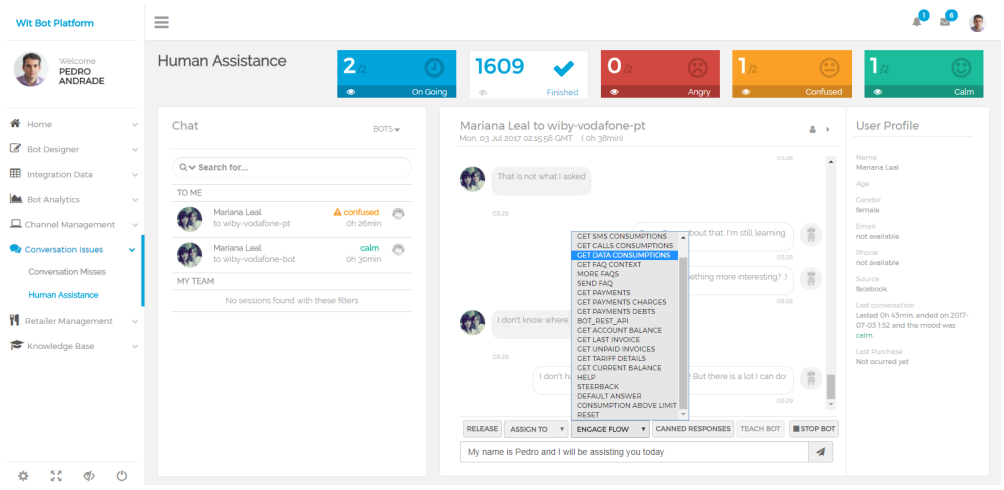

FIGURE 6.27: Engage Flow and Send Message from the Dashboard

The Telco Bot implements a special flow called '*Get Context FAQ*' that allows the operator to ask about the user's context that is necessary to answer a FAQ. When this flow is chosen, a window opens, showing all the FAQs in the knowledge base for which the user's constraints are not met yet, as represented in Figure 6.28.

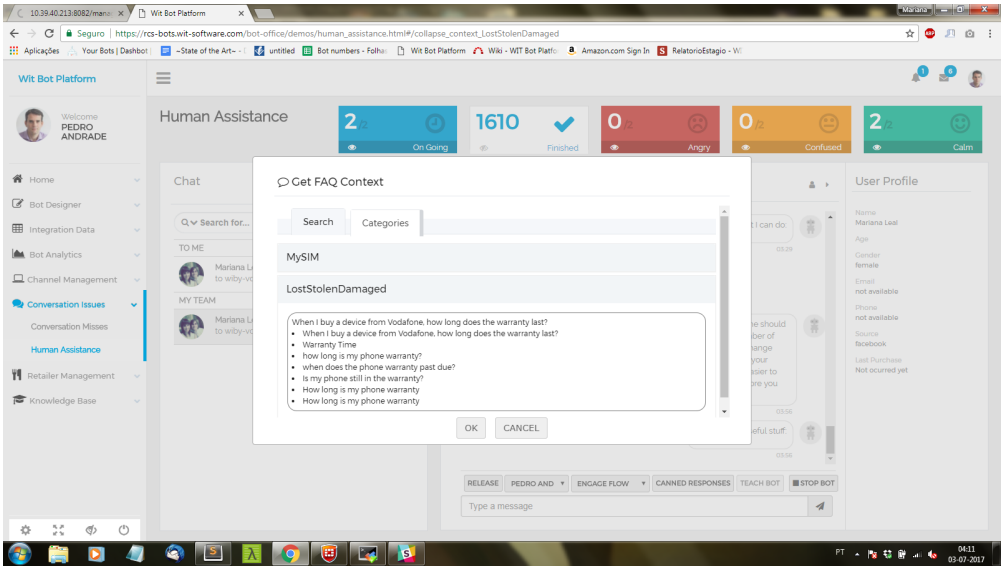

FIGURE 6.28: Engage *'Get Context FAQ* Flow

Whenever the Bot Platform detects that the context of a user was modified, it sends a socket event to the dashboard, which updates the local user's context copy that is kept in order to select what FAQs to present in the window. For more information about the available events, refer to Appendix A.

The last requirement, the teach bot, is what allows the bot to get smarter. When the bot is failing to answer a user's question, the operator is able to select the input that is

not being recognized and associate it to a answer of knowledge base, as represented in Figure 6.29. This data is sent to the knowledge base endpoint and, the next time the user asks the same question (or a very similar one), the bot will provide the right answer.

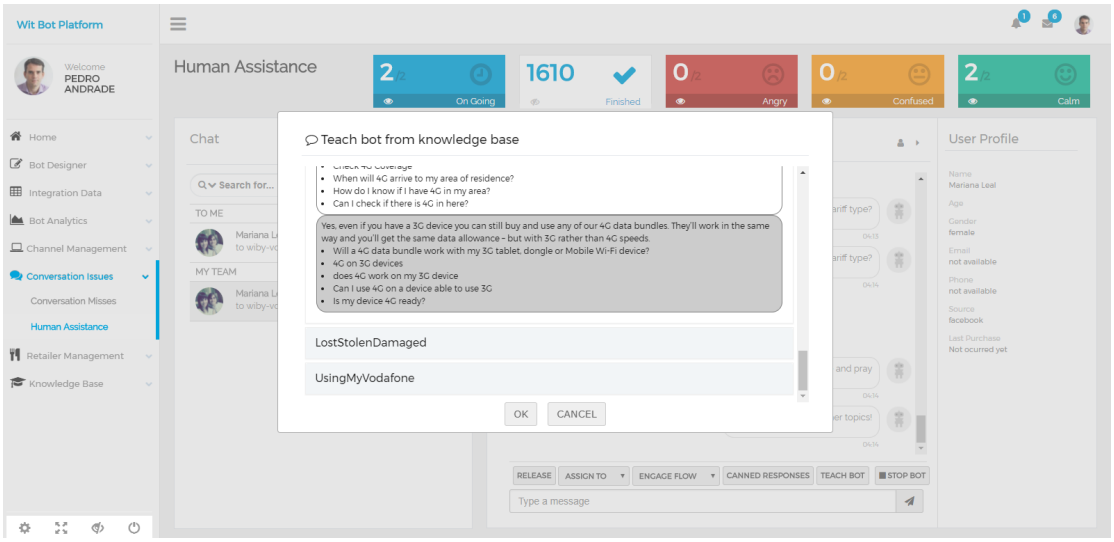

FIGURE 6.29: Teach Bot From the Knowledge Base

Other minor features include the release and the assign button, which allows the moving of a conversation from our list to the general team list and the assignment of a conversation to a special team member. However, since there has never been multiple members accessing the platform, these were never tested.

Lastly, the dashboard should allow the management of some parameters and data of the knowledge base. As so, they were added to the main menu the following tabs, under the main section 'Knowledge Base':

- **Manage knowledge sets** - interface for adding and removing the FAQs datasets supported by the bot. It is also possible to turn on and turn off the availability of a category in a given moment.
- **Learning performance** - interface that allows the monitoring of the detailed results of the last training process.
- **Query Knowledge Base** - interface for interactively querying small inputs to the classifier.
- **Settings** - interface for setting some of the knowledge base parameters such as minimum score and minimum confidence threshold.
- **Manage Constraints** - interface for creating constraints.

None of mentioned interfaces were developed from scratch, since they have already been implemented by other team members. They were embedded in the dashboard and later in the project one of the interfaces received some modifications in order to attend the necessity of adding enriched elements to some FAQs. Two forms, such as the one represented in Figure 6.30 were implemented in the individual dataset's configuration page in order to allow the creation of images and URLs in each answer.

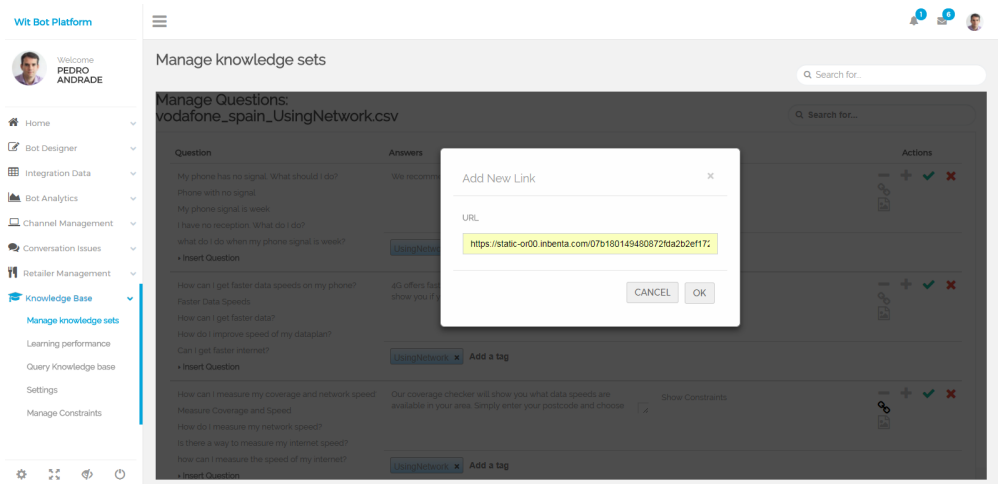

FIGURE 6.30: Add Media Elements to FAQS

One feature that ended up not being implemented was the 'Stop Button' action, that should prevent the bot from trying to answer to more user inputs while the human operator is handling the situation. The low priority of the task and the lack of time during the development phase lead to the impossibility of the implementation in the current version, although it is listed in the future work.

6.3.1 Sentiment Analysis

In Section 2.1.4.2.2 we introduced the need for a sentiment analysis module in the Human Assistance Dashboard in order to track the sentiment levels of each conversation. After some research, a set of six tools that could fit the purpose were selected. Those tools are also described in detail in the same section. A quick summary of each one is provided next:

- **TwitterHybridClassifier** - a Python module with a model that was trained with Twitter datasets, based on sentiment lexicons, emoji detection and machine learning. It was trained with three labels: positive, neutral and negative.

- **tweetment** - a Python module with a model that was trained with Twitter datasets, based on sentiment lexicons and machine learning. It was trained with three labels: positive, neutral and negative.
- **aueb.twitter.sentiment** - a Python module with a model that was trained with Twitter datasets, based on a weighted ensemble of two systems that combined sentiment lexicons, machine learning and word embeddings. It was trained with three labels: positive, neutral and negative.
- **Vader Sentiment** - a Python module that consists of a lexicon and rule-based system specifically attuned to sentiments expressed in social media, including detection of emoticons, acronyms and slang. It returns a compound score between -1 and 1 that can be corresponded to a positive, neutral or negative label.
- **IBM Natural Language Understanding** - an IBM service that provides an API for sentiment analysis requests (free up to 1000 items per day). It can analyze a sentiment towards specific target phrases and the sentiment of documents as a whole. It returns a score between -1 and 1 and one of three labels: negative, neutral or positive.
- **Stanford Core NLP** - a Java-based web server with a model that uses *Stanford Sentiment Treebank*, a corpus of fully labeled parse trees of movie reviews, and machine learning. It returns one of five labels: very positive, positive, neutral, negative or very negative.
- **Microsoft Text Analytics** - a Microsoft Cognitive Service that provides an API for sentiment analysis requests (free up to 5000 items per month). The input features to the classifier include POS tags and word embedding and it returns a score between 0 and 1.

In order to choose what classifier to integrate with the dashboard for the prototype, a mere theoretical comparison would not be enough. Some kind of empirical evaluation about what performance to expect, and how accurate each of these approaches could be was required. As such, an experiment was devised, using real data, labeled specifically for sentiment analysis experiments, and all seven classifiers were ran on it. To the best of our knowledge, this is the first experiment of its kind, with these state of the art classifiers and these datasets. The next sections describe the datasets that were used for comparison, the measures that were used for evaluation and the results and final discussion.

6.3.1.1 Testing Datasets

As mentioned in Section 2.1.4.2, there is still a lack of available benchmark tools for sentiment analysis. Besides, from the available datasets, a lot of them are labeled only as positive or negative (a binary classification), without corpus for neutral or objective sentences. It was important to evaluate the classifiers' performance in datasets with objective and subjective sentences because that is what we expect to deal in the real world scenario - most of the time, customers will only want to ask simple questions to the bot without expressing any sentiment or frustration.

SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems that started in 1998. It began with tasks of word-sense disambiguation and, since 2013, sentiment analysis of Tweets has become a recurrent competition of the series. The organizers acknowledged [191] that working with informal text genres, like tweets and SMS, brings new challenges for natural language processing, with the introduction of creative spelling and punctuation, misspellings, slang, new words, abbreviations and emojis. Realizing the lack of social media/informal text corpus available at the time, the SemEval team created a new corpus, containing Tweets and SMS that were collected and manually labeled between 2012 and 2013. In 2014 [192], the team also added a new LiveJournal corpus, with users's comments from the weblog LiveJournal [193]. This corpus is currently used by SemEval in order to understand how systems trained on Twitter corpus (other training sets provided by SemEval to the sentiment analysis competitions) perform on other sources.

Given the similarity between the context of those corpus and the scope where the sentiment analysis module should be applied, as well as the fact that they include the neutral label, they were chosen as testing datasets for the experiment. However, some transformations were applied to the datasets in order to make them more similar to an instant messaging app message. For example, hashtags and Twitter-like mentions (with the @ symbol) are elements that do not exist in normal messaging interactions. As so, hashtags were removed from the Twitter corpus and mentions were replaced with a default name. We also removed the entries that correspond to "retweet" instances, another Twitter-only element.

When the Stanford Core NLP tool was being studied during the state of the art research, it was found that, in order to train their neural network classifier, the team ended up building a new sentiment corpus [194] based on a previous binary set of movie reviews from the Rotten Tomatoes website [195]. This kind of data differs from the scope of the project, mainly because movie reviews include a lot of ironic words and some features specific to comments (for example, in the same sentence the user can praise the movie

plot and also bash on the main actor's performance) [196]. It is nevertheless interesting to evaluate the performance of the chosen classifiers in different contexts and see how volatile it is. However, the final decision should be made upon the first corpus, which carries the kind of data we are more interested in.

The datasets chosen are represented in Table 6.1, along with their label distribution.

Corpus	Total Instances	Positive Labels	Neutral Labels	Negative Labels
SMS	2091	491	1210	390
Live Journal	1142	426	412	304
Tweets	3037	1456	1066	515
Movie Reviews	8525	3565	1747	3213
	14795	5938	4435	4422

TABLE 6.1: Statistics of the datasets chosen for evaluation of the sentiment analysis tools

Table 6.2 presents some examples of instances from each corpus of the three different labels. All the corpus are composed of short texts, with Tweets and SMS having more slang, abbreviations, misspellings and punctuation elements, including smiles and emojis. Live Journal comments and Movies' reviews comprise a more precise writing, but while the former is more direct and informal, the second is trickier, combining multiple sentiments expressed in one sentence, with more complex syntactic structures.

Corpus	Positive	Negative	Neutral
SMS	"I hope just now didnt give u a huge shock. Very happy dat u give me a chance... Have an early nite.. :-)"	"Ya damn and that stupid woman needs to get fired...lol"	"So wat time n whr u wan meet me?"
Live Journal	"good to hear that you are doing so well and are happy now !"	"Some posts seem to serve no purpose but to make people pissed"	"I saw a vegetarian version the other day , in a can."
Tweets	"Excited for all of the Youth Soccer Teams today!!!!,Go get a "kick" out of this Saturday!"	"if ur anywhere near kansas state dont come watch us play in the preseason NIT on nov. 12th im gonna fight u for that too"	"Rafa will play his third round match at the BNP Paribas Open at Indian Wells on Tuesday against Marcel Granollers."
Movie Reviews	"Not all of the stories work and the ones that do are thin and scattered , but the film works well enough to make it worth watching."	"Everything's serious , poetic, earnest and – sadly – dull"	"Matches neorealism's impact by showing the humanity of a war-torn land filled with people who just want to live their lives"

TABLE 6.2: Examples of classified instances of each dataset

For future work, it would be important to gather some new datasets, preferably context-dependent ones. By the time the internship took place, it was not possible to have access to any operator's data about previous customer care interactions. That is why it is important to save future conversations in order to build a new telco corpus, that can posteriorly be hand-labeled by humans.

6.3.1.2 Experiment and Results

After applying the transformations already described to the testing datasets, they were normalized to the same form: each instance is a tab separated line with the sentence's unique identifier, the correct label and the sentence.
A Python script was implemented for each of the classifiers, according to how the classification requests should be made (calling a Python module or performing a HTTP request). The predictions were saved in files following the same format defined for the input datasets.

For the classifiers that did not follow the (negative,neutral,positive) labeling system, some adjustments were made. For example, the Stanford Core NLP has two additional labels - very positive and very negative - which were translated to their closest value (positive and negative, respectively) when being saved. The Microsoft tool only returns a score between 0 (very negative) and 1 (very positive). As such, the interval was sliced in three: from 0 to 0.35 it would correspond to a negative label, from 0.36 to 0.64 to a neutral label and from 0.65 to 1 to a positive label. These transformations standardize the output space onto our desired output space without any significant loss of information.

Aside from the predictions, the time of execution of the script was also recorded, which corresponded to the total of time needed to classify the whole dataset. From that value, the average time of classification of one instance could be determined. Other measures that were obtained were the model's accuracy and the F-measure, which includes determining the precision and the recall.

Classification accuracy is very popular in machine learning, and corresponds to the number of correct predictions made divided by the total number of predictions made. However, it should not be the only measure to address when evaluating a classifier's performance. For example, if we were testing a dataset with 90 positive sentences and 10 negative sentences, and we were using a classifier that only returns 'positive', we would have an "incredible" 90% of accuracy. In order to avoid these situations when dealing with unbalanced data, such as our case, other measures have to be evaluated and interpreted.

For each dataset and classifier, a confusion matrix was obtained. In Figure 6.31 we present a representation of a confusion matrix with the most important attributes identified.

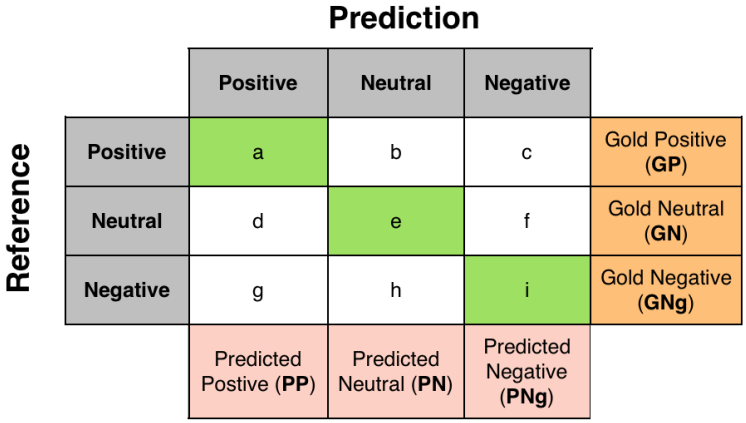

FIGURE 6.31: Confusion matrix for sentiment analysis

Precision is the ratio of items that were predicted as having the label X and that actually have that label. For example, using the matrix represented in Figure 6.31, the precision of classification of positive labels would be

$$Precision('positive') = \frac{True Positives('positive')}{Total Predictions('positive')} = \frac{a}{PP}$$

On the other hand, **recall** is the ratio of items with the label X in the gold standard that were in fact classified with that label. Following the same example, the recall of the positive class would be

$$Recall('positive') = \frac{True Positives('positive')}{Gold('positive')} = \frac{a}{GP}$$

If we would look at these measures separately, we would encounter similar problems as the already mentioned issue with accuracy. A classifier which marks everything as positive would achieve a precision of 0.5 and a recall of 1, and vice-versa. These extreme examples indicate that the two statistics may be conflicting and maximizing just one is not enough. Therefore, a third measure is used, the F-measure, which is the harmonic mean of precision and recall. F_1 is a private case of weighted F-measure where precision and recall are given the same weight and it's obtained with the formula:

$$F_1('positive') = \frac{2(Recall('positive') \times Precision('positive')}{Recall('positive') + Precision('positive')}$$

The final F-score of the classifier in the dataset is the average of the F-scores of the three labels. This is the measure that we should look at when studying the performance of the classifiers. In the specific case of this project, the F-measure of the negative cases was also very important. In the eyes of the intern and of the product owner, the Human Assistant Dashboard should be very sensitive to negative variations of the conversations. Calling the human assistant when there is actually no need for him is a much more desirable scenario than having conversations where the frustration is high and there is no signaling for intervention.

The important results are presented next. For other auxiliary elements such as confusion matrices, precision and recall values, refer to Appendix D. For each measure comparison graph, only the highest values were highlighted, for a more clear understanding. Appendix D details all the other values.

In Figure 6.32, we present the accuracies of the classifiers when applied to the four datasets.

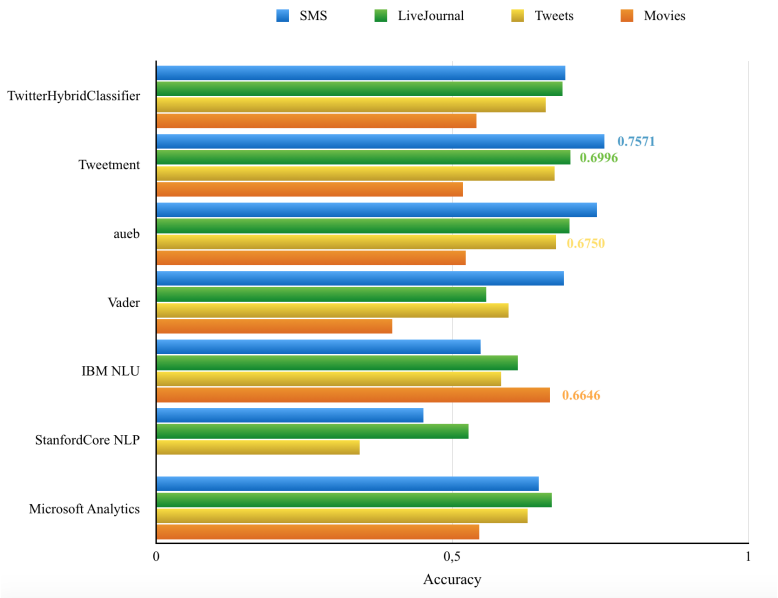

FIGURE 6.32: Accuracy results of the sentiment tools

As already mentioned, the accuracy alone should not be seen as a trustworthy evaluation measure. However, it is possible to descry some patterns. The results were overall good for the three important datasets, with most of the them surpassing 60%. The three classifiers trained with similar corpus achieved the best results, but the proprietary tools followed them close. Most of these patterns remain the same when looking at the average F-score results in Figure 6.33.

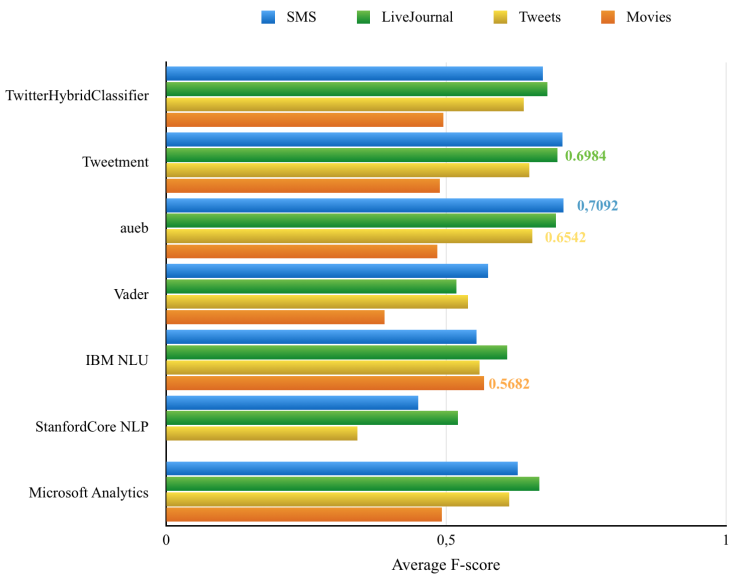

FIGURE 6.33: Average F-score results of the sentiment tools

The first three tools maintain the best results, as well as Microsoft Text Analytics. Vader Sentiment was the tool where the difference was higher, specially in the SMS and Twitter corpus, which are the more unbalanced sets. IBM NLU and Stanford CoreNLP were the tools with the worst performance in the informal datasets. However, the former had both the best accuracy and F-score with the additional movies dataset (CoreNLP was not evaluated in the movies dataset since it is trained upon part of that data). This is a good indicator that while these two classifiers may not be the best fit for a social media sentiment tracker, they are expected to work better in contexts with other type of data, for example, on sentiment analysis of e-mails or reviews.

In Figure 6.34, we present the results of the F-score concerning only the negative label. The TwitterHybridClassifier, which is the top in the other measures, performs best with the SMS corpus and, perhaps surprisingly, with the Movies corpus.

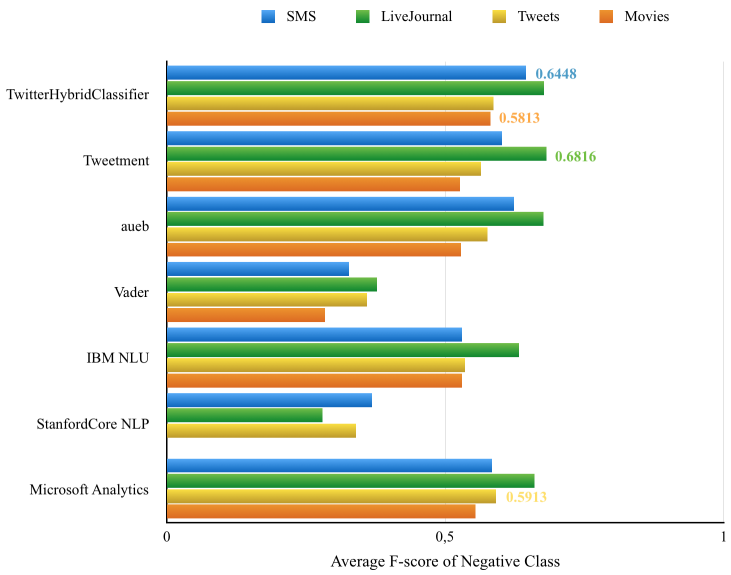

FIGURE 6.34: F-score results of the sentiment tools relative to the negative class

Lastly, in Figure 6.35, we present the average amount of seconds that each classifier takes to perform a classification. The Vader's results are not represented due to its small values, in the order of 10^{-4}. Right after it, TwitterHybridClassifier is the one who performs faster, on average. Once again, the Microsoft's tool proved to be an efficient classifier, achieving relatively small values, considering that it is a HTTP-request based system. This metric is not of significant relevance for the final decision, hence why there is not much focus on other statistical properties such as standard deviation. It is meant as a general "feel" of the performance of the different classifiers per instance, which is all within acceptable bounds (below 0.3 seconds).

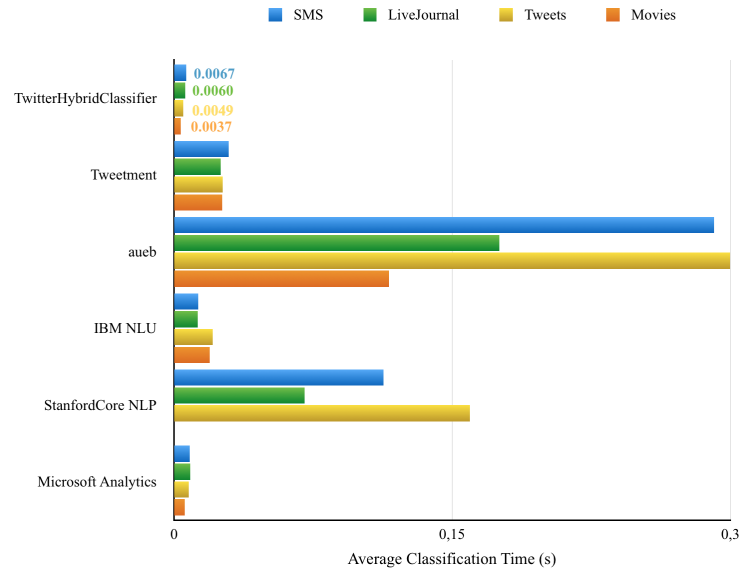

FIGURE 6.35: "Average classification time of an instance" results of the sentiment tools

Overall, the experiment produced interesting and positive results. Four of the seven classifiers achieved accuracies and F-scores above 60% in the three informal datasets and with relatively fast execution times for the majority of them. The results of the F-scores of the negative label were not disappointing either, which is another very positive outcome, given the already mentioned importance of identifying the maximum number of negative cases in the conversations.

Because of that, TwitterHybridClassifier was chosen as the classifier to integrate with the platform's prototype. A meeting with the product owner was scheduled after the results of the experiment were gathered, where he was left doing some ad-hoc tests on the top three classifiers - TwitterHybridClassifier, tweetment and aueb - and where the preference over TwitterHybridClassifier was reinforced. He indicated, however, that the second best classifier, tweetment, should also be available for interactive testing.

6.3.1.3 Implementation

Before integrating the Human Assistance Dashboard with the new sentiment tools, they needed to be available via Internet. As such, a Web Server was implemented and a REST API was created. For more details about the two elements, refer to Appendix A. A third additional feature was also implemented, which includes a simple web interface for interactive testing of the classifiers available, as represented in Figure 6.36.

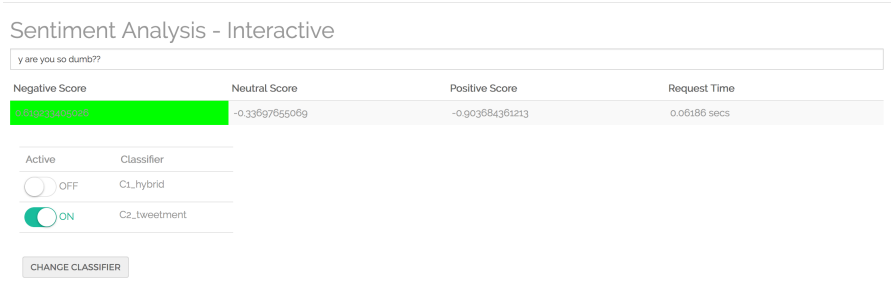

FIGURE 6.36: Web interface for testing the platform's sentiment analysis classifiers

The implemented sentiment analysis technique uses a sliding window approach. Every time there is a new text message, a sentiment analysis on that utterance is requested. The analysis returns an object with the three labels - positive, negative and neutral - and the value of strength of each sentiment. These strength values are used to compute the current average of each sentiment level by using the results of the last N messages received by the bot. N is a parameter that can be set in the configurations and that defines the size of the window of past messages that should be used to calculate the current value of each sentiment label.

As such, in pratice the algorithm keeps track of three different averages computed within the same sliding window: the average of "negative", "positive" and "neutral" sentiment levels. In each iteration, the algorithm looks at the highest value of these to determine the current sentiment level of the user. In order to avoid wrong decisions when there isn't sufficient strength associated with each sentiment, a threshold was empirically determined. Only labels which surpass this threshold are selected, otherwise the user is considered to be in a "neutral" level. This threshold had to be empirically determined due to the nature of the strength values – each strength value represents the distance of the provided sample to the separating hyperplane. There is no current standardized way of transforming this distance into a probabilistic model, although there have been some attempts in literature [197]. One can compare strength values among themselves to determine which is stronger than the other, but one cannot assert "how" strong an individual sample is on its own, because it is not mapped to a probabilistic model (i.e. between 0 and 1). The threshold was, then, empirically determined to be 0.2, after dozens of messages had been analyzed during the test phases of the product. This value is sufficient to avoid too many false positives.

The sliding window approach was chosen because although the sentiment analysis tools work on isolated sentences, our users are engaged in a conversation. It would therefore be a bad decision to ignore this fact and simply pick the sentimental analysis value of the last sentence. The sliding window approach is a simple but effective way of generalizing sentiment analysis of single sentences/tweets to more general conversations.

To further improve this approach, a "reset" mechanism was implemented, which clears the sliding window after a pre-determined amount of time or when the user changes the context of their conversation. In practice, this aims to align "conversation boundaries" with the sliding window boundaries – the sliding window should not contain results from a conversation that took place an hour ago or on a completely different topic.

Positive and neutral labels are transformed into "Calm", and "Negative" is transformed into "Angry". When the conversation status shifts to an angry state, a warning sign next to the conversation is added, notifying the operator that there is trouble in that conversation. The warning sign should disappear when the conversation state is reset to "Calm".

External actions can also take part in the sentiment label selection. It was agreed with the product owner that each time the bot could not find a match for an input, it should immediately send an event to the dashboard. In its turn, the dashboard would set the conversation status to "Confused" and add a warning sign next to it. This status can only be reverted when the operator engages a flow or teaches the bot a new utterance, meaning that the problem was handled.

Chapter 7

Verification and Validation

One of the several phases associated with the process of software engineering is the quality assurance phase. This stage can happen during or after the development phase and is of high importance since it is where the validation and verification of all the constituent parts of the developed system take part.
In an agile development process, ideally both verification and validation activities occur as close to simultaneously as possible. That is what happened during the internship, since the user stories were constantly being updated and refined.

The software's **verification** is the process of evaluating the work outcome of a development phase to determine whether they meet the the requirements and design specifications imposed for that phase. It will help to determine whether the software is of high quality, whether the system is well-engineered and error-free, but will not ensure that the system itself is useful. The process of verification happened during the meetings with the product owner and the rest of the development team, where the intern had to describe how the code was designed and written. Other times, when one of the team members reported a problem in the code, a quick meeting was scheduled between him and the intern, in order to inspect the source of the problem.

While the verification process tries to answer the question 'Are we building the product right?', **validation** deals with the question 'Are we building the right product?' [198]. In a business sense, it is the process of evaluating the product to check whether the software meets the stakeholders' expectations and requirements. This is where the testing phase takes place as a dynamic mechanism of validating, to determine whether the product satisfied the specified business requirements.
During the internship, there was no actual client with real demands. The intention was to create iterative prototypes with a value proposition good enough to attract potential stakeholders. In order to do that, during the second semester, the product was used in

some public and private external demonstrations conducted by the product owner. He was the one responsible for controlling the project's backlog and adapting it according to the feedback received in those demonstrations. Some of these include the Mobile World Congress 2017 [199], the CommunicAsia 2017 [200] and meetings with Vodafone Spain.

7.1 Functional Testing

Functional testing is a type of black-box testing, meaning that the test suite is based on the project specifications and with no knowledge of the implemented code. Its levels can range from unit testing, where little pieces of code such as one function is tested, to system testing, where each requirement is tested as a whole.

At first, during the development phases, the functional testing was performed in an undocumented way after the implementation of certain modules. When the demonstrations of the system started to happen, some requirements had to be fully validated before each one of them occur.

Later in the project, a set of automatic tests cases began to be developed. With more and more tests scenarios appearing, at some point it will not be reliable to perform each test manually, thus the necessity of some automation tool to help the process. The tool is a combination of two components: Mocha [201], a Node.js test framework that allows for asynchronous testing and flexible reporting, and Chai [202], a behaviour driven development assertion library. This last library provides assert and expect statements that allow us to compare the output of a test case to the output that was desired for a given input. For example, in order to check if the right user was being returned from the database, we can write:

```
db.saveAttribute(USER.username,"personalInfo.phone","0000",function(err,res){
  expect({ ans : res }).to.have.deep.property('ans.personalInfo.phone','0000')
  done()
})
```

The tool was later integrated with a third library, Mochawesome, in order to provide a user interface that facilitates the developer or the product owner to find what tests the software did not pass and why. Figure 7.1 shows part of that interface.

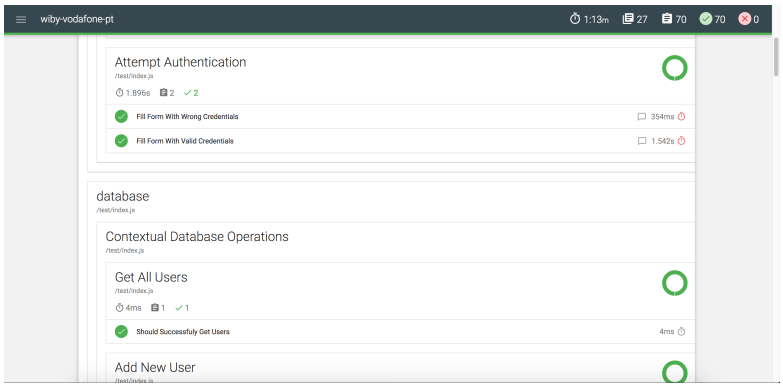

FIGURE 7.1: Report generated from the automatic testing process

Every test case is clickable, allowing the tester to see what was the test input, the expected output and the actual result. Figure 7.2 shows an example.

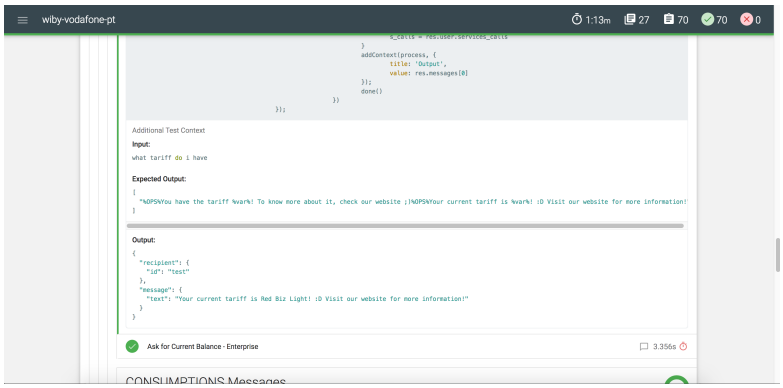

FIGURE 7.2: Example of the input, expected output and actual output of a test case in the test report

The development of this tool started in June, as so, there was no time to implement a large set of test cases. That is not problem in terms of work planning since it was not a scheduled task, it is an extra attribute, and did not interfere with the manual execution of the remaining tests. It ended with test cases ranging from simple database operations tests to whole requirements related with the services aspect of the bot. Additionally, it is ready to deal not only with text responses but also with messaging graphic elements, namely the carousel and the quick replies.

The remainder of this section describes the test cases that were performed. For each one we assigned a target, a designation, an execution mode and the final result. The fact that the overall results are positive does not mean that the software is not error prone. Exhaustive testing is not practical not even for the simplest programs, even less for a software that deals with natural language inputs that can be very lengthy and different. The final test suite includes the mainstream examples of input that can occur and that were used regularly during the demonstrations.

In Table 7.1 we present the basic unit tests that were performed to the database methods. These simple assertions were implemented as an introduction of the intern to the testing tools, before stepping on to the "next level". Although basic, they focus on a very critical part of the Bot Platform that is constantly used during conversations' handlings.

Target	Description	Execution	Result
Bot Platform	Get All Users	Automatic	✓
Bot Platform	Add New User from Facebook	Automatic	✓
Bot Platform	Get User By Channel ID	Automatic	✓
Bot Platform	Get User By Username	Automatic	✓
Bot Platform	Update User Personal Information	Automatic	✓
Bot Platform	Delete User	Automatic	✓
Bot Platform	User is not returned after deletion	Automatic	✓
Bot Platform	User information cannot be updated after deletion	Automatic	✓

TABLE 7.1: Unit Testing on database operations

In Table 7.2 we present the first Telco Bot related test cases. They already include authentication requirements, basic text interactions and enriched responses, for example, when the bot is greeted, it should return a greeting message and a carousel with the main menu (for more information refer to the requirements' Appendix C).

Target	Description	Execution	Result
Telco Bot	User cannot chat while he is not registered	Automatic	✓
Telco Bot	User Logs in with success	Automatic	✓
Telco Bot	User tries to login with invalid credentials	Automatic	✓
Telco Bot	User asks "what is melodrama?"	Automatic	✓
Telco Bot	User asks "what is your name?"	Automatic	✓
Telco Bot	User asks "How's the weather?"	Automatic	✓
Telco Bot	User thanks the bot ('thanks','txs','thank you','thanks :)','ty')	Automatic	✓
Telco Bot	User greets the bot ('hi','hey','whats up','hello','good morning")	Automatic	✓
Telco Bot	User asks for the menu ('help','menu','what can i ask','wheres the menu')	Automatic	✓
Telco Bot	User says goodbye ('bye bye','see ya', 'bye','see you soon','goodbye')	Manual	✓
Telco Bot	User asks for a joke ('joke','tell me a joke')	Manual	✓

TABLE 7.2: Chitchat Interactions testing

There are additional tests that were performed not shown in Table 7.2. These were not added because although they are implemented, their quality was not the same as that of the tests presented. As part of future work, they can become part of the "official" testing suite.

In Table 7.3 we present the test cases executed to check if the service component of the bot is working as expected. The tests described were executed multiple times with the following variations:

- **Different User Account** - the same tests were ran using two different user accounts: one was a pre-paid and the other was an enterprise. This way more scenarios can be covered because the pre-paid user did not have invoices information, for example, while the enterprise did.
- **Different User's Services Calls** - the same tests for were also ran with different configurations of the user's services calls values. As stated in Appendix C, when one service has been asked much more than the other services of the same category, they should appear right away. For example, when the user selects the 'Account Balance' button, the Telco bot, instead of returning all the possible quick replies, returns the last bill information followed by the remaining possible quick replies.

The results were all positive for each one of the combinations mentioned.

Note that in addition to the automated tests, the bot and bot platform were extensively tested "on-site" during multiple demos performed at different events and different contexts. The system almost always remained operational, serving as a "field test" of the bot's capabilities, stability and quality of code. As previously mentioned, the importance of automated tests cannot be understated, but given the broad scope of the internship and the constant testing which the system faced, the team is pleased with the stability and reliability of the proposed and implemented solution.

Target	Description	Execution	Result
Telco Bot	User selects 'Account Balance' from main menu	Automatic	✓
Telco Bot	User selects 'Consumptions' from main menu	Automatic	✓
Telco Bot	User selects 'Payments History' from main menu	Automatic	✓
Telco Bot	User asks for his account balance ('hows my account balance','my account')	Automatic	✓
Telco Bot	User asks for his consumptions ('my consumptions','where are the recent consumptions')	Automatic	✓
Telco Bot	User asks for his payments logs ('how about my payments','do you have my payments history')	Automatic	✓
Telco Bot	User selects 'Balance' from quick replies	Automatic	✓
Telco Bot	User asks for his balance ('hows my balance','how much money i have')	Automatic	✓
Telco Bot	User selects 'Last Invoice' from quick replies	Automatic	✓
Telco Bot	User asks for his last invoice ('do you have my last bill','where the last invoice')	Automatic	✓
Telco Bot	User selects 'Unpaid Invoices' from quick replies	Automatic	✓
Telco Bot	User asks if he has unpaid invoices ('do i have anything to pay','do i have unpaid bills?')	Automatic	✓
Telco Bot	User selects 'Tariff' from quick replies	Automatic	✓
Telco Bot	User asks what is his tariff ('whats my tariff','what tariff do i have rn')	Automatic	✓
Telco Bot	User selects 'Data' from quick replies	Automatic	✓
Telco Bot	User asks for his data consumptions ('hows my dataplan','how many mb i have left')	Automatic	✓
Telco Bot	User selects 'SMS' from quick replies	Automatic	✓
Telco Bot	User asks for his SMS consumptions ('how many sms i have left','do i have sms?')	Automatic	✓
Telco Bot	User selects 'Calls' from quick replies	Automatic	✓
Telco Bot	Users asks for his calls consumptions ('how many minutes do i have left','my calls')	Automatic	✓
Telco Bot	User selects 'Debts' from quick replies	Automatic	✓
Telco Bot	User asks about his last debt ('when was the last debt','tariffs last debt')	Automatic	✓
Telco Bot	User selects 'Charges' from quick replies	Automatic	✓
Telco Bot	User asks about his last charge ('when was my last charge','phone charges')	Automatic	✓
Telco Bot	User asks question while conversation is on the middle of a different flow	Automatic	✓

Table 7.3: Bot service's requirements functional test cases

Chapter 8

Conclusion and Future Work

This internship focused on the design and implementation of a series of bot technologies to be integrated with several platforms, so as to help strengthen WIT's vision as an innovator in the mobile and telco market. The prototypes developed in the context of this internship involved several different aspects, such as Natural Language Processing, evaluation of existing and trending bot platform solutions, integration with Facebook Messenger, RCS services and the Alexa platform, integration with BPMN tools, existing and rapidly evolving bot platforms, as well as existing operator/Vodafone services, among others. It was a rich experience which allowed me to grow several skills related to teamwork, cooperation, rapid application development, rapid prototyping, testing, requirements elicitation, deployment scenarios, and others. In addition to the bot, a set of modules and technologies regarding sentiment analysis had to be developed and deployed.

Although the concept of a bot platform might seem simple at first sight, this is certainly not the case. The bot platform presented in this internship possesses several interdependent components, and the broad scope of this work required working on nearly all of these components: I was required to make performance experiments regarding sentiment analysis tools; state of the art evaluations of current bot platforms; improvements of the current bot platform to support, for example, additional BPMN functionality; extensions to the current telco bot frontend to support the plethora of new functionality introduced with the NLP components of the internship; development of new modules for integration with other platforms such as the Alexa; development of new web services and algorithms for sentiment analysis; integration with and management of Facebook pages for the bots; rapid prototyping to allow important decisions at the end of each sprint. With the help

of WIT's IT support team, several parallel deployments of the bot platform were maintained simultaneously, all the while these were subjected to demonstrations and new feature ideas.

In the end, the experience of working within this team has no doubt been rewarding and lead to a prototype of which we can all be proud of, and which has consistently been well-received and acclaimed in its many demonstrations. The undoubtedly stressful times where the platform required ongoing adaptation to new features, while remaining sufficiently backwards compatible to be available for demonstrations resulted in a resilient and tested system.

The WIT Bot Platform is now a sophisticated platform which allows operators to monitor, intervene in and evaluate any number of concurrent conversations taking place between bots and users in any of the company's existing platforms, such as Facebook Messenger and RCS. The operators can not only manually intervene, but they are notified about possibly angry or dissatisfied users using state-of-the-art sentiment analysis tools.

As part of this work, in addition to the several components developed in several languages and frameworks such as Python, Node.js, Express, Angular, AIML and others, we introduced two main core additions to the state of the art in NLP and sentiment analysis. The first is a considerable analysis of existing bot/NLP development tools with regards to several features. This analysis, if published, would allow new adopters of bot technology to quickly gauge which tools are best suited for their work, and this undoubtedly fills a gap in the state of the art in bot tools. The second addition to the state of the art regards an empirical evaluation of sentiment analysis tools, ranking them according to specific criteria. These two experiments were a core part of the state of the art of this work and provided invaluable help in the decision about which tools to use and why. Our experiments were the first of its kind to consider the seven popular tools and APIs used for sentiment analysis on the specified datasets, ranking them according to specific criteria. Once again, were this analysis to be published, it would aid newcomers pick the best tool for their sentiment analysis needs. These two experiments were a core part of the state of the art of this work and provided invaluable help in the decision about which tools to use and why.

As such, I can consider this to have been a successful internship, where all the required components were developed, with rapid adaptations to the new flow of incoming requirements. The technology was picked and developed using sound principles and carefully documented experimentation, it was tested with automated tests and constant demos for validation, and it resulted in what we consider to be a maintainable code base which interacts with many different components.

Lastly, and although we consider this to be a successful work, there is always additional work which can be done. As such, for future work, the human assistance dashboard shall have a stop bot action, auto-suggestions for human responses and an auto-triage of conversations to humans. As for the telco bot, the future work includes the possibility of asking for a real human, a troubleshooting module and an evaluation score after each interaction. In the machine learning field, a lot is yet to be done. WIT is now investing in a new development team that will, in time, build a set of machine learning projects which may produce new classifiers for sentiment analysis or NLP based on the currently implemented algorithms in the prototype. These future algorithms might, for instance, have better performance or have a specific target audience, such as telco customers, for which they provide better replies.

Appendices

Appendix B

Risks

In this appendix are comprised the periodic results of the risk evaluation throughout the project. The first evaluation occured in November, 2016, and it was redone in January, 2017. For a detailed description of each risk, refer to Section 3.3.

B.0.1 Second Evaluation - January, 2017

In the beginning of 2017, in the mid term of the internship, the risks were re-evaluated and the risks matrix redesigned. There were no new risks identified. The resulting matrix is represented in Figure B.1.

As expected, some risks were mitigated, mostly the ones concerning the expertise of the intern on the technologies required for the project.

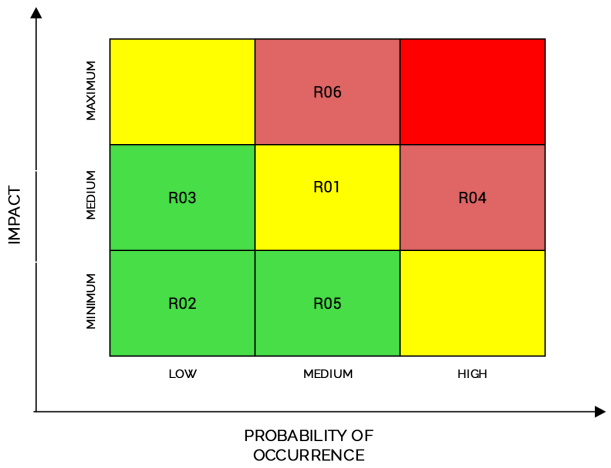

FIGURE B.1: Risks Matrix - Second Evaluation (January,2017)

Although an additional set of risk reevaluation moments were planned, the ever-changing nature of the project, coupled with the stability and reliability of the service, as attested by several live demonstrations of the platform, meant that no additional formal risk evaluations were performed. This does not imply that risk assessments were not performed in ad-hoc fashion throughout the internship. Indeed, during the SCRUM sprints, the status of the project and its potential risks were discussed. It was recognized that after the initial analysis had been performed, there were no risks with high probability that should be particularly focused on. As such, as aforementioned, there are no further formal risk evaluations to be presented.

Appendix D

Sentiment Tools' Results

This appendix contains the detailed results obtained while evaluating the performance of the sentiment analysis tools. In Section 6.3.1 we presented the more important measures: accuracy, time of execution, average F-score and F-score of the negative label. In this appendix we also include the F-scores for the other labels, positive and neutral, as well as the confusion matrices. The discussion of the important results is made in Section 6.3.1 while this appendix only holds the auxiliary metrics that contributed to find those same results.

D.1 Accuracy, F-scores and Time of Classification

The first tables include the results for the main evaluation metrics - accuracy, time and F-scores. In Table D.1 are the results for the SMS corpus, in Table D.2 the results for the LiveJournal corpus, in Table D.3 the results for the Twitter corpus and in Table D.4 the results for the Movies corpus.

	Twitter Hybrid Classifier	tweetment	aueb.twitter.sentiment	Vader Sentiment	IBM NLU	Stanford CoreNLP	Microsoft Text Analytics
Accuracy	0.6911	0.7571	0.7476	0.6887	0.5481	0.4510	0.6461
F-score (all labels)	0.6731	0.7078	0.7092	0.5749	0.5541	0.4500	0.6276
F-score (negative label)	0.6448	0.6022	62.39	0.3279	0.5299	0.3691	0.5843
F-score (neutral label)	0.7447	0.8255	0.8075	0.6225	0.6031	0.5112	0.6965
F-score(positive label)	0.6298	0.6958	0.6964	0.7743	0.5295	0.4698	0.6031
Average Time of Classification (s)	0.0067	0.0296	0.2913	0.0002	0.0132	0.1129	0.0086

TABLE D.1: Evaluation metrics' results of the sentiment tools in the SMS corpus

	Twitter Hybrid Classifier	tweetment	aueb.twitter. sentiment	Vader Sentiment	IBM NLU	Stanford CoreNLP	Microsoft Text Analytics
Accuracy	0.6821	0.6996	0.6979	0.5569	0.6103	0.5271	0.6681
F-score (all labels)	0.6809	0.6984	0.6958	0.5180	0.6091	0.5212	0.6667
F-score (negative label)	0.6775	0.6816	0.6769	0.3776	0.6325	0.5645	0.6602
F-score (neutral label)	0.6605	0.6909	0.7015	0.6076	0.4993	0.4230	0.6343
F-score(positive label)	0.7048	0.7229	0.7090	0.5688	0.6958	0.5762	0.7055
Average Time of Classification (s)	0.0060	0.0253	0.1754	0.0002	0.0128	0.0702	0.0088

TABLE D.2: Evaluation metrics' results of the sentiment tools in the LiveJournal corpus

	Twitter Hybrid Classifier	tweetment	aueb.twitter. sentiment	Vader Sentiment	IBM NLU	Stanford CoreNLP	Microsoft Text Analytics
Accuracy	0.6576	0.6730	0.6750	0.5953	0.5825	0.3434	0.6273
F-score (all labels)	0.6382	0.6488	0.6542	0.5389	0.5593	0.3420	0.6124
F-score (negative label)	0.5864	0.5640	0.5762	0.3598	0.5354	0.3404	0.5913
F-score (neutral label)	0.6097	0.5640	0.6750	0.6038	0.4459	0.3046	0.5537
F-score(positive label)	0.7187	0.7166	0.7116	0.6533	0.6966	0.3810	0.6924
Average Time of Classification (s)	0.0049	0.0263	0.5534	0.0003	0,0209	0.1595	0.6124

TABLE D.3: Evaluation metrics' results of the sentiment tools in the Twitter corpus

	Twitter Hybrid Classifier	tweetment	aueb.twitter. sentiment	Vader Sentiment	IBM NLU	Microsoft Text Analytics
Accuracy	0.5408	0.5181	0.5230	0.3987	0.6646	0.5458
F-score (all labels)	0.4954	0.4890	0.4842	0.3901	0.5682	0.4927
F-score (negative label)	0.5813	0.5263	0.5281	0.2848	0.5426	0.5546
F-score (neutral label)	0.2943	0.3105	0.3084	0.3532	0.5748	0.2781
F-score(positive label)	0.6108	0.6302	0.6285	0.5325	0.5763	0.6454
Average Time of Classification (s)	0.0037	0.0260	0.1159	0.0003	0,0193	0.0059

TABLE D.4: Evaluation metrics' results of the sentiment tools in the Movies corpus

References

[1] S. Curtis, "Instant messaging overtakes texting in the UK." http://www.telegraph.co.uk/technology/news/10568395/Instant-messaging-overtakes-texting-in-the-UK.html, 2014. [Online; accessed July 2017].

[2] B. Intelligence, "Messaging apps are now bigger than social networks." http://www.businessinsider.com/the-messaging-app-report-2015-11, 2016. [Online; accessed July 2017].

[3] S. Kojouharov, "Ultimate Guide to Leveraging NLP and Machine Learning for your Chatbot." https://goo.gl/5SLNHC, 2016. [Online; accessed July 2017].

[4] "All languages have underlying structural rules that make meaningful communication possible.." https://www.boundless.com/psychology/textbooks/boundless-psychology-textbook/language-10/introduction-to-language-60/the-structure-of-language-234-12769/. [Online; accessed July 2017].

[5] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[6] A. Collomb, C. Costea, D. Joyeux, O. Hasan, and L. Brunie, "A study and comparison of sentiment analysis methods for reputation evaluation," *Rapport de recherche RR-LIRIS-2014-002*, 2014.

[7] D. Maynard and A. Funk, "Automatic detection of political opinions in tweets," in *Extended Semantic Web Conference*, pp. 88–99, Springer, 2011.

[8] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[9] A. Statnikov, D. Hardin, I. Guyon, and C. F. Aliferis, "A Gentle Introduction to Support Vector Machines in Biomedicine." `https://med.nyu.edu/chibi/sites/default/files/chibi/Final.pdf`, 2009.

[10] W. De Mulder, S. Bethard, and M.-F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Computer Speech & Language*, vol. 30, no. 1, pp. 61–98, 2015.

[11] J. Zheng, "How To Build a Recipe Slack Bot Using Watson Conversation and Spoonacular API." `https://goo.gl/oQzKGp`, 2016. [Online; accessed July 2017].

[12] "A better way of building software." `https://www.scrum.org/Resources/What-is-Scrum`. [Online; accessed July 2017].

[13] "WIT Software, S.A.." `http://www.wit-software.com/`. [Online; accessed July 2017].

[14] G. K. Agyapong, "The effect of service quality on customer satisfaction in the utility industry-a case of vodafone (ghana)," *International Journal of Business and management*, vol. 6, no. 5, p. 203, 2011.

[15] "Consumer study finds overwhelming dissatisfaction with ivr." `http://www.interactions.com/press-releases/consumer-study-finds-overwhelming-dissatisfaction-with-ivr/`, 2012. [Online; accessed July 2017].

[16] Jacada, "How to Visualize your Customer Service and IVR Experience." `https://www.jacada.com/images/WhitePapers/pdfs/White_Paper_How_to_Visualize_your_customer_service_IVR_experience.pdf`, 2014. [Online; accessed July 2017].

[17] A. Mittal, A. Agrawal, A. Chouksey, R. Shriwas, and S. Agrawal, "A comparative study of chatbots and humans," *Situations*, vol. 2, p. 2.

[18] Statista, "Number of apps available in leading app stores as of June 2016." `https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`, 2016. [Online; accessed July 2017].

[19] comScore, "U.s. mobile app report," tech. rep., 2014.

[20] "Gartner inc.." `http://www.gartner.com/technology/home.jsp`. [Online; accessed July 2017].

[21] "Top strategic predictions for 2017 and beyond," tech. rep., 2016.

[22] J. Verhage, "The Bot Economy Is Growing Even Faster Than the App Economy Did." `https://www.bloomberg.com/news/articles/2016-09-15/the-bot-economy-is-growing-even-faster-than-the-app-economy-did`, 2016. [Online; accessed July 2017].

[23] "Vodafone Portugal – Comunicações Pessoais, S.A." `http://www.vodafone.pt`. [Online; accessed July 2017].

[24] H. Bidgoli, *The Internet encyclopedia*, vol. 1. John Wiley & Sons, 2004.

[25] D. Greene and D. O'Mahony, "Instant messaging & presence management in mobile adhoc networks," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pp. 55–59, IEEE, 2004.

[26] A. Wold, "Multi-threaded conversations in instant messaging clients," 2007.

[27] A. Garcia and J. B. Jacobs, "The interactional organization of computer mediated communication in the college classroom," *Qualitative Sociology*, vol. 21, no. 3, pp. 299–317, 1998.

[28] M. Petronzio, "A Brief History of Instant Messaging." `http://mashable.com/2012/10/25/instant-messaging-history/#TKW9p3WgOkqt`, 2012. [Online; accessed July 2017].

[29] J. Tyson and A. Cooper, "How instant messaging works," *How Stuff Works*, 2005.

[30] "WhatsApp." `https://www.whatsapp.com/`. [Online; accessed July 2017].

[31] J. Chavin, A. Ginwala, and M. Spear, "The future of mobile messaging: Over-the-top competitors threaten sms," tech. rep., 2012.

[32] "Chat app messaging overtakes SMS texts, Informa says." `http://www.bbc.com/news/business-22334338`, 2013. [Online; accessed July 2017].

[33] D. T. Tohmatsu, "Short messaging services versus instant messaging: value versus volume." `https://goo.gl/Y6XQzt`, 2014. [Online; accessed July 2017].

[34] Statista, "Most popular mobile messaging apps worldwide as of April 2016, based on number of monthly active users (in millions)." `https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/`, 2016. [Online; accessed July 2017].

[35] SimilarWeb, "The Most Popular Messaging App in Every Country." `https://www.similarweb.com/blog/worldwide-messaging-apps`, 2016. [Online; accessed July 2017].

[36] "Facebook Messenger." `https://www.messenger.com/`. [Online; accessed July 2017].

[37] "Viber." `https://www.viber.com`. [Online; accessed July 2017].

[38] "Line." `https://line.me`. [Online; accessed July 2017].

[39] "Telegram, a new era of messaging." `https://telegram.org/`. [Online; accessed July 2017].

[40] C. Stephanidis, *User interfaces for all: concepts, methods, and tools*. CRC Press, 2000.

[41] E. S. Raymond and R. W. Landley, "The art of unix usability," *Retrieved September*, vol. 17, p. 2008, 2004.

[42] J. Brown and M. Davis, "The evolution and future of the graphical user interface in personal computing," 2003.

[43] A. Van Dam, "Post-wimp user interfaces," *Communications of the ACM*, vol. 40, no. 2, pp. 63–67, 1997.

[44] S. Oviatt, "Human-centered design meets cognitive load theory: designing interfaces that help people think," in *Proceedings of the 14th ACM international conference on Multimedia*, pp. 871–880, ACM, 2006.

[45] M. Turk and G. Robertson, "Perceptual user interfaces (introduction)," *Communications of the ACM*, vol. 43, no. 3, pp. 32–34, 2000.

[46] J. O'Shea, Z. Bandar, and K. Crockett, "Systems engineering and conversational agents," in *Intelligence-Based Systems Engineering*, pp. 201–232, Springer, 2011.

[47] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may i help you?," *Speech communication*, vol. 23, no. 1, pp. 113–127, 1997.

[48] J. Chai, J. Lin, W. Zadrozny, Y. Ye, M. Stys-Budzikowska, V. Horvath, N. Kambhatla, and C. Wolf, "The role of a natural language conversational interface in online sales: A case study," *International Journal of Speech Technology*, vol. 4, no. 3-4, pp. 285–295, 2001.

[49] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface*. Springer, 2016.

[50] "Apple Siri." `http://www.apple.com/ios/siri/`. [Online; accessed July 2017].

[51] "Google Assistant." `https://assistant.google.com/`. [Online; accessed July 2017].

[52] "Microsoft Cortana." https://www.microsoft.com/en/mobile/experiences/cortana/. [Online; accessed July 2017].

[53] N. Statt, "Hands-on with Facebook M: the virtual assistant with a (real) human touch." http://www.theverge.com/2015/10/26/9605526/facebook-m-hands-on-personal-assistant-ai, 2015. [Online; accessed July 2017].

[54] "Slackbot: personal assistant and helpful bot." https://get.slack.help/hc/en-us/articles/202026038-Slackbot-personal-assistant-and-helpful-bot-. [Online; accessed July 2017].

[55] P. Robles, "Five pioneering examples of how brands are using chatbots." https://econsultancy.com/blog/68046-five-pioneering-examples-of-how-brands-are-using-chatbots/. [Online; accessed July 2017].

[56] T. Stolfa, "The Future of Conversational UI Belongs to Hybrid Interfaces." https://goo.gl/w6o5Vh, 2016. [Online; accessed July 2017].

[57] D. Marcus, "Messenger Platform gets an update." https://www.facebook.com/notes/david-marcus/messenger-platform-gets-an-update/10155014173359148, 2016. [Online; accessed July 2017].

[58] M. L. Mauldin, "Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition," in *AAAI*, vol. 94, pp. 16–21, 1994.

[59] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[60] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[61] M. J. Pereira, L. Coheur, P. Fialho, and R. Ribeiro, "Chatbots' greetings to human-computer communication," *arXiv preprint arXiv:1609.06479*, 2016.

[62] R. Epstein, "The quest for the thinking computer," *AI magazine*, vol. 13, no. 2, p. 81, 1992.

[63] "jabberwacky.com." http://www.jabberwacky.com/. [Online; accessed July 2017].

[64] "Cleverbot." http://www.cleverbot.com/. [Online; accessed July 2017].

[65] R. Wallace, "The elements of aiml style," *Alice AI Foundation*, 2003.

[66] "ChatScript." `https://github.com/bwilcox-1234/ChatScript`. [Online; accessed July 2017].

[67] "Mitsuku Chatbot." `http://www.mitsuku.com/`. [Online; accessed July 2017].

[68] J. L. Hutchens, "How to pass the turing test by cheating," *School of Electrical, Electronic and Computer Engineering research report TR97-05. Perth: University of Western Australia*, 1996.

[69] H. J. Levesque, "On our best behaviour," *Artificial Intelligence*, vol. 212, pp. 27–35, 2014.

[70] L. Bradeško and D. Mladenić, "A survey of chatbot systems through a loebner prize competition," 2013.

[71] E. Reiter and R. Dale, "Building applied natural language generation systems," *Natural Language Engineering*, vol. 3, no. 01, pp. 57–87, 1997.

[72] M. Yao, "Conversational Skills To Teach Your Chatbot." `http://www.topbots.com/bot-design-conversational-ui-ux-skills-teach-your-chatbot/`, 2016. [Online; accessed July 2017].

[73] M. Jia, "9 secrets to creating a world-class customer service bot." `http://venturebeat.com/2016/12/05/9-secrets-to-creating-a-world-class-customer-service-bot/`, 2016. [Online; accessed July 2017].

[74] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.

[75] S. C. Kleene, "Representation of events in nerve nets and finite automata," tech. rep., DTIC Document, 1951.

[76] B. W. Kernighan and R. Pike, *The Unix programming environment*, vol. 270. Prentice-Hall Englewood Cliffs, NJ, 1984.

[77] J. Hutchins, "The first public demonstration of machine translation: the georgetown-ibm system, 7th january 1954," *noviembre de*, 2005.

[78] E. Kumar, *Natural language processing*. IK International Pvt Ltd, 2011.

[79] N. Chomsky, *Syntactic structures*. Walter de Gruyter, 2002.

[80] E. Bach, "An introduction to transformational grammars," 1965.

[81] R. Huybregts, "The weak inadequacy of context-free phrase structure grammars," *Van periferie naar kern*, pp. 81–99, 1984.

[82] C. Culy, "The complexity of the vocabulary of bambara," *Linguistics and Philosophy*, vol. 8, no. 3, pp. 345–351, 1985.

[83] W. A. Woods and R. Kaplan, "Lunar rocks in natural english: Explorations in natural language question answering," *Linguistic structures processing*, vol. 5, pp. 521–569, 1977.

[84] "SHRDLU." `http://hci.stanford.edu/winograd/shrdlu/`. [Online; accessed July 2017].

[85] E. D. Liddy, "Natural language processing," 2001.

[86] K. S. Jones, "Natural language processing: a historical review," in *Current issues in computational linguistics: in honour of Don Walker*, pp. 3–16, Springer, 1994.

[87] D. Jurafsky and J. H. Martin, *Speech and language processing*, vol. 3. Pearson, 2014.

[88] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[89] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[90] D. Fischer, "Natural language understanding - state of the art and some current research," 2003.

[91] "WordNet: A lexical database for English." `https://wordnet.princeton.edu/`. [Online; accessed July 2017].

[92] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[93] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[94] F. C. Pereira and D. H. Warren, "Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks," *Artificial intelligence*, vol. 13, no. 3, pp. 231–278, 1980.

[95] C. J. Fillmore, *Toward a modern theory of case*. Department of Health, Education and Welfare, 1968.

[96] D. S. Wile and J. C. Ramming, "Guest editorial: Introduction to the special section 'domain-specific languages (dsls)'," *IEEE Transactions on Software Engineering*, vol. 25, no. 3, p. 289, 1999.

[97] S. Das and M. Chen, "Yahoo! for amazon: Extracting market sentiment from stock message boards," in *Proceedings of the Asia Pacific finance association annual conference (APFA)*, vol. 35, p. 43, Bangkok, Thailand, 2001.

[98] S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima, "Mining product reputations on the web," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 341–349, ACM, 2002.

[99] M. Tsytsarau and T. Palpanas, "Survey on mining subjective data on the web," *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 478–514, 2012.

[100] M. L. Richins and T. Root-Shaffer, "The role of evolvement and opinion leadership in consumer word-of-mouth: An implicit model made explicit," *NA-Advances in Consumer Research Volume 15*, 1988.

[101] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "Phoaks: A system for sharing recommendations," *Communications of the ACM*, vol. 40, no. 3, pp. 59–62, 1997.

[102] X. Jin, Y. Li, T. Mah, and J. Tong, "Sensitive webpage classification for content advertising," in *Proceedings of the 1st international workshop on Data mining and audience intelligence for advertising*, pp. 28–33, ACM, 2007.

[103] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining.," in *LREc*, vol. 10, 2010.

[104] F. Å. Nielsen, "Afinn." `http://www2.imm.dtu.dk/pubdb/p.php?6010`, 2011.

[105] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining.," in *LREC*, vol. 10, pp. 2200–2204, 2010.

[106] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[107] R. Mihalcea, C. Banea, and J. M. Wiebe, "Learning multilingual subjective language via cross-lingual projections," 2007.

[108] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, p. 271, Association for Computational Linguistics, 2004.

[109] L. Cabral and A. Hortacsu, "The dynamics of seller reputation: Evidence from ebay," *The Journal of Industrial Economics*, vol. 58, no. 1, pp. 54–78, 2010.

[110] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[111] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.

[112] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

[113] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 2, pp. 841–848, 2002.

[114] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[115] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[116] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *Machine learning: ECML-98*, pp. 137–142, 1998.

[117] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, ACM, 1992.

[118] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[119] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[120] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1974.

[121] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[122] N. Fraser, "Neural network follies," 1998.

[123] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[124] L. Wang, *Support vector machines: theory and applications*, vol. 177. Springer Science & Business Media, 2005.

[125] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.

[126] P. D. Gluckman, J. S. Wyatt, D. Azzopardi, R. Ballard, A. D. Edwards, D. M. Ferriero, R. A. Polin, C. M. Robertson, M. Thoresen, A. Whitelaw, *et al.*, "Selective head cooling with mild systemic hypothermia after neonatal encephalopathy: multicentre randomised trial," *The Lancet*, vol. 365, no. 9460, pp. 663–670, 2005.

[127] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, 2009.

[128] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *Proceedings of the ACL student research workshop*, pp. 43–48, Association for Computational Linguistics, 2005.

[129] Z. Zhang, Q. Ye, Z. Zhang, and Y. Li, "Sentiment classification of internet restaurant reviews written in cantonese," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7674–7682, 2011.

[130] D. Bholane Savita and D. Gore, "Sentiment analysis on twitter data using support vector machine,"

[131] M. R. Saleh, M. T. Martín-Valdivia, A. Montejo-Ráez, and L. Ureña-López, "Experiments with svm to classify opinions in different domains," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14799–14804, 2011.

[132] M. D. Conover, B. Gonçalves, J. Ratkiewicz, A. Flammini, and F. Menczer, "Predicting the political alignment of twitter users," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pp. 192–199, IEEE, 2011.

[133] W. G. Baxt, "Application of artificial neural networks to clinical medicine," *The lancet*, vol. 346, no. 8983, pp. 1135–1138, 1995.

[134] H. Kordylewski, D. Graupe, and K. Liu, "A novel large-memory neural network as an aid in medical diagnosis applications," *IEEE Transactions on Information Technology in Biomedicine*, vol. 5, no. 3, pp. 202–209, 2001.

[135] A. Miller, B. Blott, and T. Hames, "Review of neural network applications in medical imaging and signal processing," *Medical and Biological Engineering and Computing*, vol. 30, no. 5, pp. 449–464, 1992.

[136] H. I. Avi-Itzhak, T. A. Diep, and H. Garland, "High accuracy optical character recognition using neural networks with centroid dithering," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 17, no. 2, pp. 218–224, 1995.

[137] C. H. Li and S. C. Park, "A novel algorithm for text categorization using improved back-propagation neural network," in *International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 452–460, Springer, 2006.

[138] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[139] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201–1211, Association for Computational Linguistics, 2012.

[140] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631, p. 1642, 2013.

[141] P. M. Sosa and S. Sadigh, "Twitter sentiment analysis with neural networks," 2016.

[142] W. B. Claster, D. Q. Hung, and S. Shanmuganathan, "Unsupervised artificial neural nets for modeling movie sentiment," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on*, pp. 349–354, IEEE, 2010.

[143] R. Moraes, J. F. Valiati, and W. P. G. Neto, "Document-level sentiment classification: An empirical comparison between svm and ann," *Expert Systems with Applications*, vol. 40, no. 2, pp. 621–633, 2013.

[144] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[145] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, *et al.*, "Tackling the poor assumptions of naive bayes text classifiers," in *ICML*, vol. 3, pp. 616–623, Washington DC), 2003.

[146] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, IBM New York, 2001.

[147] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2, pp. 103–130, 1997.

[148] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," *Information Sciences*, vol. 181, no. 6, pp. 1138–1152, 2011.

[149] S. Tan and J. Zhang, "An empirical study of sentiment analysis for chinese documents," *Expert Systems with applications*, vol. 34, no. 4, pp. 2622–2629, 2008.

[150] Q. Ye, Z. Zhang, and R. Law, "Sentiment classification of online reviews to travel destinations by supervised machine learning approaches," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6527–6535, 2009.

[151] K. T. Durant and M. D. Smith, "Predicting the political sentiment of web log posts using supervised machine learning techniques coupled with feature selection," in *International Workshop on Knowledge Discovery on the Web*, pp. 187–206, Springer, 2006.

[152] P. Melville, W. Gryc, and R. D. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1275–1284, ACM, 2009.

[153] H. Kang, S. J. Yoo, and D. Han, "Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews," *Expert Systems with Applications*, vol. 39, no. 5, pp. 6000–6010, 2012.

[154] P. P. Balage Filho, L. V. Avanço, T. A. S. Pardo, M. d. G. V. Nunes, *et al.*, "Nilc_usp: an improved hybrid system for sentiment analysis in twitter messages.," in *International Workshop on Semantic Evaluation, 8th*, ACL Special Interest Group on the Lexicon-SIGLEX, 2014.

[155] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, ACM, 2004.

[156] S. M. Mohammad and S. Kiritchenko, "Using hashtags to capture fine emotion categories from tweets," *Computational Intelligence*, vol. 31, no. 2, pp. 301–326, 2015.

[157] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *arXiv preprint arXiv:1308.6242*, 2013.

[158] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 2009, p. 12, 2009.

[159] S. Giorgis, A. Rousas, J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos, "aueb. twitter. sentiment at semeval-2016 task 4: A weighted ensemble of svms for twitter sentiment analysis.," in *SemEval@ NAACL-HLT*, pp. 96–99, 2016.

[160] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth international AAAI conference on weblogs and social media*, 2014.

[161] "Sentiment Treebank." `https://nlp.stanford.edu/sentiment/treebank.html/`. [Online; accessed July 2017].

[162] M. Vakulenko, "Chatbots: From hype to "figuring it out"." `https://www.visionmobile.com/blog/2016/09/chatbots-hype`, 2016. [Online; accessed July 2017].

[163] "IBM." `http://www.ibm.com`. [Online; accessed July 2017].

[164] "Chatfuel." `https://chatfuel.com/`. [Online; accessed July 2017].

[165] "Botsify." `https://botsify.com/`. [Online; accessed July 2017].

[166] "Object Management Group - Business Process Model and Notation." `http://www.bpmn.org/`. [Online; accessed July 2017].

[167] N. Bush, "Why you don't need proprietary bot software," *ALICE AI Foundation [online] available at: http://www. alicebot. org/articles/bush/dontpayalotforthatbot-01. html, downloaded on Aug*, vol. 8, p. 2007, 2001.

[168] M. d. G. B. Marietto, R. V. de Aguiar, G. d. O. Barbosa, W. T. Botelho, E. Pimentel, R. d. S. França, and V. L. da Silva, "Artificial intelligence markup language: A brief tutorial," *arXiv preprint arXiv:1307.3091*, 2013.

[169] R. S. Wallace, "The anatomy of alice," in *Parsing the Turing Test*, pp. 181–210, Springer, 2009.

[170] R. Wallace, "AIML 2.0 draft specification released." `http://alicebot.blogspot.pt/2013/01/aiml-20-draft-specification-released.html`, 2013. [Online; accessed July 2017].

[171] "PandoraBots." `http://www.pandorabots.com/`. [Online; accessed July 2017].

[172] S. Writer, "There's a robot in my office." `https://businesstech.co.za/news/business/107713/theres-a-robot-in-my-office/`, 2015. [Online; accessed July 2017].

[173] "GaitoBot - aiml chatbot hosting." `https://www.gaitobot.de/gaitobot/`. [Online; accessed July 2017].

[174] A. Mignogna, "Simple AIML Editor." `https://riotsw.com/sae.html`, 2013. [Online; accessed July 2017].

[175] "RiveScript." `https://www.rivescript.com/`. [Online; accessed July 2017].

[176] "Natural language toolkit." `http://www.nltk.org/`. [Online; accessed July 2017].

[177] "Stanford corenlp – a suite of core nlp tools." `http://stanfordnlp.github.io/CoreNLP/`. [Online; accessed July 2017].

[178] K. Schwaber and J. Sutherland, "The scrum guide-the definitive guide to scrum: The rules of the game," *Available on-line at: http://www. scrum. org/storage/scrumguides/Scrum% 20Guide*, 2016.

[179] H. Ni, A. Chen, and N. Chen, "Some extensions on risk matrix approach," *Safety Science*, vol. 48, no. 10, pp. 1269–1278, 2010.

[180] M. Zuckerberg, "Building Jarvis." `https://www.facebook.com/notes/mark-zuckerberg/building-jarvis/10103347273888091/`, 2016. [Online; accessed July 2017].

[181] "GSM Association." `https://www.gsma.com/`. [Online; accessed July 2017].

[182] " bpmn - BPMN 2.0 execution engine." `https://www.npmjs.com/package/bpmn`. [Online; accessed July 2017].

[183] "AngularJS - Superheroic JavaScript MVW Framework ." `https://angularjs.org/`. [Online; accessed July 2017].

[184] F. P. Brooks Jr, *The mythical man-month, anniversary edition: Essays on software engineering*. Pearson Education, 1995.

[185] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[186] C. Alistair, *Writing effective use cases*. Addison-Wesley Boston, 2001.

[187] D. Clegg and R. Barker, *Case method fast-track: a RAD approach*. Addison-Wesley Longman Publishing Co., Inc., 1994.

[188] I. K. Chaniotis, K.-I. D. Kyriakou, and N. D. Tselikas, "Is node. js a viable option for building modern web applications? a performance evaluation study," *Computing*, vol. 97, no. 10, pp. 1023–1044, 2015.

[189] "Frequently Asked Questions." `https://support.vodafone.co.uk/`. [Online; accessed July 2017].

[190] "Vodafone - Ayuda." `https://ayudacliente.vodafone.es/particulares/temas-de-ayuda/`. [Online; accessed July 2017].

[191] J. HLTCOE, "Semeval-2013 task 2: Sentiment analysis in twitter," *Atlanta, Georgia, USA*, vol. 312, 2013.

[192] S. Rosenthal, A. Ritter, P. Nakov, and V. Stoyanov, "Semeval-2014 task 9: Sentiment analysis in twitter.," in *SemEval@ COLING*, pp. 73–80, 2014.

[193] "LiveJournal." `http://www.livejournal.com/`. [Online; accessed July 2017].

[194] "Sentiment Analysis on Movie Reviews." `https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews`. [Online; accessed July 2017].

[195] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, pp. 115–124, Association for Computational Linguistics, 2005.

[196] G. Vinodhini and R. Chandrasekaran, "Sentiment analysis and opinion mining: a survey," *International Journal*, vol. 2, no. 6, pp. 282–292, 2012.

[197] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[198] B. Boehm, "Software risk management," *ESEC'89*, pp. 1–19, 1989.

[199] "Mobile world congress." `https://www.mobileworldcongress.com/`. [Online; accessed July 2017].

[200] "Communicasia." `http://www.communicasia.com/`. [Online; accessed July 2017].

[201] "Mocha - simple, flexible, fun." `https://mochajs.org/`. [Online; accessed July 2017].

[202] "Chai assertion library." `http://chaijs.com/`. [Online; accessed July 2017].

[203] "Message+ - Frequently Asked Questions." `http://www.vodafone.com/content/apps/ext/messageplus/android/en_au/faqs.html`. [Online; accessed July 2017].

[204] "Call+ - Frequently Asked Questions." `http://www.vodafone.com/content/apps/ext/callplus/android/en_gb/faqs.html`. [Online; accessed July 2017].

[205] "Backup+ - Frequently Asked Questions." `https://www.vodafone.com/content/apps/ext/backupplus/en_gb/faqs.html`. [Online; accessed July 2017].