Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Rasa-ptbr-boilerplate: FLOSS project that enables Brazilian Portuguese chatbot development by non-experts

Autor: Arthur Rocha Temporim de Lacerda

Orientador: Dr. Renato Coral Sampaio

Brasília, DF

2019

Arthur Rocha Temporim de Lacerda

Rasa-ptbr-boilerplate: FLOSS project that enables Brazilian Portuguese chatbot development by non-experts

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Renato Coral Sampaio

Coorientador: Dra. Carla Silva Rocha Aguiar

Brasília, DF

2019

Arthur Rocha Temporim de Lacerda

Rasa-ptbr-boilerplate: FLOSS project that enables Brazilian Portuguese chatbot development by non-experts/ Arthur Rocha Temporim de Lacerda. – Brasília, DF, 2019-

49 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Renato Coral Sampaio

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. Chatbot. 2. Reuse. I. Dr. Renato Coral Sampaio. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Rasa-ptbr-boilerplate: FLOSS project that enables Brazilian Portuguese chatbot development by non-experts

CDU 02:141:005.6

Agradecimentos

I would like to first thank God for all the support given by Him to me on all this work and Software Engineering course. I want to thank too my family, friends and the love of my life Clarice, that helped me a lot on all this journey.

I would like to special thank to my co-Advisor Carla Rocha, for the countless advice as advisor and friend during all my undergraduate, without her participation this work wouldn't be possible. Additionally, I would like to thanks my Advisor Renato Coral, for all the patience and support given during all seconds of this work.

Finally, I would like to thank the lab that changed my heading and gave me huge challenges to forge a University of Brasilia Software Engineer, thank you all from LAPPIS.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Chatbots possuem a capacidade de conversar com pessoas por meio de imitação do comportamento humano. Atualmente, chatbots são capazes de desempenhar tarefas simples como responder perguntas sobre um determinado contexto e desempenhar tarefas complexas como o gerenciamento completo de residências. No entanto, o desenvolvimento de um projeto de chatbot requer uma equipe completa formada por vários especialistas, que podem consumir tempo e recursos.

É comum projetos de chatbots terem requisitos de software semelhantes e apenas se difenciar no domínio da solução específico o que poderia resultar na reutilização de software de código aberto (OSS) relacionado à chatbots. Neste trabalho, é examinado como os projetos de chatbot podem se beneficiar da reutilização no nível do projeto (reutilização de caixa preta). Foi demonstrado que é possível combinar estrategicamente a arquitetura e os diálogos com a utilização do modelo de processo CRISP-DM em novos contextos e propósitos de conversação. A principal contribuição deste trabalho é a apresentação de um projeto de chatbot chamado Rasa-ptbr-boilerplate com configurações e integrações de tecnologias voltado para a reutilização de forma que não especialistas sejam capazes de desenvolver um chatbot como caixa-preta.

Palavras-chave: FLOSS. OSS. FAQ chatbot. Black-Box reuse. Portuguese chatbot. e-government. CRISP-DM. Rasa. Chatbot boilerplate.

Abstract

Chatbots have the ability to talk to people through the imitation of human behavior. Currently, chatbots are able to perform simple tasks such as answering questions about a particular context and performing complex tasks such as complete home management. However, the development of a chatbot project requires a full team of many experts, which can consume time and resources.

It is common for chatbot projects to have similar software requirements and only to differ in the domain of the specific solution which could result in the re-use of open source software (OSS) related to chatbots. In this work, it is examined how chatbot projects can benefit from reuse at the project level (black box reuse). It has been shown that it is possible to strategically combine the architecture and dialogues with the use of CRISP-DM process model in new contexts and conversational purposes. The main contribution of this work is the presentation of a chatbot project called Rasa-ptbr-boilerplate with configurations and integrations of technologies aimed at the reuse so that non-specialists are able to develop a chatbot as a black box.

Keywords: FLOSS. OSS. FAQ chatbot. Black-Box reuse. Portuguese chatbot. e-government. CRISP-DM. Rasa. Chatbot boilerplate.

List of Figures

List of Tables

Lista de abreviaturas e siglas

ML	Machine Learning
NLP	Natural Language Processing
NLU	Natural Language Understanding
CRISP-DM	CRoss Industry Standard Process for Data Mining
FLOSS	Free Libre Open Source Software
TAIS	Tecnologia de Aprendizado Interativo do Salic
SALIC	Sistema de Apoio às Leis de incentivo à Cultura

Contents

1 Introduction

In the last century the question "Can machines think?"was asked by Turing (1950) in his so called 'imitation game' that challenges someone in a dialogue between 3 actors if the answer is being provided bu a real human or a computer. At that time, talking with a machine was a very tricky thing, but nowadays it is getting hard to keep away from chatting with robots. Until recent years, the term chatbot was not widely known, but this technology that simulates conversations has been growing (SCOTT, 2018). Even appearing to be a current concept its history is ancient. The origin comes from the creation of Eliza and its use in the Doctor program (WEIZENBAUM, 1983). Eliza is a "family of programs"created in 1966 by Weizenbaum, which searches for keywords and when it finds them, responds to the sentence according to rules associated with a keyword script. The Doctor was the first program to use Eliza to imitate a psychiatrist.

The definition of a Chatbot is a computer program that interacts with humans through text and audios and has the style of a human-like conversation (SHAIKH, 2016).

Currently, there are examples of chatbots like Siri, Apple's virtual assistant capable of interacting with voice commands, Microsoft's Cortana, and Google Assistant (A.; JOHN, 2015). These are great examples, and part of their success is due to the use of speech as a way of interaction (CALLAWAY; SIMA'AN, 2006).

There are also other chatbots applied to different contexts, which shows that its versatility can be explored. An example is the AgronomoBot a smart chatbot applied to agriculture, which was developed by in a partnership between USP and IFBA to seek and present data collected from wireless sensors implanted in a vineyard (MOSTAçO; CAMPOS; CUGNASCA, 2018). Another example is the study by Dutta, Joyce e Brewer (2018), which shows the viability of the use of chatbots in the context of information security. The study points out that it is possible to use conversational agents to inform more about the subject.

1.1 The Problem

Chatbots can be used in various contexts with many different objectives, but their implementation and maintenance is not easy, and not always a case of success occurs. An example is the SAGA bot, which was implemented in a commercial context, but poor content lifting led it to failure (FILIPCZYK, 2016). This is one of the factors that can hinder or to enable the chatbot application. A wrong choice of technology or its misuse may also compromise the quality of the chatbot.

Even with all these challenges to be faced, LAPPIS (Laboratório Avançado de Produção Pesquisa e Inovação em Software) is developing a virtual assistant (chatbot) in partnership with the Ministry of Citizenship. It is the chatbot called Tais (Tecnologia de Aprendizado Interativo do Salic), and its objective is to assist in the matters related to the Rouanet Law.

Throughout the development of this virtual assistant, several problems were encountered and overcome, but the lack of references in this new research area increased the level of difficulty in this process.

Thus, the objective of this work is to propose a chatbot development process to improve the robot's behavior in an incremental form. A study and survey of metrics for chatbot analysis are also made to measure if the proposal is capable of meeting the requirements.

1.2 Objectives

The main goal of this work is to automate chatbot development activities through a complete pre-configurated project. To achieve this, is it necessary to:

- Identify tools, roles, and gaps present in chatbot projects.
- Understand the chatbot development process and activities.
- Develop a solution with automated pipelines and configurations making the chatbot development as a black box.

1.3 Work Structure

This work is structured as follows:

- **Chapter 1 - Background:** gives an introduction to relevant themes of this work.
- **Chapter 2 - Methodology:** explains how this work was produced and presents its studies and choices.
- **Chapter 3 - Results and Discussion:** reports findings done in this work.
- **Chapter 4 - Conclusion:** presents the impact of all the results and suggest future works.

2 Background

This chapter presents the background necessary for the understanding of the technical content of the solution and problem context. It also presents images and tables with correlated chatbot context data.

2.1 Chatbot

Eliza in 1966, Colby in 1975 and Alice in 2009 are precursors of the modern chatbots. Even succeeding in different periods, they have one characteristic in common, which is being based on rules created by hand (SHUM; HE; LI, 2018). There are now more sophisticated forms and architectures to build chatbots.

Shum, He and Li (2018) describe the architecture of a chatbot with a natural language processing layer, however, it is possible to describe all types of chatbots depending on the technology, or degree of difficulty as can be seen in Figure 1.

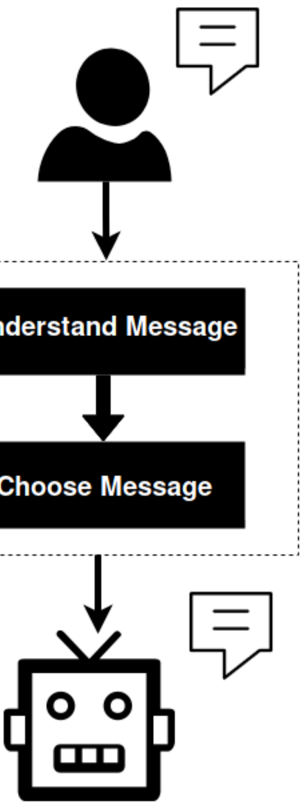

Figure 1 – Chatbot project layers overview

The layers shown in Figure 1 are described as follows:

1. Interpretation (Understand Message): The first step to be taken after receiving the user's message (through speech or text), is to interpret a possible command that the human is tending to inform chatbot. This step can be done by countless forms, such as regular expressions (regex), searching for keywords or patterns in the message, or just checking pre-defined patterns.

2. Core (Choose Message): After identifying the user's intention it is necessary to choose the best response known by the chatbot. Similar to the previous step, there are some ways to implement this functionality. It is necessary to store the responses that chatbot can send using a database or defined files, but the most important point in this step is how to choose the best answer to the question. In this activity, it is also possible to implement ways for the chatbot to understand contexts.

There are some ways to implement the second layer of a chatbot, and the definition of a proper technique of the core is relevant to the chatbot's behavior and evolution. Table 1 describes each message chooser type in an alphabetical order.

Table 1 – Chatbot core layer types

Core layer type	**Description**	**Tools**
Conditional	Made with conditional structures and generaly maps each intention with one response	Hubot
Flow	Made using a pre-defined conversation order built by the chatbot developer	Botkit
Graph	Is similar to Flow, but can go from each point of the conversation to any other	Botpress, Dialog Flow, IBM Watson
Machine Learning (ML)	Made using ML models, the chatbot learns how to answer by given conversations examples	Rasa
Ontology	Made describing the chatbot domain using ontologies	OntBot

About the chatbot development, there are many frameworks, tools, and approaches. The following items describe the most prominent tools in alphabetical order.

1. **Botkit:** a chatbot platform that has javascript as its primary programming language, one of its main features is the Botkit Studio API where it is possible to map the actions of chatbots online. It is open source however, there is a limit on using the provided API (XOXCO Inc, 2017).

2. **Botpress:** an on-prem, OSS bot-building platform. It delivers a dialogue flow editor to manage the chatbot behavior combined with NLU layer, analytics to see how the bot usage, multi-channel integration with messengers, authoring-UI to non-developers manage the deployed chatbot and SDK & API to manage the integration with external software (BOTPRESS, 2019).

3. **Dialog Flow:** an online platform that allows the creation of chatbots. Google Inc is supporting one of its strengths. Its focus is to build chatbots without the need to enter a code. (Google Inc, 2019)

4. **Hubot:** a chatbot platform made by GitHub to automate the company's chat room. Is made using CoffeeScript on Node.js and is OSS. To develop a chatbot using this software is needed to develop functions responsible for answering each user message, as the functions are in *JS* (JavaScript) it is possible to make API calls, math operations, and other things when handling a user message. Hubot can understand user messages by regex operations too (GitHub Inc, 2019).

5. **IBM Watson:** a complete platform for developing chatbots with both paid license plans and free versions. Different from RASA and Botkit, it is a closed source, proprietary solution.

6. **OntBot:** an ontology-based approach to model and operate chatbots. This solution has a knowledge base with the resulted mapped ontological tables, an NLP module to process user input, an Inference Engine which is the main component of the architecture and submodules responsible for choosing the chatbot's answer. (AL-ZUBAIDE; ISSA, 2011)

7. **RASA:** is an open-source framework that uses machine learning for creating chatbots that can understand contexts. Its implementation is mainly done in the Python programming language. Its first release was in October 2017 and it already has an active community. This framework is responsible for choosing what the best response the chatbot should send to the user is. It does this through machine learning algorithms and conversations examples. The framework has a layer to understand user messages, this the part concerning the interpretation framework, it also uses artificial intelligence algorithms to extract the user's intention from messages and sends it to the RASA core (Rasa Technologies GmbH, 2018).

There are other tools with distinct features in the context of chatbots, as shown in Figure 2, presented in Gregori (2017). This Figure shows tools related to chatbot development and presents providers that use these tools in software generally with easy user interfaces. The blue layer presents companies that develop chatbot solutions to their customers. The center circle presents messenger platforms, which are the main channel between users and chatbots.

Although in Figure 2 there are many examples of tools, providers, customers, and messengers, it still omitted some important players in the chatbot/virtual assistant world, such as Apple, Facebook, Google, Microsoft, and Samsung, each of which embedded these technologies on their respective computational platforms.

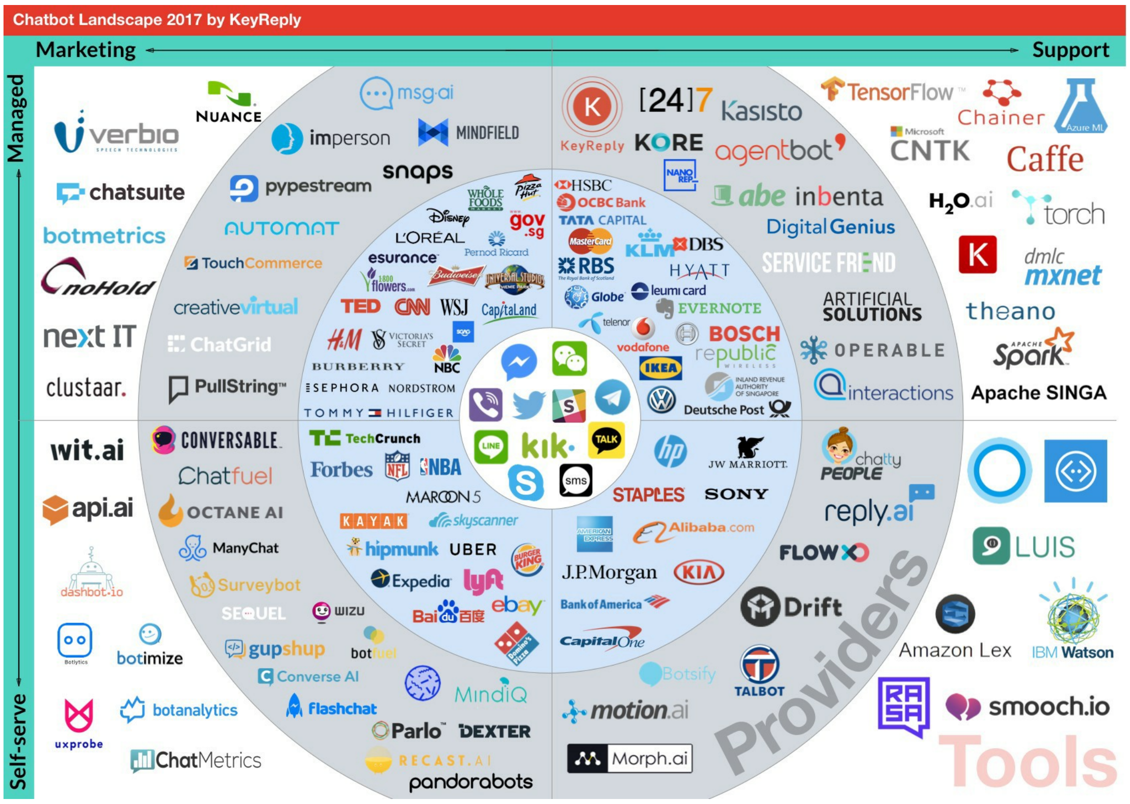

Figure 2 – Chatbot landscape

Source: (BRADEšKO; MLADENIć,)

It is also relevant to classify chatbots independent of technologies or the tools they use, but rather according to the function that the conversational agent exercises. In this context, chatbots can be classified into informative, collaborative, and autonomous according to a study done by Paikari e Hoek (2018).

Informative chatbots search for information that may be relevant for the activity being performed, such as showing the weather for a cyclist before leaving to pedal. *Collaborative* Conversational Assistants are those that aim to enrich contexts in order to complement the activity that is running, such as reporting the steps of a recipe for a cook. Finally, there are *autonomous* chatbots, which are those able to complete activities that affect some part of the final result alone. As a virtual assistant that serves users and after identifying the problem redirects to the right industry.

2.2 CRISP-DM

Natural Language Processing (NLP) is a common part of chatbot development, and ML can be used in other chatbot parts too, so an ML process can be applied to this part of a chatbot project. The CRoss Industry Standard Process for Data Mining (CRISP-DM) is one process that aims to make data mining projects more repeatable (WIRTH; HIPP, 2000). Figure 3 presents a diagram of the CRISP-DM process, according to Wirth e Hipp (2000) which activities are described below.

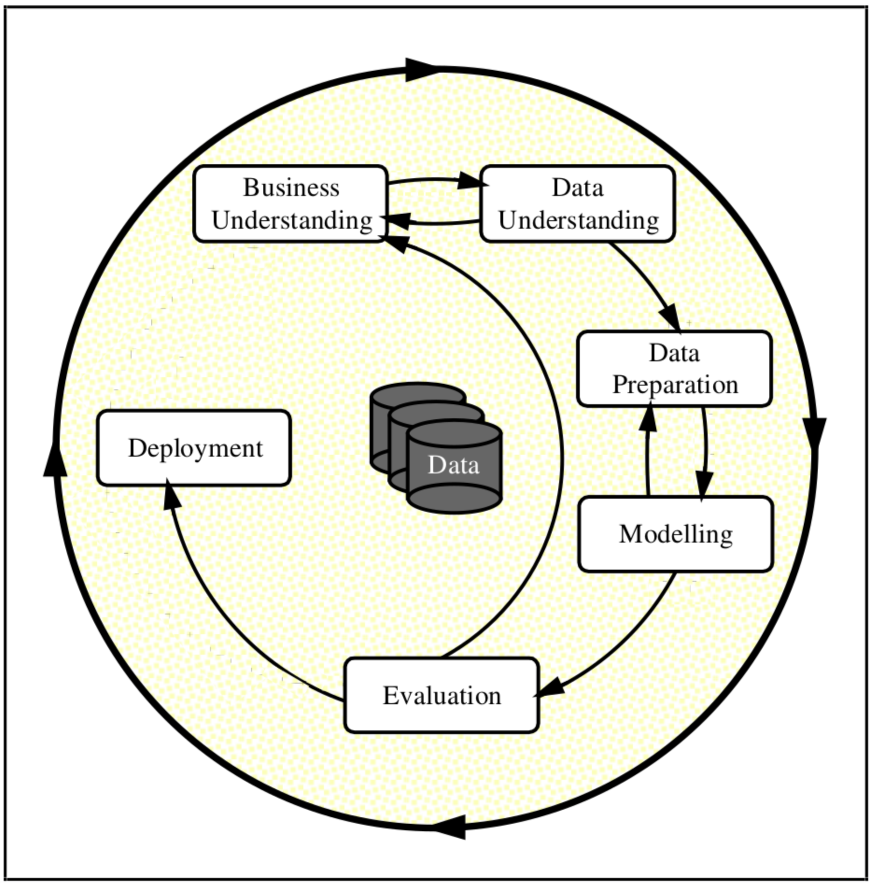

Figure 3 – CRISP-DM phases

Source: (WIRTH; HIPP, 2000)

- **Business Understanding:** This first phase is focused on understanding the project objectives and requirements from a business knowledge perspective into a data mining problem definition;

- **Data Understanding:** The objective of this activity is to get familiar with the data identifying data quality problems and form hypotheses;

- **Data Preparation:** This phase covers all activities to construct the final dataset from the initial raw data;

- **Modeling:** In this activity modeling techniques are applied. Data Preparation and Modeling have a close link with data problems realization and ideas for constructing new data;

- **Evaluation:** On this activity high-quality models from previous activities should be used, and the evaluation of these models is an important activity before the deployment of it;

- **Deployment:** The knowledge gained need sto be organized and presented in a way that the customer can use it, this is the main objective of this activity.

2.3 Reuse in Chatbot Projects

Code reuse allows for previously tested and quality-assured code to be implemented in another system and provides benefits by simply adding and enhancing system features (PRIETO-DIAZ, 1993). In chatbot projects, both white-box and black-box reuses are common. However, most of the research in literature focuses on the reuse of training datasets (PAETZEL et al., 2018; GAO; XU; CALLISON-BURCH, 2015).

A great challenge is the effectiveness of chatbots in non-English speaking countries. When the objective is to build a chatbot in languages other than English, dialogue datasets are the most significant difficulty (GAO; XU; CALLISON-BURCH, 2015), although it is still possible. In (TAVANAPOUR; BITTNER, 2018), a German chatbot was built to guide an idea submission process. ParlAI is an open-source platform that aims to minimize this dataset by providing a unified framework for sharing, training, and testing dialog models (MILLER et al., 2017). It aims to reinforce reuse of training datasets by sharing a repository of the corpus, utterances, and machine learning models. Another alternative is the adoption of a crowdsourcing approach to write utterances. Paetzel et al. in (PAETZEL et al., 2018) evaluate how untrained crowd workers (crowdsourcing) can scale chatbots and still maintain coherence in the chatbot personality, affective behavior, and vocabulary (Lacerda; Aguiar, forthcoming).

2.4 Free Libre Open Source Software (FLOSS)

Free Libre Open Source Software or Free Software is defined as the one that respects the freedom of the user to use it for any purpose, adapt the software to meet new needs, the freedom to distribute copies and the freedom to modify the program and redistribute copies for any purpose. These guaranties are known as the four fundamental freedoms of Free Software and were defined by the GNU project (FOUNDATION, 2019).

Free Software projects present themselves in society as an alternative of ethical and socially responsible technological practice, respecting the time and space, including users, as potential contributors and thus avoiding the natural problems caused by the belief in technology neutrality, common in the philosophical perspectives of Instrumentalism and Determinism, currents of thought which take off and disconnect any social, environmental or individual impact whether positive or negative caused by technological development (NEDER, 2010).

In the academic and scientific context, Free Software offers unique opportunities, especially in Software Engineering, as it can be used for educational purposes as well as engineering practices since its source code is freely and publicly available. Open source software development communities are often distributed globally, are widely available, generating a huge range of possibilities for study and Software Engineering practices and techniques, based on and real-world products (KON et al., 2011).

Still, in the face of the opportunities for Science, these Free Software products can be used as objects of academic study since both source code and process, usually with globally distributed teams, are freely available, these invaluable data to test hypotheses, theories, and laws in real contexts, using mature and robust products. The development and contribution to Software tools Free in research and scientific studies, it also provides a significant increase in the capacity reproduction of scientific studies, an important requirement for the advancement and consolidation of the knowledge generated in such works (KON et al., 2011).

Today there are some motivations to use Open Source Software, and one of them is the actual usage of this kind of solution. Open source is how modern organizations and traditional organizations build software. IBM, Adobe, and Microsoft are participating in the Open source community for example. (BALTER, 2015). According to Asay (2018) it is possible to cite more organizations that have employees actively contributing to open source projects on GitHub, for example, Microsoft, Google, Red Hat, IBM, Intel, Amazon.com, SAP, ThoughtWorks, Alibaba and GitHub.

3 Methodology

This section describes the process, assets, tools, and technology choices used to develop this work, as well as the methods that are being used to enable reuse of chatbot development.

This work had three main phases, namely Immersion, Knowledge Abstraction, and Implementation which are described below:

- **Imersion** is the participation in a chatbot development project (Tais) in a real context, with the objective to understand all important roles, tools, and activities necessary to develop a chatbot.
- **Knowledge Abstraction** is the study and organization of the data collected in the previous phase aiming to understand what is generic, specific, and reusable in a chatbot development process. Diagrams and descriptions were made in this phase.
- **Implementation** is the phase where the project was defined and developed using data from previous phases. The objective of it was the application of the proposed solution. Software Development methods like XP and DevOps were applied.

To synthesize the study of the chatbot, the first step was to understand a chatbot development process. For this purpose the CRISP-DM was chosen and studied to decribe the activities and the relationship between actors, activities, assets, and tools. An adaptation and a description of each process phase applied to chatbot development was done.

Next, the chosen chatbot development framework on the real project was the RASA, which is ML based. Rasa is the standard infrastructure layer for conversational AI, as shown in previews sections. With this tool, the chatbot development can be done in the message understanding layer and message choosing layer. The context of the development of a chatbot in this work was defined in Rasa chatbots too, so the comparison and results are mainly about this technology usage.

Messenger tools are necessary to be the interface between the user and the chatbot. Two messengers were chosen to be used in this work namely RocketChat and Telegram. This was also due to the previous knowledge about these technologies while developing Tais and the ease of configuration of it using Rasa as the chatbot development tool.

Jupyter Notebook was the tool used to understand the bot's behavior with Rasa and python methods and functions. With evaluation methods given by Rasa, it is possible to check and improve the chatbot even without its deployment, checking the similarity and confidence of message understanding and message choosing confidence given by Rasa, and other data can be used with jupyter-notebooks like confusion matrixes and conversation graphs given by Rasa too.

Tools to understand the chatbot usage by the user are necessary too. In this case, Kibana and ElasticSearch are used to collect data and generate metrics about users, messages, behaviors and users intentions. With this kind of tool, it is possible to analyze a massive amount of data through indicators.

4 Results and Discussion

4.1 Results

This section presents the results achieved in this work. Some of the results were used as a base to validate ideas and made the boilerplate project evolve. The main subject of this section is the rasa-ptbr-boilerplate, which is the proposed solution to this work. The boilerplate has begun with the objective to have all configurations of the Tais chatbot, generic data, and tool integrations able to use as a black box to other chatbot projects.

This solution is a FLOSS Rasa boilerplate named 'rasa-ptbr-boilerplate' and can be found at http://github.com/lappis-unb/rasa-ptbr-boilerplate.

In the following sections, I describe all the information, data, tools, configurations, and content identified that could be reused in other chatbot projects according to the defined methodology phases.

4.1.1 Imersion

The immersion phase occurred during a project development in a software engineering laboratory at the University of Brasília (UnB) called LAPPIS. The objective of the project was to build a chatbot to help Brazilian citizens to understand a cultural law.

The work presented here is about the development of a chatbot called Tais (Tecnologia de Aprendizado Interativo do Salic), made by LAPPIS. It is a conversational assistant with the objective of helping people understand a culture law in Brazil. The project began in October 2017, and had over 31 students and professionals participating in its development. Eeach member with different backgrounds such as designers, data scientists, and software engineers. I participated in it from March 2018 until May 2019, in a total of 15 months as a Software Engineer student.

Using a FAQ history and other documents given by Special Secretary of Culture from Brazil, we from LAPPIS started to understand these assets and convert it to "chatbot interactions."At the beginning of the project, Tais did not have a good interactions but it evolved over time until it reached a satisfactory user interaction experience. Each of its evolution iterations were made possible through better Rasa configurations and machine learning parameters.

Tais was mainly developed using python and Rasa as a chatbot framework. During the development of Tais, many issues and problems were overcome by using analysis of chatbot conversations through data collection, tunning of all configurations of the hyperparameters of machine learning in the two layers and the integration between messenger tools, chatbot framework and analytics tools. All based on FLOSS solutions.

Tais' codebase can be accessed at GitHub in this link: https://github.com/lappis-unb/tais. Table 2 presents some statistic from the GitHub repository listed in alphabetical order. The service is online and accessible at http://leideincentivoacultura.cultura.gov.br.

Table 2 – Tais project data collected at 11/06/2019

Item	Value
Commits	1258
Contributtors	31
Forks	16
Issues	392
License	GPL-3.0
Pull Requests	124
Releases	19
Stars	33

The immersion period made it possible to identify tools, assets, contents, and activities that could be reused in other chatbot projects. In the next section I show what information was selected to be reused.

4.1.2 Knowledge Abstraction

Each Rasa chatbot project has some common configurations that can be pre-configurated and pre-written. After that initial setup, inserting the base data for the specific chatbot would be sufficient to have a working version. However, to achieve a good solution, the development of a chatbot still needs specific actors with different backgrounds. These roles were identified, and their relationship is shown in Figure 4.

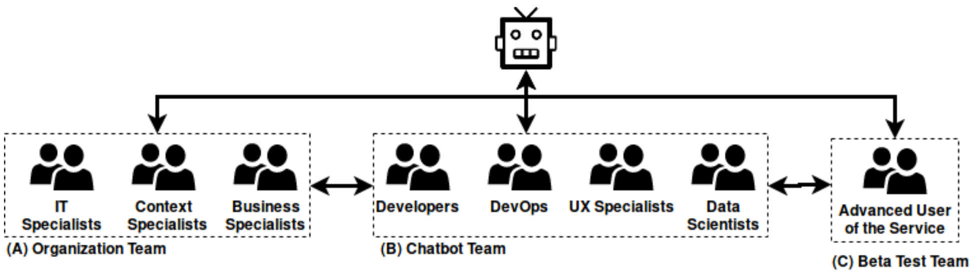

Figure 4 – Actual chatbot development necessary roles (a),(b),(c).

The "Chatbot Team" shown as B (Figure 4) presents the roles identified as needed to develop a Rasa chatbot. Developers are needed to implement python functions and build an integration between the necessary tools. The DevOps role is vital to grant the deployment of the chatbot and its maintenance and evolution. UX Specialists take care

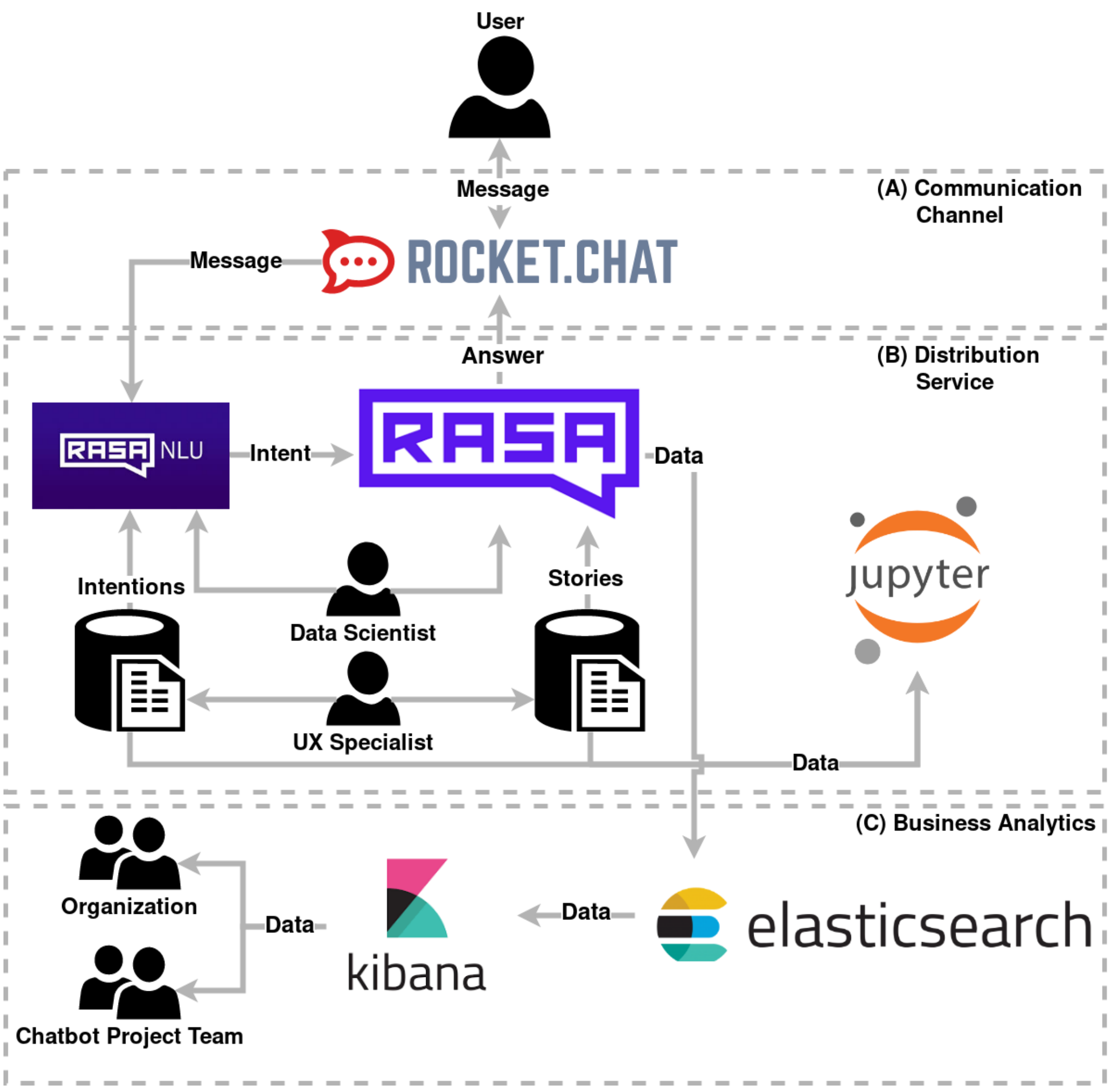

Figure 5 – Tais project overview.

Source: (Lacerda; Aguiar, forthcoming)

of all dialogue information, granting that the chatbot messages answer the user questions without problems. The Data Scientists are essential to calibrate hyperparameters and improve the chatbot behavior and the Rasa NLU and Core layers.

In Tais' case, it can understand 83 user intentions, but 70 of these intents are specific of Tais' culture law context, and 13 intents could be used as generic intents. About Rasa stories and utters, these data were not directly selected to be reused as intents, and it just was used as a base to create new generic stories and utters.

But not just roles and content data were selected to be reused. During the development of Tais, tool integrations were necessary. Figure 5 shows the relationship between actors, tools and content in the Tais chatbot development. It is important to see the integration made between Rasa and RocketChat, ElasticSearch, Kibana, and jupyter-notebooks.

Figure 5 is divided into three layers, (A) Distribution is the layer that interfaces chatbot with the user, in the Tais project, Rocketchat was the messenger used. (B) Creation is the layer with all tools and data needed to develop the chatbot. Rasa is the primary tool with the Intentions and Stories, jupyter is important to use Rasa evaluation functions and improve the chatbot behavior. (C) Business Analytics is important to understand the usage of the chatbot by collecting data about dialogues and storing it using ElasticSearch, and with Kibana metrics, dashboards, and graphs are generated to interpret all this information.

During the development of this work, some roles were identified as relevant to the chatbot development, each one has a specific activity, and the same actor can perform some of them. Indeed, one objective of the boilerplate is to minimize the need for expert actors performing each role. Table 3 shows each identified role and its description.

Table 3 – Chatbot roles overview

Roles	**Description**
User	Responsible for having conversations with the chatbot
Data Scientist	Responsible for maintaining and evolving the chatbot models and machine learning algorithms
UX Specialist	Responsible for improving the chatbot interactions like messages, images, buttons and any other thing that will be sent to the user
Maintainer	Responsible for maintaining the chatbot project and making necessary choices about it
Chatbot Specialist	Responsible for maintaining and evolving the chatbot software in any part of it
DevOps Specialist	Responsible for granting the reliability of the chatbot solution and user access to it

With the proposed boilerplate, the need for specialists could be minimized in the chatbot development. In this case, developers should be enabled to perform the specialist roles following the tutorials and using the pre-configured tools. Figure 6 presents the roles needed to develop a chatbot using the boilerplate and their relationship.

Another important result was the identification of the assets needed to develop a chatbot. Each asset is important to some part of the development, deployment or evolution of the solution. On Table 4 these assets is shown and the tools that are implemented in the solution.

All the knowledge abstraction was required to understand the needs of a chatbot project. With the results shown in Figure 3 and the chosen tools presented in Table 4,

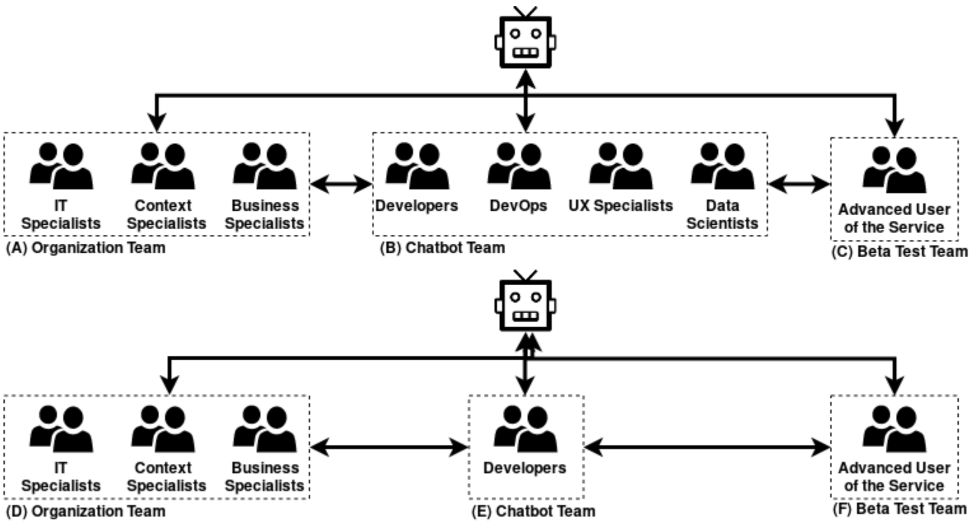

Figure 6 – Chatbot Roles achieved with boilerplate (d),(e),(f).

Source: (Lacerda; Aguiar, forthcoming)

Table 4 – Chatbot tools overview.

Tools	**Description**	**Rasa-ptbr-boilerplate Tools**
Messenger	A software that allows user and chatbots to have a conversation	Rocketchat, Telegram
Message Understander	A layer in the chatbot solution to understand user messages	Rasa (nlu)
Message Chooser	A layer in the chatbot solution to choose the best message to send to the user	Rasa (core)
Middleware	A layer in the chatbot solution to make the connection between the chatbot and any other external applications	Rasa (Custom Actions)
User Intention Data	A database about classification of user messages into intentions	Rasa Intents files
Dialogue Data	A database with conversation examples to train the chatbot	Rasa Stories files

enough knowledge was collected to begin the next step of this work, the implementation.

4.1.3 Implementation

The proposed solution is a Rasa boilerplate with content data in the Portuguese language, capable of abstracting Rasa chatbot configuration, environment, and context data. A GitHub project was created and can be accessed in the following link: http://github.com/lappis-unb/rasa-ptbr-boilerplate. This project comes with tools integrations, generic Rasa dialogues pre-configurated, and a docker pre-configurated, all of it is to achieve the objective of minimizing the development of a chatbot and the chatbot implementation as black-box.

The first step of the implementation was to clone the Tais project, with all Tais project files. I created the boilerplate project on GitHub and added the boilerplate remote link to Tais' project cloned in my environment. With the GitHub configuration done, I started to remove all specific content data from Tais (namely content about culture law), and made the docker environment variables generic. To make the repository more light, I removed all Tais commits, so the first commit of the boilerplate has the Tais knowledge but generically as a start point. It is possible to see every configuration and content in the first commit in this link: https://github.com/lappis-unb/rasa-ptbr-boilerplate/commit/c216c4438faeff3e40b63d15746fa9c8acd9f298.

The boilerplate starts with a generic dialog data pre-configurated, which have 13 intents from Tais and other 19 which were added to reach a total of 32 generic intents that could be reused in other chatbot contexts. Table 5 shows the title of all of these intents.

Table 5 – Generic intents selected to be reused

Intent		
cumprimentar	time	de_onde_voce_eh
despedir	linguagens	relationship
out_of_scope	genero	me
negar	star_wars	filhos
diga_mais	piada	filme
tudo_bem	license	signo
elogios	onde_voce_mora	
religiao	como_estou	triste
esporte	playlist	hobby
comida	cor	bff
historia	risada	action_test

The **rasa-ptbr-boilerplate** is a project in evolution but already has given some data that shows the project could reach its propose. In the next section, I present the results and data of the boilerplate project.

After the chatbot development immersion, some tools and roles were identified as necessary to implement the solution, but during the conduction of this study, its clear that adaptions to each asset, role, tool, context data and any other information relevant to chatbot project development needs to be evaluated, planned and applied to each new project.

4.1.4 Rasa-ptbr-boilerplate overview

The boilerplate has begun with Tais project configurations, but evolved to explore more of the Rasa's technology in an attempt to attend the needs of other chatbot projects. To achieve this, the boilerplate the has interactions between tools, data and roles described in Figure 7.

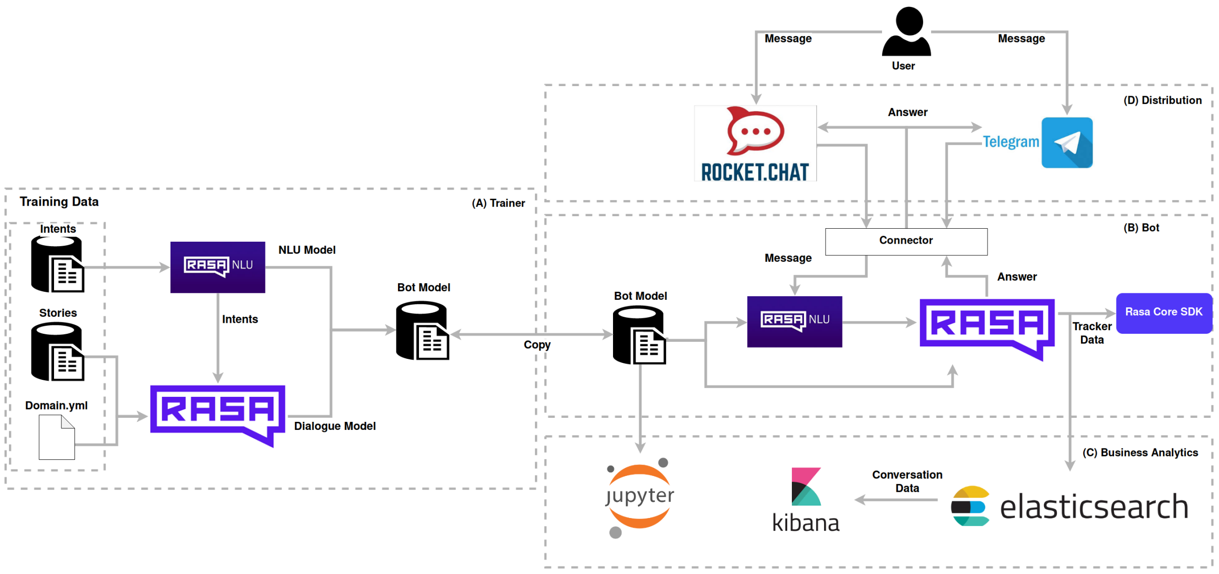

Figure 7 – Rasa-ptbr-boilerplate project overview.

Figure 7 summarizes the rasa-ptbr-boilerplate with the relationship of important parts like dialogue NLU and core datasets, config files, tools relationships like Rasa integrations and actor interactions as user messages sending through messengers. All of it is delivered as a black box by the boilerplate to new developers and users, so it will not be needed to build all this architecture again and all the tools integrations.

The boilerplate aims to minimize the chatbot development by delivering a black box solution, until the date of the publication of this work, some other projects started to use the boilerplate and built chatbots using this solution as the base. Table 6, shows a comparison of chatbot projects:

- **Rasa-ptbr-boilerplate** is the base project with all the development needed configurations. This project can be accessed in this link: https://github.com/lappis-unb/rasa-ptbr-boilerplate.

- **Lappisudo** is an informative chatbot with dialogues about LAPPIS. This project is a fork of the boilerplate and was made by the same team that developed the boilerplate. This project can be accessed in this link: https://github.com/lappis-unb/lappisudo.

- **Dirce** is a chatbot made to help Brazilian people that live in São Paulo and need to interact with the public defense about law issues. This project is a fork of the boilerplate and was made by the same team that developed the boilerplate. This project can be accessed in this link: http://github.com/defensoriapublicasp/dirce.

- **IEEE** is a chatbot made by IEEE Computer Society UnB. This project is a fork of the boilerplate and has no relationship with the team that developed the boilerplate. This project can be accessed in this link: https://github.com/IEEEComputerSocietyUNB/rasa-ptbr-boilerplate.

Table 6 – Chatbot projects data collected at 11/06/2019

	Boilerplate	**Lappisudo**	**Dirce**	**IEEE**
Commits	78	97	114	41
Contributtors	10	9	4	8
Forks	29	1	1	0
Issues	19	19	20	0
License	GPL-3.0	GPL-3.0	GPL-3.0	GPL-3.0
Pull Requests	18	6	7	1
Releases	5	2	2	3
Stars	18	0	4	0

It is important to see that the number of commits are different and can be lower than the boilerplate project, as the IEEE project because of the fork time of the project. An early fork can make the forked project be with fewer commits and status than the base project because it keeps evolving and receiving more commits.

4.1.5 Chatbot Development using CRISP-DM

The following sections describe the chatbot solution using the CRISP-DM activities and the participation of the boilerplate in each activity. Each section describes the CRISP-DM activity with a description, the inputs needed to perform the activity, tools suggested to be used, roles that are important to the activity and the outcome. Additionally a Rasa-ptbr-boilerplate role in each topic is highlighted in each section.

It is important to say that the information shown in Table 3 and Table 4 are linked with chatbot development using CRISP-DM.

4.1.5.1 Business Understanding

This is the first step of a chatbot project. It is common activity in many different documents as being part of any software development effort.

One of the most common problems in a chatbot development is to manage the expectations of the customers, and it is worse when the customer does not understand how chatbots work and what is the software limitations and conversation limitations. One example of difficulty is knowing when you have enough data to begin the development. When you think you already know everything about the project but the Business Understanding is not clear, probably the project will have some issues.

- **Inputs:** FAQ history, Human attendance documents, customer chatbot descriptions.
- **Tools:** Software requirement elicitation techniques are good tools for this activity like brainstorming and interview.
- **Roles:** For this step a business specialist and a chatbot specialist are necessary to converge in the data understanding.
- **Outcome:** Dialogue diagrams with organized chatbot knowledge and good understanding by each chatbot project stakeholder.
- **Rasa-ptbr-boilerplate role:** The boilerplate does not give any specific tool to this part of the chatbot development.

4.1.5.2 Data Understanding

After all the needed data was collected, it is possible to start the development of the chatbot. The first step is to raise some hypothesis of good conversations. All data collected at the Business Understanding step is used for the first chatbot conversation development. A good procedure is to divide and classify each document looking for the better transformation of it into chatbot conversation data.

The main difficulty of this activity is understanding the customer context, knowledge and specific gaps, because without it, the outcome data could be shallow.

- **Inputs:** Dialogue diagrams and organized context knowledge.
- **Tools:** Jupyter notebooks, spreadsheets and mind maps are good tools to document and give an overview of the chatbot project data.
- **Roles:** Data scientist, UX specialist, chatbot specialist and the project maintainer are the core roles needed for this activity.
- **Outcome:** Improved dialogue diagrams, mind maps and a defined data set.
- **Rasa-ptbr-boilerplate role:** The boilerplate does not give any specific tool to this part of the chatbot development.

4.1.5.3 Data Preparation

To transform the collected data into conversations, it is necessary to understand data as user intentions and data as chabot messages. These two steps are essential to every other step of the chatbot development.

The data that the maintainer shows is commonly not what the user interacts with. A log with all attendance made by humans is not enough to understand how to convert questions of FAQ like data into chatbot interactions. It is important to identify all main intentions with given data but this activity is difficult too.

- **Inputs:** Reliable dialogue diagrams, organized chatbot knowledge and dataset definition.
- **Tools:** Dialogue diagrams with intents and linked answers are important tools to verify if the actual dataset has the desired behavior.
- **Roles:** Chatbot specialist is the main role in this activity, but the data scientist is essential too for this step.
- **Outcome:** Usable chatbot dialogue datasets.
- **Rasa-ptbr-boilerplate role:** The boilerplate can help on this step by using the tutorials and documentation about how to organize the chatbot dialogue content.

4.1.5.4 Modeling

Now, with any information collected, it is necessary to start the **conversation modeling**. Summing up, this step's main result is the chatbot's **intents** and **stories**.

The difficulties of modeling are on how to:

- Write each user intention;
- Generate user intentions examples;
- Write all chatbot messages (utters);
- To structure all the conversation into Rasa stories structure.

This step is responsible for linking all user intentions with chatbot messages, so mistakes at this point can damage the chatbot behavior.

- **Inputs:** Naive chatbot data set.
- **Tools:** Jupyter notebook is a good tool for the management of datasets and is suggested to chatbot data sets too.

- **Roles:** Data Scientist is the main role of this activity, but the chatbot specialist is recommended too.
- **Outcome:** NLU and Core trained models.
- **Rasa-ptbr-boilerplate role:** The boilerplate is useful for this activity with pre-configurated Rasa evaluation methods about NLU and core parts implemented in jupyter notebooks.

4.1.5.5 Evaluation

After the dataset is developed and before the deployment, it is necessary to check if the generated chatbot model is good to the desired objective. The main part of this activity is to deliver the chatbot to a beta test group and collect the dialogue data. With the indicators from the conversation, more improvements in any part of the chatbot project should be applied.

- **Inputs:** NLU and Core updated models.
- **Tools:** Dialogue analytics tools.
- **Roles:** Data scientist are important to define and interpret the dialogue data, chatbot and UX specialists are needed too to avoid miss understandings. User is another participant role in the application of beta tests.
- **Outcome:** Chatbot dialogue behavior.
- **Rasa-ptbr-boilerplate role:** The boilerplate is helpful in this step by using the Kibana with ElasticSearch configuration. In this platform, it is possible to manage dashboards and charts with dialogue data.

4.1.5.6 Deployment

An important thing about the deployment is collecting more data about chatbot usage. When the bot is up and running to the final users, the way that each user sends and interacts with messages needs to be collected and analyzed. Chatbot frameworks can give many ways to make the user interact with the chatbot, so the choosing of message platforms and configuration of it is an important steps of this activity.

- **Inputs:** Chatbot dialogue behavior.
- **Tools:** DevOps tools and techniques are useful for this step. All the project integrations need to be done in this step too.

- **Roles:** DevOps team is the main role needed for this activity. User and Maintainer are related to this activity roles too.
- **Outcome:** Access point to the chatbot with data dialogue collection.
- **Rasa-ptbr-boilerplate role:** The boilerplate has pre-configurated integration between messengers, Rasa and analytics tools. All these integrations are made using docker containers, so this step is pretty assisted by the boilerplate.

With all this information the boilerplate is the proposed solution to get all the data, assets, roles and tools organized and enabling non-experts to develop a chatbot as a black box. Table 7 summarizew all this information about chatbot development using CRISP-DM.

Table 7 – CRISP-DM applyied to chatbot

Process Activities	**Roles**	**Tool Types**
Business Understanding	UX Specialist, Maintainer and Chatbot Specialist	Requirements elicitation techniques
Data Understanding	UX Specialist, Maintainer, Chatbot Specialist and DevOps Specialist	Xmind and Draw.io
Data Preparation	Data Scientist, UX Specialist, Chatbot Specialist and DevOps Specialist	Rasa Intents and Stories
Modeling	Data Scientist, Maintainer, Chatbot Specialist and DevOps Specialist	Rasa NLU evaluation methods, Rasa Core evaluation methods, Rasa Intents and Stories, and jupyter
Evaluation	User, Data Scientist, UX Specialist, Maintainer, Chatbot Specialist and DevOps Specialist	Rasa Intents and Stories, Kibana with elasticsearch dashboards and charts and Rasa evaluation methods
Deployment	User, Maintainer, Chatbot Specialist and DevOps Specialist	Messenger, Rasa Intents and Stories, Docker and docker-compose

4.2 Discussion

During my participation on the Tais chatbot project, it was clear to me that this kind of software has specific gaps and problems, but I realized that it is possible to reuse some solutions of it in other chatbot projects, avoiding the need for specialists and more effort.

Figure 4 presents the initial roles needed to a chatbot development. But not just specific roles are needed. Tools integrations and configuration are important activities in chatbot development. With this work, it was possible to understand and improve the chatbot architecture showed in Figure 5 to the architecture shown in Figure 7.

Beyond the technical solution, the Rasa-ptbr-boilerplate has been used as the base for more than 20 projects as shown by the GitHub forks data on Table 6, this is an important number to this specific boilerplate solution in less than a year of its creation.

Unfortunately it's not possible to affirm that the boilerplate has solved the initial objectives from this work with the actual data and information. Further studies need to be done to affirm that the boilerplate is enabling non-experts to develop a chatbot as a black box.

5 Conclusion

The development of FAQ chatbot projects is similar among different contexts. Activities such as the search for technologies, determine the best ones to the project, configure and integrate each service of the solution are necessary, alongside with content creation, as designing the interaction.

In this work, the experience and knowledge gathered from a FAQ chatbot project applied to a boilerplate project aiming to make the development of other chatbots be easier with its usage is presented. The CRISP-DM process was raised as a hypothesis as a machine learning chatbot project process.

The goals defined in this work were reached by identifying the data, tools, roles and activities related to a chatbot development through the immersion in a running chatbot project. The understanding of this context was reached by tools and roles relationships associated with the CRISP-DM process model. Finally, a boilerplate project was developed as a solution to the gaps identified in this work.

5.1 Future Works

Here we proposed the usage of CRISP-DM in chatbot development, but the application of this process was not made and measured. This is an important part of validating the hypothesis raised here.

It is important to verify if the boilerplate is achieving its purpose. A suggestion is to measure 2 RASA Portuguese chatbot projects built with and without the boilerplate and evaluate if the development using the boilerplate present positive results.

Referências

A., S.; JOHN, D. Survey on Chatbot Design Techniques in Speech Conversation Systems. *International Journal of Advanced Computer Science and Applications*, v. 6, n. 7, 2015. ISSN 21565570, 2158107X. Disponível em: <http://thesai.org/Publications/ViewPaper?Volume=6&Issue=7&Code=ijacsa&SerialNo=12>. Cited in page 19.

AL-ZUBAIDE, H.; ISSA, A. A. OntBot: Ontology based chatbot. In: *International Symposium on Innovations in Information and Communications Technology*. Amman, Jordan: IEEE, 2011. p. 7–12. ISBN 978-1-61284-675-0 978-1-61284-672-9 978-1-61284-674-3. Disponível em: <http://ieeexplore.ieee.org/document/6149594/>. Cited in page 23.

ASAY, M. *Who really contributes to open source*. 2018. Disponível em: <https://www.infoworld.com/article/3253948/who-really-contributes-to-open-source.html>. Cited in page 27.

BALTER, B. *Six reasons why you might consider open source*. 2015. Disponível em: <https://opensource.com/life/15/12/why-open-source>. Cited in page 27.

BOTPRESS. *Botpress - A Chatbot Maker & Development Framework*. 2019. Disponível em: <https://botpress.io/>. Cited in page 23.

BRADEšKO, L.; MLADENIć, D. A Survey of Chabot Systems through a Loebner Prize Competition. p. 4. Cited in page 24.

CALLAWAY, C.; SIMA'AN, K. Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship. *Computational Linguistics*, v. 32, n. 3, p. 451–452, set. 2006. ISSN 0891-2017, 1530-9312. Disponível em: <http://www.mitpressjournals.org/doi/10.1162/coli.2006.32.3.451>. Cited in page 19.

DUTTA, S.; JOYCE, G.; BREWER, J. Utilizing Chatbots to Increase the Efficacy of Information Security Practitioners. In: NICHOLSON, D. (Ed.). *Advances in Human Factors in Cybersecurity*. Cham: Springer International Publishing, 2018. v. 593, p. 237–243. ISBN 978-3-319-60584-5 978-3-319-60585-2. Disponível em: <http://link.springer.com/10.1007/978-3-319-60585-2_22>. Cited in page 19.

FILIPCZYK, B. Chapter 12 - Success and failure in improvement of knowledge delivery to customers using chatbot—result of a case study in a Polish SME. p. 15, 2016. Cited in page 19.

FOUNDATION, I. F. S. *What is free software? - GNU Project - Free Software Foundation*. 2019. Disponível em: <https://www.gnu.org/philosophy/free-sw.en.html>. Cited in page 26.

GAO, M.; XU, W.; CALLISON-BURCH, C. Cost optimization in crowdsourcing translation. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2015)*. Denver, Colorado: [s.n.], 2015. Cited in page 26.

GitHub Inc. *HUBOT Hubot is your friendly robot sidekick. Install him in your company to dramatically improve employee efficiency.* 2019. Disponível em: <hhttps://hubot.github.com/>. Cited in page 23.

Google Inc. *Dialogflow.* 2019. Disponível em: <https://dialogflow.com/>. Cited in page 23.

KON, F. et al. Free and open source software development and research: Opportunities for software engineering. In: IEEE. *2011 25th Brazilian Symposium on Software Engineering.* [S.l.], 2011. p. 82–91. Cited in page 27.

Lacerda, A.; Aguiar, C. Floss faq chatbot project reuse - how to allow nonexperts to develop a chatbot. forthcoming. Cited 3 time in pages 26, 33 e 35.

MILLER, A. H. et al. Parlai: A dialog research software platform. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, abs/1705.06476, 2017. Cited in page 26.

MOSTAçO, G. M.; CAMPOS, L. B.; CUGNASCA, C. E. AgronomoBot: a smart answering Chatbot applied to agricultural sensor networks. p. 14, 2018. Cited in page 19.

NEDER, R. T. A teoria crítica de andrew feenberg: racionalização democrática, poder e tecnologia. *Brasília: Observatório do Movimento pela Tecnologia Social na América Latina/CDS/UnB/Capes*, p. 49–66, 2010. Cited in page 27.

PAETZEL, M. et al. Incremental acquisition and reuse of multimodal affective behaviors in a conversational agent. In: *International Conference on Human-Agent Interaction 2018.* [S.l.: s.n.], 2018. p. 92–100. Cited in page 26.

PAIKARI, E.; HOEK, A. van der. A framework for understanding chatbots and their future. In: *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '18.* Gothenburg, Sweden: ACM Press, 2018. p. 13–16. ISBN 978-1-4503-5725-8. Disponível em: <http://dl.acm.org/citation.cfm?doid=3195836.3195859>. Cited in page 24.

PRIETO-DIAZ, R. Status report: Software reusability. *Software, IEEE*, v. 10, p. 61 – 66, 06 1993. Cited in page 26.

Rasa Technologies GmbH. *RASA Open source tools to build contextual AI assistants.* 2018. Disponível em: <https://rasa.com/>. Cited in page 23.

SCOTT, B. D. Evaluation, development & testing of an educational support chatbot. p. 130, 2018. Cited in page 19.

SHAIKH, A. A Survey On Chatbot Conversational Systems. p. 3, 2016. Cited in page 19.

SHUM, H.-y.; HE, X.-d.; LI, D. From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, v. 19, n. 1, p. 10–26, jan. 2018. ISSN 2095-9184, 2095-9230. Disponível em: <http://link.springer.com/10.1631/FITEE.1700826>. Cited in page 21.

TAVANAPOUR, N.; BITTNER, E. A. C. Automated Facilitation for Idea Platforms: Design and Evaluation of a Chatbot Prototype. p. 9, 2018. Cited in page 26.

TURING, A. Computing machinery and intelligence. n. Mind 49: 433-460, 1950. Cited in page 19.

WEIZENBAUM, J. ELIZA — a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, v. 26, n. 1, p. 23–28, jan. 1983. ISSN 00010782. Disponível em: <http://portal.acm.org/citation.cfm?doid=357980.357991>. Cited in page 19.

WIRTH, R.; HIPP, J. Crisp-dm: Towards a standard process model for data mining. In: CITESEER. *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. [S.l.], 2000. p. 29–39. Cited in page 25.

XOXCO Inc. *Botkit Building Blocks for Building Bots*. 2017. Disponível em: <https://botkit.ai/>. Cited in page 22.