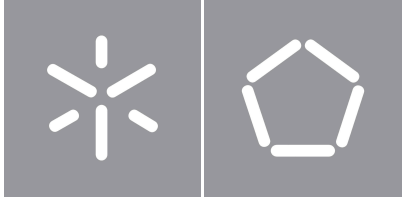
University of Minho
School of Engineering

João Miguel Santos Sá

Multilanguage chatbot with artificial intelligence for client support

october 2023



University of Minho
School of Engineering

João Miguel Santos Sá

Multilanguage chatbot with artificial intelligence for client support

Master's in Informatics Engineering

Natural Language Processing and Artificial Intelligence
Dissertation supervised by
Paulo Jorge Freitas Oliveira Novais
Dalila Alves Durães

october 2023

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

This work has been partially supported by the project “IBPS - Intelligent Batch Processing System” , with the reference POCI-01-0247-FEDER-069998, co-financed by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement.

I would like to express my heartfelt gratitude to my supervisor for all of her assistance throughout the dissertation writing process, and for playing a critical role in the success of this study, from research and analysis of the state of the art to methodology selection, implementation, and completion. I want to offer my heartfelt appreciation to Alexandra for her focused instruction and amazing example.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, october 2023

João Miguel Santos Sá

Resumo

A inteligência artificial surgiu nos anos 50, com a criação de um espaço de estudo com o objetivo principal de desenvolver máquinas inteligentes. A sua evolução foi abrupta e hoje existem mesmo programas que, ao escreverem um conjunto de palavras, conseguem gerar imagens surpreendentes com base no que foi escrito em segundos, algo que um pintor famoso faria, mas que demoraria horas ou mesmo dias a concluir.

Em vários setores empresariais, desde a educação aos cuidados de saúde, entre outros, tem havido um aumento notável do interesse e da utilização de chatbots. Esta tendência crescente abriu caminho para a implementação de chatbots em diversas áreas. Especificamente, estes chatbots servem como intermediários informativos em empresas como a *Retail Consult*, uma empresa de tecnologia especializada em desenvolvimento de software na área do retalho e na utilização de processamento em lote, que envolve o tratamento de grandes volumes de dados.

Além disso, o sistema de chatbot foi concebido para reconhecer e compreender uma multiplicidade de línguas, incluindo o inglês, que será a língua principal do utilizador, bem como o português e o espanhol, assegurando uma comunicação na língua preferida do utilizador. Esta capacidade multilingue é particularmente crucial para a *Retail Consult*, dada a sua presença global com escritórios espalhados por vários países. Sem dúvida, esta versatilidade linguística acrescenta um valor imenso ao produto. Para conseguir este reconhecimento linguístico, foram utilizadas técnicas de inteligência artificial e de processamento da linguagem natural, que foram implementadas através de um quadro bem estruturado concebido para a criação de chatbots e de software similar, incluindo assistentes virtuais.

O desenvolvimento de um serviço com estas características está preparado para aumentar significativamente a produtividade interna da organização. Por exemplo, um analista de sistemas pode perguntar diretamente ao chatbot o que pretende, poupando tempo que, de outra forma, teria sido gasto na procura e consulta de informações à base de dados. Quer se trate de determinar o número de trabalhos em execução ou de identificar os que apresentam falhas, este sistema procura simplificar as operações e melhorar a eficiência dentro da empresa.

Com base no trabalho apresentado na introdução, onde discutimos os papéis cruciais do processamento em lote e da integração do chatbot no ambiente em específico, agora aprofundamos a dinâmica do setor de retalho nesta análise minuciosa. O objetivo está centrado na criação de um chatbot versátil capaz de comunicar com os utilizadores em várias línguas, fornecer respostas precisas e, acima de tudo, auxiliar na tradução de idiomas. Além disso, os resultados confirmam a competência do chatbot na identificação e tradução precisa de idiomas, com base em nossa análise avançada, onde investigamos a importância do processamento em lote e introduzimos a estrutura de IA conversacional RASA.

Esses elementos, contribuem para uma experiência mais envolvente e satisfatória para o utilizador. Ou seja, este trabalho permitiu e ampliou o nosso conhecimento sobre a interação entre processamento em lote, tecnologia de chatbot e comunicação multilíngue no contexto do retalho.

Palavras-Chave Processamento de linguagem natural (PNL), linguagem natural (LN), aprendizagem automática (AA), aprendizagem profunda (AP), inteligência artificial (IA), processamento em Lote (PL)

Abstract

Artificial intelligence emerged in the 1950s with the establishment of a research field aimed at developing intelligent machines as its main objective. Its evolution was abrupt, and today there are even programs that, by writing a set of words, can generate astonishing images based on what has been written in seconds, something that a famous painter would do, but which would take hours or even days to complete.

In various business sectors, from education to healthcare, among others, there has been a notable increase in interest in and use of chatbots. This growing trend has paved the way for the implementation of chatbots in various areas. Specifically, these chatbots serve as information intermediaries in companies such as *Retail Consult*, a technology company specializing in software development in the area of shredding and the use of batch processing, which involves handling large volumes of data.

In addition, the chatbot system has been designed to recognize and understand a multitude of languages, including English, which will be the user's main language, as well as Portuguese and Spanish, ensuring communication in the user's preferred language. This multilingual capability is particularly crucial for *Retail Consult*, given its global presence with offices in several countries. Undoubtedly, this linguistic versatility adds immense value to the product. To achieve this linguistic recognition, artificial intelligence, and natural language processing techniques will be used, which will be implemented through a well-structured framework designed for the creation of chatbots and similar software, including virtual assistants.

The development of a service with these characteristics is set to significantly increase the organization's internal productivity. For example, a systems analyst can ask the chatbot directly what they want, saving time that would otherwise have been spent searching for and consulting information in the database. Whether it's determining the number of jobs in progress or identifying those with faults, this system seeks to simplify operations and improve efficiency within the company.

Building on the work presented in the introduction, where we discussed the crucial roles of batch processing and chatbot integration in the specific environment, we now delve into the dynamics of the retail sector in this in-depth analysis. Our goals are centered on creating a versatile chatbot capable of communicating with users in multiple languages, providing accurate responses, and, above all, assisting

with language translation. In addition, our results confirm the chatbot's competence in identifying and accurately translating languages, based on our advanced analysis, where we investigated the importance of batch processing and introduced the RASA conversational AI framework.

These elements, together, contribute to a more engaging and satisfying user experience. So, the research has broadened our knowledge of the interaction between batch processing, chatbot technology, and multilingual communication in the retail context, which can guide future advances in this field, as emphasized throughout our study.

Keywords Natural Language Processing (NLP), Natural Language (NL), Machine Learning (ML), Deep Learning (DL), Artificial Intelligence (AI), Batch Processing (BP)

Contents

List of Figures

List of Tables

Part I

Introductory material

Chapter 1

Introduction

1.1 Contextualization and Motivation

Retail sale is the sale of products or services to the final consumer, while wholesale is the vending of these to other companies/resellers, who will resell the products to the final consumer. In other words, in retail sales, we can sell several or a few products to several different buyers, while wholesale implies the sale of several products to a single buyer editorial de Conceito.de [2020].

Retail sales can be carried out in physical stores or online, while wholesale sales are usually carried out also by telephone, email, or orders on specialized e-commerce sites. Retail prices are generally higher than original wholesale prices because they include a profit for the reseller editorial de Conceito.de [2020].

The market where the retailer operates and all the business dynamics are important factors that, consequently, lead to several concerns such as the profitability factor, because like any company, retailers want to make profits to keep their business running, that is, they have to control their expenses and maximize their income. Another concern that retailers face is competition and this can come from companies in the same sector or another, as is the case with the online sale of products. The way to mitigate it would be for the retailer to find a way to stand out from the rest, suppliers are also part of this cake, as retailers must have a pleasant, healthy, and lasting relationship with their suppliers, as they depend a lot on these people to obtain their products. Another aspect to note is stock management, which must undoubtedly be managed efficiently to avoid having products that are expiring or not run out of products to sell, finally, and perhaps the most important thing is to have satisfied customers with the shopping experience and the product they purchase, to achieve this, the retailer needs to sell good quality products and offer excellent customer service Alves [2015].

Regarding the business, we imagine this scenario in which we are facing the process of selling a perishable good. Firstly, that has to be grown by a farmer, then the farmer if he wants to, can sell several units of his product, if so, the said goods are transported from the field to a central market and then a

certain amount of them is sold to a small local grocery store which, in turn, sells to the final consumer.

In short, the grocery store functions similarly to the retailer, and the entire business dynamic is undeniably complex, as the grocery store does not only sell perishable goods and it is impossible to manage it without technological solutions, that is, to support retailers in the various aspects of their own business, it is necessary to resource to several computational applications, and this is exactly where *Retail Consult* comes in, offering this type of specialized telecommunications.

Retail Consult is a multinational IT company, which provides strategic solutions such as implementation, development, training, and service support, to help its clients in their specific retail business with a strong focus on the solutions provided by Oracle Retail and related technology, these aforementioned solutions address a broad range of retail business needs, such as planning, omnichannel, supply chain management, merchandising, insights and science RetailConsult [2023].

All of these solutions serve to successfully transform the business from customers to users of the application suite that Oracle Retail provides. *Retail Consult* is one of Oracle's leading software implementation partners, with an unrivaled track record of helping customers.

When it comes to planning, with purchases becoming more and more complex, and customer expectations rising, having the right product is no longer enough. Hyper-personalized, omnichannel thinking is also essential to change, and this is the mindset that Oracle Retail delivers RetailConsult [2023]. Oracle's omnichannel suite enables retailers to provide personalized cross-channel experiences for their customers by seamlessly integrating e-commerce, stores, customer relations, order management, and loss-prevention systems RetailConsult [2023]. The Oracle Retail Insights and Science suite combines artificial intelligence, machine learning, and decision science with data captured from Oracle Retail applications and third-party data. The unique property of these self-learning applications is that they detect trends, learn from results, and increase their accuracy the more they are used RetailConsult [2023]. Concerning merchandising, one of the main challenges in retail is efficiently and effectively providing the right amount of information available in a usable and consumable format. With the explosion of social media, omnichannel shopping, and the pressure to expand into new formats and countries, this has never been more important RetailConsult [2023]. Finally, with an effectively managed supply chain, retailers can reduce inventory and cut operating costs, while increasing sales. Oracle Retail supply chain management solutions empower retailers to plan and execute management strategies RetailConsult [2023].

Retailer businesses are done through the implementation of various applications strategically defined for a given purpose and the insertion of these numerous business areas. The integration of this application is only achieved through batch processing.

Batch processing is a crucial technique employed in handling extensive datasets, necessitating interaction with external systems and internal maintenance Oracle [2017]. It operates autonomously, sparing users from constant monitoring unless an error arises. In the event of an error within a specific job, an administrator is required to manually rectify and rerun the failed program. This entire process is orchestrated by scheduling software, making it fully automated. These automated tasks are commonly referred to as "jobs."

Moreover, during batch processing, applications become temporarily inaccessible to users as they are locked out. However, this temporary inconvenience is balanced by the flexibility of setting batch processing preferences according to specific requirements. This means that administrators can easily identify each job's unique identity, its start and end times, and gain insights into the duration of each task, among other details Oracle [2017]. This approach offers both advantages and disadvantages, which are worth exploring further.

The batch execution circumstances encompass a diverse array of elements within the operational environment of the chatbot. This environment is characterized by a multifaceted composition that incorporates distributed systems, modular components, virtual machines, autonomous agents, central control nodes (referred to as masters), and a multitude of specific tasks, among other components. These aspects play a pivotal role in shaping the operational landscape of the chatbot, and they present numerous benefits and challenges when it comes to its integration and performance.

It is also crucial to have significantly good batch operating and processing, as it is critical in the day-to-day dynamics of the retailer. However, the lot must be managed and monitored with great care and rigor, both on the side of those producing and providing the monitoring process service and the consumer. As a result, it is critical to build a solution, such as a chatbot, that mediators between the human component and batch processing, as well as the IT operations that are being run.

In the future, examples of questions the chatbot could answer would be: "Did that particular job have an error?" or "What is the minimal execution time of the critical path for a specific day?" or "When will the batch end?".

1.2 Objectives

In this section, we articulate the specific objectives of this dissertation. These objectives serve as the foundational framework guiding the research endeavor. They play a crucial role in delineating the direction and purpose of our study. By defining these objectives with precision, we establish a clear path for our research, enabling a focused exploration of chatbots and their applications. Using artificial intelligence technology and natural language processing, the purpose of this project is to develop a chatbot that can detect and recognize multiple languages and respond appropriately to questions in the same language. Customer support will improve and reach new heights with the development of a service that can assist with basic chores.

This chatbot is software that can interact with customers via text messages, primarily in English. Furthermore, we have successfully extended its functionality to support multiple languages, with a particular emphasis on Portuguese and Spanish. This is significant for *Retail Consult* because it is a multinational company, and the incorporation of these languages into the chatbot has greatly benefited our business. Furthermore, we hope to emulate natural human dialogue to better align with everyday communication, bridging the gap between human and machine interaction.

Besides, the product is to help the customer in the communication part, in case he has any doubts, the chatbot must be able to respond within a particular knowledge domain. Summarizing, we can specify what are the main objectives of the chatbot to organize our thought when we pass to the implementation phase. So the chatbot will be able to de Trabalho [2022]:

- Recognize and answer in the language that is being used, such as Portuguese, Spanish, or English;
- Switch language during conversation;
- Engage users in the conversation using natural language processing, close to real assistants;
- Answer users questions and offer information related to the batch processing, providing automatic answers;
- Capable of making corrections, such as terminating a job as desired by the user.

1.3 Research Methodology

The research methodology that we are using is the agile Scrum technique Sachdeva [2016] to create a multifunctional chatbot capable of answering queries and correcting commands in multiple languages. There were numerous steps to the development process Sachdeva [2016].

We began the project with a planning phase in which we defined the chatbot's objectives, primary functionalities, and languages it should support. This stage was critical for creating a firm basis for future growth.

During the chatbot's development, the team used an iterative technique. Our primary focus was on developing capabilities that would allow the chatbot to answer assertively and clearly to users regardless of the language in which they interacted. This included enhancing the chatbot's capacity to interpret user questions and offer correct and helpful responses.

The chatbot's quality maintenance did not involve rigorous testing processes. Instead, it relied solely on internal testing efforts, without incorporating usability studies to assess interface intuitiveness or command correction testing to improve the chatbot's ability to handle complex requests, such as corrections to poorly written instructions.

The progressive inclusion of support for many languages was a key component of the development. This enabled the chatbot to serve a diverse range of consumers in many languages, ensuring an effective multilingual experience.

The Scrum approach was crucial in developing a flexible and responsive chatbot. This was critical to address the needs of consumers in a continually changing environment. Surprisingly, the lack of direct feedback from users throughout development had no detrimental impact on the Scrum methodology's success in providing the anticipated multifunctional chatbot.

1.4 Dissertation Structure

This dissertation digs into a investigation and the creation of a multilingual chatbot with multiple language support to aid in the demonstration of data generated by batch processing.

The dissertation is broken into two main chapters: introduction and dissertation core. In the first main chapter, we provide context and motivation for this program in the Introduction section, where we discuss the objectives, the technique used, and the research methodology. Then in section Background and State-of-the-art, we provide background information and an assessment of the state of the art. We

also describe batch processing in the context of the existing business, examine technology and concept research, conduct a literature assessment, and identify the challenges and problems to be addressed. Finally, we presented some problems and challenge in the section Problem and Challenges.

In the second main chapter it's divide in four section: Contribution, Applications, Evaluations and Results, and Conclusions and Future Work, which highlights the contributions of chatbot development to science, practical applications, and implementation details, including features such as languages and automatic responses, among others. So, in the last section, Conclusions and Future Work, we summarize the key findings and recommend avenues for future research and application enhancement.

Chapter 2

Background and State of the Art

2.1 Enterprise Batch Processing

In the technology area, batch processing is a method where we execute information suited to run programs that are called jobs. This approach is a low-cost solution for processing big volumes of data quickly. After starting the procedure, the computer will only halt if a mistake or irregularity is found and will alert the relevant employees or management. This efficient data processing approach stresses the significance of human-machine interaction, which is a key component in the greater landscape of technological systems.

To delve deeper into the realm of batch processing and its implications, it is essential to establish three fundamental domains: the functional domain, the technological domain, and the human domain. These domains serve as the foundation for comprehending how batch processing systems operate and their impact on human-machine interaction. By dissecting these domains, we gain valuable insights into the intricate relationships between technology, functionality, and the human element, ultimately contributing to a holistic understanding of the role and significance of batch processing in modern computing environments.

The functional domain is introduced into the batch execution process, and it is directly related to the functionalities that are conducted internally and how they are designed and constructed Spring [2023].

This execution process is made up of several modules, each of which operates in a different area of *Retail Consult*'s business context. Each of these modules, which are made up of databases and applications, is assigned a specific agent, and the agent is assigned a specific task or set of tasks with certain dependencies. Finally, there is an agent with a higher hierarchical level than the others, known as the master, who determines the order of execution and job precedence Wikipedia [2023].

The technological domain corresponds to the environment where the batch is processed and, in turn, executed, as we mentioned earlier, an agent has a task and is associated with a module and each agent has a virtual machine assigned to it, which can be shared with another agent and they can run the same

application and also a specific port. As for the size of each machine, it was established according to the number of wires needed by customers Oracle [2017].

Moving on, the master agent gives tasks to the agents, and the execution is done using the FIFO method (first in first out), which means that the first agent to receive the job will be the first to perform it, automatically switching from passive to active mode. Furthermore, each agent must grant itself two crucial components: a maximum load and a weight per thread. If the entire weight of this agent's threads is less than the maximum load, it is an agent to whom you may be assigned new tasks; otherwise, it is not Wikipedia [2023].

As previously said, batch processing requires a significant amount of time to finish since we are working with a big quantity of data. Unless there is an error that requires manual intervention, there is no human participation at this stage. Producing these processing mistakes can occasionally have a substantial negative impact on the business and the client waiting for service.

As for the development, this can have an impact from the point of view of time and resource constraints, because if an error occurs during batch processing, the system will be stopped, which will suffer delays and aggravate the use of resources Wikipedia [2023].

Another critical aspect to emphasize is data integrity. Data damage or loss can result from batch processing mistakes, which can be damaging to the developer and system users. Furthermore, if the system experiences faults during batch processing, the user may have an unsatisfying experience. After all, they may be unable to obtain the information they seek.

Finally, there is the consideration factor, because if errors in the batch cause serious problems in the system, they may negatively impact the developer's and, in some cases, the system's opinion.

In conclusion, to minimize the impacts that errors cause in batch processing, the team or person who developed the system must have carefully designed and tested batch processing systems and implemented robust switches that handle and recover extremely well from mistakes.

On the other hand, if there are errors during batch processing, the customer may also be affected. A potential impact would be the fact that the error itself prevents the necessary data from being processed outside the time requested by the customer, which would affect him, as he wouldn't receive the expected results and could harm his commercial activity or cause other types of disorders.

If the batch processing error leads to incomplete and inaccurate results, the customer will receive this data and, consequently, may lead to incorrect decisions in the business area. Finally, if there is a batch processing error for data loss, the customer may not have access to all the necessary information.

2.2 Technology and Concept

Chatbots and virtual assistants have emerged as revolutionary innovations in the dynamic landscape of the technological age, dramatically altering human-machine interaction. So, in this section, several questions must be answered, such as what virtual assistants are. A virtual assistant (VA) is a piece of software that helps users with a variety of tasks such as scheduling appointments, setting reminders and alarms, creating lists, answering questions, and providing information and directions THIRUPALU and SURESH [2023]. Virtual assistants are designed to use artificial intelligence (AI) and natural language processing (NLP) technologies to understand and answer to user requests Nithuna and Laseena [2020]. Many virtual assistants also can learn and adapt to a user's preferences and habits over time Campagna et al. [2017]. They are growing more popular as a means to simplify daily duties and make it simpler to keep organized and connected.

As we know, there are some virtual assistants profitable in the market, such as Cortana from Microsoft, Alexa implemented by Amazon, Siri by Apple, and finally Home belonging to Google. Nonetheless, the aforementioned virtual assistants have two different ways to interact with the user, they can recognize voice and text Kepuska and Bohouta [2018].

However, we will concentrate on text interaction because the goal of this project is to create a chatbot. And, while virtual assistants are more well-known, chatbots exist and are already available in the market, such as JivoChat, Zendesk, Zenvia, Cliengo, and Intercom, which are also some of the main platforms for implementing chatbots, as reported on the b2b stack Blog Feitoza et al. [2021].

Different from a virtual assistant, as the name implies, a chatbot is a type of natural language processing (NLP) system that is designed to understand and generate human-like text and speech. NLP is a subfield of artificial intelligence (AI) that focuses on the interaction between computers and human (natural) languages. Chatbots use algorithms and machine learning techniques to analyze and interpret user input and generate appropriate responses based on a predefined set of rules or a database of information Nithuna and Laseena [2020].

One of the most difficult aspects of designing chatbots is their natural language processing capabilities, which need a thorough grasp of the structure and meaning of human language, as well as the capacity to provide logical and suitable answers. This necessitates the employment of techniques such as word and phrase parsing, semantic analysis, and context-aware answer production.

To summarize, chatbots are designed to mimic human conversation and provide assistance or information to users through text or voice interactions. It will work as the first line of support for processing in

batch, focusing on answering questions related to the operability and infrastructure part of batch execution.

One critical factor that should be considered is the functions and justifications for implementing a chatbot system. A chatbot, in a technical context, is an automated conversational agent capable of interpreting user inquiries, using established algorithms (which will be elaborated on later), and delivering accurate responses by user expectations. Regarding the last question, the reason for its development is that this is a market that is constantly growing, giving employees time to focus on more important tasks, preventing customers from waiting for responses, and, most importantly, providing the customer with a proactive interaction.

Another essential question to ask is what tools exist to create a chatbot. Fortunately, at this moment all the developers have the lucky to have a variety of solutions, such as RASA, and DeepSeepch by the creators of Mozilla Firefox and OVAL (Stanford Open Virtual Assistant Lab) and they have one thing in common they are all free software tools Nacimiento-García et al. [2020].

Yet to add, we have already defined two concepts that are quite relevant to the problem under study. We may say, in a simple way, that a virtual assistant has other features that the chatbot does not have, which is to help the customer in matters such as scheduling a meeting or notifying the user 15 minutes before the time of the meeting.

Based on our investigation of the chosen implementation technology, we believe that the capacity to communicate URLs over messaging channels has emerged as a key feature. This functionality distinguishes the tool from a standard chatbot and is more akin to a virtual assistant. This claim is supported by empirical data and fits within the larger context of human-computer interaction research.

To summarize, the capacity to provide links via messaging is a critical feature in the context of RASA, the selected framework for implementation. This capacity has been empirically proven and works more like a virtual assistant than a traditional chatbot. Furthermore, I'd like to underline that the RASA framework has a high level of confidence that a smooth transition to a virtual assistant will be achievable in the future. This claim is reinforced by the framework's inherent adaptability and growth, as demonstrated in several scientific studies and practical implementations.

So, as previously stated, RASA is the software we will use in this dissertation. It is a machine-learning framework that aids in the creation of a chatbot and allows us to automate text and voice conversations Nacimiento-García et al. [2020].

We can divide RASA into two global parameters that are:

- Task-oriented - RASA responds to what the user wants;
- Dialogue system - The user will have what he wants simply by talking to an autonomous system

with bidirectional conversation.

For RASA or every task-based dialogue system, the first step, and one of the most important thing to do is understanding the text. If a human being produces something, in this case, in text format to the framework the NLU, i.e. natural language understanding will transmit that information to machine-readable Bocklisch et al. [2017].

They convey data using rule-based mechanisms such as regular expressions, similar to a state machine. To extract specific pieces from text, such as an email address, you only need to be familiar with specific patterns. Their efficiency, however, reduces when confronted with unfamiliar content RASA [2023a]. For this reason, arises the neural methods, which are transform-based model an example of this is DIET (Dual Intent and Entity Transformer), which, sort things that the users say based on example that had been provided, so it need training examples, and the more the better and this one can handle future things. RASA uses both methods, but the neural one is the core of the framework RASA [2021]. Yet to add, the other task is to know what to do next, in other words, what should the assistant say or do next? RASA uses a rule-based that is analogous to building a large dialogue tree in which the discourse follows specified paths based on certain rules or conditions RASA [2021].

For the implementation of the RASA assistant, we will need minimal files and they are the domain and config files. The files belonging to the data we will need are nlu and the goal of Natural Language Understanding (NLU) is to extract structured information from user messages, this usually includes the user's intent and any entities their message contains, the stories that show potential conversational flows, and engagement rules, for example, when someone says goodbye we have to do it too RASA [2023a].

Above we didn't speak about two files. Relatively to the domain file, this is the most important file in the RASA assistant, this file contains everything our assistant knows like Mai and Maxim [2021]:

- Responses - These are usually phrases or words that the assistant can say to the user;
- Intents - These are categories of phrases or words users say;
- Slots - These are variables remembered over the course of a conversation;
- Entities - These are pieces of information extracted from incoming text;
- Actions - These add application logic and extend what your assistant can do.

One of the most important components when it comes to developing a chatbot or virtual assistant is entities. RASA has three methods for extracting entities.

The first method is to use pre-made templates like spaCY to extract entities like names and places, which I will explain deeply.

Regarding spaCY, the Named-entity recognition, also known by the initials NER is a branch of natural language processing (NLP) that is divided into two important stages Audebert et al. [2020]:

- extract and identify entities;
- classify the identified entities.

Named-entity recognition (NER) uses "tokenization", which consists of converting the natural text into small parts, called tokens, and applying techniques for example supervised machine learning-based systems, rules-based systems, dictionary-based systems to identify names, product names, and locations, for example, in some cases, the word starts with a capital letter. To help explain better, in the following images we can see an example using this API of extracting entities from a small text that tells us about Microsoft (Figure 1) Techtarget [2023].

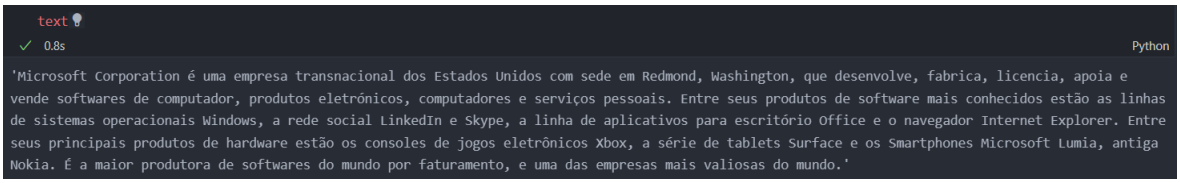


Figure 1: Example of a code snippet.

In this image, we only see a simple text, and after applying natural language processing and choosing the language we want to work and doing the code presented under we have this result (Figure 2):

Figure 2: Extract, identify and classify the entities.

First, we have the entity as we can see, and next, we have the entity type. In other words, Microsoft Corporation is an organization (ORG), United States is a local (LOC), etc Audebert et al. [2020].

After this part is completed, intelligence mechanisms can be applied artificially, because, with NLP implemented, the chatbot will already have the ability to distinguish terms that, in turn, could be identical. For example, if the chatbot to be developed was in the financial sector, without the NLP part, it would have difficulties in realizing that bulletin and invoice refer to the same word.

Another model used is the duckling, but this one focuses more on numerical terms, such as data removal, money, and distances, among others.

The second method used with some weight in this framework is regular expressions. Finally, we have machine learning models, because if there are no pre-made models or patterns for using regular expressions to extract certain entities, this is what we should use. An example of a powerful machine learning model that comes with RASA is the DIET classifier, is a multi-task transformer architecture that handles both intent classification and entity recognition together. It provides the ability to plug and play various pre-trained embeddings like BERT, GloVe, ConveRT, and so on. In our experiments, there isn't a single set of embeddings that is consistently best across different datasets. A modular architecture, therefore, is especially important. RASA [2023a].

It is also important to answer certain questions, because they are required to understand how Rasa works and how you can use it successfully to create a chatbot tailored to the needs of your project. Throughout the development process, from model selection through testing and deployment, they assist you in making educated decisions.

- What models are being trained?
- How does RASA gain knowledge about what the user might say?
- What data do you use for training?
- Is it possible to insert our models?
- How do we test what we develop from our model?

When we start a project, it already has a default model, that is, the knowledge base it presents is a model of RASA itself. As for, the knowledge that RASA has about what the user might say is contained in the nlu file of type YAML. In turn, the data that is used for training is defined by the developer in the stories, nlu, and rules files RASA [2023a].

It is also possible to add specific models if the existing models are not accurate enough to achieve the propose goals, and we can test what we develop from our model using the rasa test command.

The system architecture is comprised of numerous components that collaborate to provide a unified user experience. The user interface, which can be a console or a graphical interface, is the first component and allows the user to communicate with the agent. The user can enter data into the interface, and the agent can respond with output RASA [2023a].

The next component is the Natural Language Understanding (NLU) pipeline, which is responsible for processing user statements. The pipeline is trained using NLU models, which are defined in the yaml config file. These models are used to understand the intent behind the user's statements and extract relevant information RASA [2023a].

The Dialogue Policy component is responsible for deciding the next action in the conversation based on the context. This component takes the information extracted from the NLU pipeline and uses it to determine the best response RASA [2023a].

The Action Server component is used by the agent to request specific actions and receive the corresponding events. This component acts as an intermediary between the agent and the backend services RASA [2023a].

The File System component handles the data models that will be trained, as determined by the development team. This component stores the models and ensures they are accessible to the other components in the system RASA [2023a].

The Tracker Store component briefly stores the content of the dialogue so far, providing a history of the conversation that can be used to inform future decisions. This component is crucial for maintaining the context of the conversation RASA [2023a].

Finally, the Lock Store component ensures that messages for a given identifier are processed in the correct order. This component is used to prevent race conditions and ensure the correct processing of messages RASA [2023a].

In conclusion, the system architecture is designed to provide a smooth and seamless interaction between the user and the agent, and it consists of various components working together to process user input, determine the best response, and provide the user with the information they need (Figure 3) RASA [2023a].

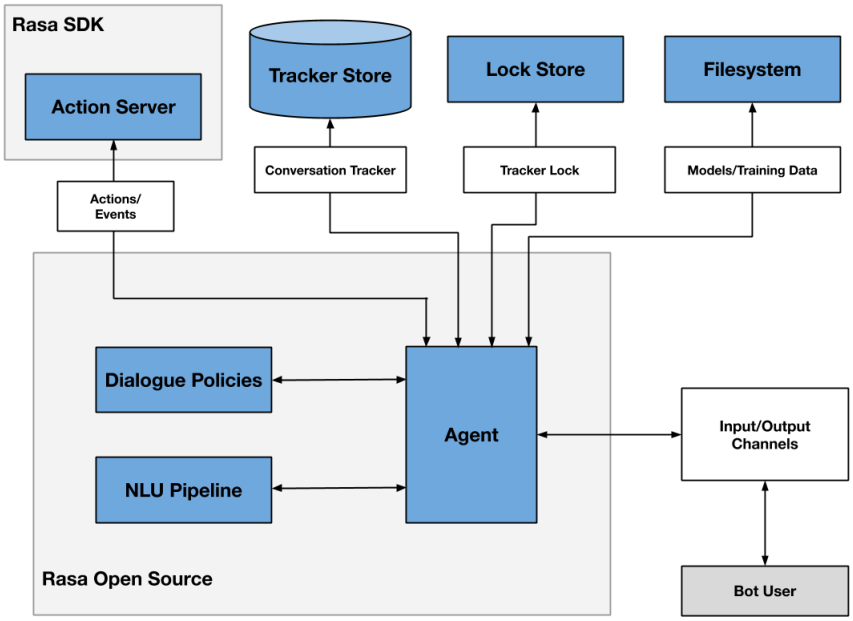


Figure 3: Definition of Chatbot Architecture Nacimiento-García et al. [2020].

2.3 Literature Review

In the previous topic, We spoke about the various virtual assistants and chatbots on the market, indicating those used the most by the public. Nevertheless, we are going to focus on explaining their specifications in detail. The requirements used in this study included whether the tools were open-source, whether they permitted the addition of new knowledge for personal or professional purposes, their features, benefits, and drawbacks, their ability to recognize both entities and various languages, and how they did so, their first interaction with the user, and how they implemented and developed this dialogue. Finally, to fully comprehend everything, we shall address a few crucial queries:

1. What are the common features?

2. What are the differences?

3. What groups do they insert?

First of all, I would like to emphasize that research has been done on the following virtual assistants: Alexa, Google Home, the Facebook Messenger bot, RASA, and Deep Speech.

Relatively to the first topic, what these assistants have in common are entity recognition and multilanguage recognition, and the best one is that they are all open-source, which means, they allow the public to use the source code to study, change, or modify the software without paying.

Regarding multilanguage recognition, we have two important aspects to analyze, the way that they reach multilanguage and how the user speaks in another language to the assistant. Besides, they have these two things in common (entities and multilanguage recognition), but they have specific approaches to them. Yet to add, all of the virtual assistants will fail unless Apple allows a new knowledge base, permits us to train our models, and allows us to use them as we want and need.

In this examination, we will give many virtual assistant comparison tables that address crucial characteristics and qualities to aid readers in selecting the best virtual assistant for their requirements. The ease of use of artificial intelligence has been linked to our daily encounters with technology via virtual assistants, which have evolved to become a vital part of our lives.

However, it is crucial to note that not all tables may have the data needed throughout the search. This is a result of the field of artificial intelligence and virtual assistants constantly growing, as well as the accessibility and availability of data at the time of data gathering.

These tables are meant to give a thorough overview of each virtual assistant, enabling a comparison of its features and functionalities. With this knowledge, readers will be better equipped to select a virtual assistant that perfectly matches the demands of their projects or applications.

To better understand the benefits and drawbacks of each virtual assistant, as well as the libraries utilized in their development, we give comparative tables of the virtual assistants examined.

We have the "Mozilla Deep Speech" voice recognition system compared in the table. It is Python-based open-source software created by Mozilla Firefox. The technology is renowned for text-to-audio conversion and for supporting entity and multilingual recognition. Good voice recognition performance and an emphasis on privacy are some benefits. The difficulty in interpreting spoken words, however, is a drawback. Through transfer learning, the system also enables the development of new knowledge bases Nacimiento-García et al. [2020](Table 1).

Table 1: Mozilla Deep Speech.

Comparison Criteria		**Description**
Name		Mozilla Deep Speech Nacimiento-García et al. [2020]
Company / open source		Mozilla Firefox / Yes relased under MPL (Mozilla Public License)
Technology		Python
Features	Multilanguage recognition	Yes
	Entities recognition	Yes
	Interaction virtual assistant?	A stream of audio is inputted into DeepSpeech, which turns the stream of audio into a series of characters in the chosen alphabet.
Application Field/Knowledge base		General Settings (but this one is used to convert speech to text)
Approach for multilanguage recognition		You cannot use DeepSpeech because it does not detect the language (de/en/...) Your audio should be transcribed to see what has high confidence values.
allow building new knowledge base		Yes, you can remove specific layers from a pre-trained model and initialize new layers for your target data using DeepSpeech's transfer-learning method, which also incorporates pre-trained data sets. But we can also include new sets)
Advantages		Excellent speech-to-text results, good privacy
Disadvantage or limitations		Voice words recognition difficult to understand

Comparison Criteria	Description
Libraries used	They employ a model developed using machine learning methods and based on the Deep Speech research paper from Baidu.

An overview of Amazon's virtual assistant Alexa is provided in the Table 2. Alexa is a potent tool that offers recognition of over 50 languages and entities and is based on technologies similar to those shown. By using the word "Alexa" to activate the assistant, Users can communicate with it naturally Hoy [2018]. Alexa's multilingual capabilities are made possible by the self-supervised learning approach, which trains the models using massive monolingual datasets or unlabeled text. Entity resolution also guarantees that user requests are correctly interpreted Hoy [2018].

Alexa offers simplicity of use, online shopping, music playback, alarms, reminders, and more, highlighting benefits and features. It is crucial to note the system's drawbacks, such as potential recognition failures, delayed answers, and disconnections. The chart also reveals that using Q&A and information from the company, a new knowledge base may be created Hoy [2018].

In conclusion, the table provides a clear and detailed review of the Alexa virtual assistant and its key characteristics (Table 2).

Table 2: Virtual Assistant Alexa.

Comparison Criteria		Description
Name		Alexa Hoy [2018], Kepuska and Bohouta [2018]
Company / open source		Amazon / Yes
Technology		Javascript, java, Python, C#, Go, Ruby, or PowerShell
Features	Multilanguage recognition	Yes (50+ languages)
	Entities recognition	Yes
	Interaction virtual assistant?	The words and phrases that users can use to direct Alexa to perform a task are defined by the voice interaction model.
	First interaction	User says an awanke word - Alexa
Application Field/Knowledge base		General Settings
Approach for multilanguage recognition		Amazon used a self-supervised learning pretraining strategy, in which the models are trained on sizable, monolingual datasets or unlabeled texts, to give Alexa multilingual capabilities.

Comparison Criteria	Description
Approach for entities recognition	The user's request for a slot value is resolved by Alexa using entity resolution for a single, recognized entity.
Approach for interaction	Relatively to voice recognize Alexa uses ASR (automatic speech recognition)
Allow building new knowledge base	Yes (Q&A on your organization's data)
Advantages	Is a tool that is simple to use, capable of online shopping, and equipped with nonstop music, timers, alarms, and reminders. MyAyan [2022]
Disadvantage or limitations	mishearing, slow response and bloatware, connection drops, stop responding MyAyan [2022]
Libraries used	Alexa Skills Kits provides a library of built-in intents

In comparison to Google Home, a Virtual Assistant was created by Google that enables entity identification, voice interactions, support for many languages, a general knowledge base, and the capacity to build new intelligence clusters. While it has numerous benefits, like syncing and frequent updates, there are also drawbacks, such as the voice quality, which isn't fully natural, and privacy issues PeopleBank [2018] (Table 3).

Table 3: Google Assistant.

Comparison Criteria		Description
Name		Google Assistant/Google Home Hoy [2018], Kepuska and Bohouta [2018]
Company / open source		Google / Yes
Technology		Javascript, go, c++, java
Features	Multilanguage recognition	Yes (44 languages)
	Entities recognition	Yes
	Interaction virtual assistant?	When you ask for assistance, Google Assistant can use information from your Google Account to provide you with what you need
	First interaction	Say "Hey Google" and ask your question or give a command
Application Field/Knowledge base		General Settings
Approach for multilanguage recognition		We need to change the settings

Comparison Criteria	Description
Approach for entities recognition	The entity type for each intent parameter specifies how data from an end-user expression should be extracted
Allow building new knowledge base	Yes (New Intelligence Clusters let you build devices that respond to whole-home context such as presence, for more helpful experiences)
Advantages	Synchronization, upgrade day by day, has many functionalities PeopleBank [2018]
Disadvantage or limitations	No natural voice, privacy concern PeopleBank [2018]
Libraries used	Google Assistant library for Python, supported by Raspberry Pi 3

A comparison of Cortana's Virtual Assistant is shown in the table. To comprehend its features and functionalities, it highlights pertinent comparison criteria.

The application for the Cortana virtual assistant focuses on general settings and supports the creation of additional knowledge bases. However, the settings for multilingual support must be changed. Several customer support functions, including meeting assistance and calendar management, are utilized by the assistant Microsoft [2022].

Cortana has a significant drawback in that it may be susceptible to malware infection, jeopardizing its security.

Consequently, it enables a more unbiased study of its strengths and weaknesses for the objectives for which it is intended (Table 4).

Table 4: Virtual Assistant Cortana.

Comparison Criteria		Description
Name		Cortana Hoy [2018], Kepuska and Bohouta [2018]
Company / open source		Microsoft / Yes
Technology		C++
Features	Multilanguage recognition	Yes (19 languages)
	Entities recognition	Yes
	Interaction virtual assistant?	EI - emotional intelligence

Comparison Criteria		Description
	First interaction	When we launch Cortana, the following message appears: Type a message for me in the text box or click the microphone icon to speak to me.
Application Field/Knowledge base		General Settings
Approach for multilanguage recognition		We need to change the settings.
Approach for entities recognition		NLP, DL, computer vision
Approach for interaction		Cortana's exploration of emotional intelligence (EI) is based on computer vision and machine learning (ML) methods.
Allow building new knowledge base		In the most recent version of Cortana for Windows, users can search for documents and quickly compose emails.
Advantages		Various customer support functionalities (e.g. meetings, calendar assistance, etc.) Microsoft [2022]
Disadvantage or limitations		Cortana can be tricked into installing malware PandaSecurity [2022]
Libraries used		Cortana Skills Kit

A comparison of virtual assistants is presented in the table, with a special emphasis on Apple's Siri, its assistant. It includes crucial details about Siri, including its underlying technology, attributes, domain of use, and strategies for entity identification and language recognition Reuters [2022].

The popular and frequently used virtual assistant Siri was created by Apple and is optimized for iOS devices. With a Python technical foundation, Siri offers multilingual recognition, enabling users to communicate with the assistant in a variety of languages. Additionally, it contains entity identification capabilities that enable a deeper comprehension of user commands and inquiries.

The assistant operates through voice interaction, with Siri's servers handling command processing. Siri does have significant limits, such as privacy concerns and difficulties interacting with kids while being widely used and acclaimed for its usability and communication abilities.

In summary, the table provides a summary of the key characteristics, benefits, and drawbacks of Siri, enabling a comparison between this personal assistant and other products on the market (Table 5).

Table 5: Virtual Assistant Siri.

Comparison Criteria		**Description**
Name		Siri Hoy [2018], Kepuska and Bohouta [2018]
Company / open source		Apple / Yes (has a version that is open source)
Technology		Python
Features	Multilanguage recognition	Yes
	Entities recognition	Yes
	Interaction virtual assistant?	Siri servers receive and process your speech inputs. In every situation, Apple will get transcripts of your communications in order to fulfill your requests
	First interaction	Siri speaks back to you when you ask her a question. After saying "Hey Siri," ask a question or make a request
Application Field/Knowledge base		General Settings: 98 percent of iPhone owners have at least once used this virtual assistant
Approach for multilanguage recognition		To give the computer a perfect depiction of the spoken text to learn, humans read passages in a variety of accents and dialects. They record a variety of noises as well. They then create a language model to forecast word sequences from there
Approach for entities recognition		Apple uses huge, datasets to provide Siri with an effective model of speech recognition which is then trained on varying datasets that are made up of voice samplings from lots of people, which allows Siri to recognize all sorts of accents, inflections, and pace of speech
Allow building new knowledge base		No (Siri Data and your requests are not used to build a marketing profile)
Advantages		Very easy to use, good in comunication Medium [2022]
Disadvantage or limitations		don't work well with children, privacy issues, lack of function Medium [2022]

The table provides a comparison of many features of the Facebook company's "Facebook Messenger bot," which is housed inside the BlenderBot open-source project. Erlang is used for the chat component of this bot's front, whereas Java and C++ are used elsewhere. The bot recognizes entities and offers recogni-

tion in several languages. It also enables the creation of a customized knowledge base. Greater consumer interaction and simplicity in scaling are two of its benefits. Despite its skills, it's crucial to remember that this bot cannot completely replace human engagement in encounters Smutny and Schreiberova [2020]. The table's content describes the methods for entity and entity-entity interaction, multilingual recognition, and messaging, as well as the libraries that are used, like the "messenger-bot-library" (Table 6).

Table 6: Facebook Messenger bot.

Comparison Criteria		**Description**
Name		Facebook Messenger bot Smutny and Schreiberova [2020]
Company / open source		Facebook / Yes (BlenderBot)
Technology		Php for frontend, erlang used for chat, java and c++ in other places
Features	Multilanguage recognition	Yes
	Entities recognition	Yes
	Interaction virtual assistant?	Uses the same things as RASA
	First interaction	A Messenger session is automatically started when someone clicks the "Message" button on your Facebook page (or website), allowing them to type a query and start a conversation with your bot
Application Field/Knowledge base		Education Chatbots
Approach for multilanguage recognition		We need to change on the settings
Approach for entities recognition		As the parameter entities on yaml files
Approach for interaction		Use the same things of RASA
Allow building new knowledge base		Yes
Advantages		Higher Customer Engagement, easy scalability Martech [2022]
Disadvantage or limitations		It won't replace human touch Martech [2022]
Libraries used		messenger-bot-library is a python library for facebook messenger bot

The RASA framework, an open-source platform designed to produce chatbots and virtual assistants with advanced natural language understanding skills, is described in depth in the table. Python and NLU (Natural Language Understanding) are the foundations of RASA, which enables programmers to build interactive and customized conversational systems RASA [2023a].

The comparison criteria are first presented in the table, which is then followed by details regarding RASA. It gives a comprehensive summary of the most crucial elements, including the name of the company

(or whether it is open source), the technology employed, and the application domain of RASA, which includes programmers, conversational teams, and businesses in general.

The table includes a list of some of the libraries used by RASA, including SpaCY and TensorFlow, to help readers better understand the framework. In summary, this table is a helpful resource for presenting, contrasting, and comprehending the key aspects of RASA, assisting teams and developers in selecting the best platform for their virtual assistant and chatbot projects (Table 7).

Table 7: RASA framework.

Comparison Criteria		**Description**
Name		RASA Singh et al. [2019], Bocklisch et al. [2017]
Company / open source		RASA / Yes
Technology		python and NLU (Natural Language Understanding)
Features	Multilanguage recognition	Yes (it is possible to implement a chatbot that allows multilingualism
	Entities recognition	Yes
	Interaction virtual assistant?	RASA uses intents and responses
Application Field/Knowledge base		Developers, conversational teams and enterprises
Approach for multilanguage recognition		We can train X models, one for each language, and then load X agents, selecting which agent should receive the message using a language detector
Approach for entities recognition		RASA uses spaCY for names, places, and organizations related to this issue. This system employs ducks for numerical capture, and if that isn't enough, we can also use regex or neural approaches.
Approach for interaction		RASA communicates with the user through actions, as well as these units' intentions (which are statements that the user may make to the assistant) and answers (which are groups of things that the assistant may make)
Advantages		Easy to integrate, interactive learning RASA [2023a]
Disadvantage or limitations		Anyone interested in using RASA to develop a virtual assistant should be familiar with chatbots and NLP (natural language processing) RASA [2023a]
Libraries used		SpaCY, TensorFlow, etc

At this point, we just need to answer the question related to the groups, that is, I think we can divide the research into three large groups, and these are virtual assistants for commerce (Alexa, Siri, Google Home), and research (RASA, Deep Speech). However, Cortana and the Facebook Messenger bot do not have a commercial version, but they also serve to help the user, they must be inserted in a different group.

This information is all part of a single table that was carried out during the research on the topic of a chatbot and virtual assistant, to acquire more knowledge both for writing the document and also to assist in the implementation of the chatbot which, as I said earlier, will be capable to transit to a virtual assistant if its that the need and can further help a person using the product. Below, we can check information about the title of the papers, focus, context, data they used, the methods and their evaluation, and conclusion.

The project "Matilda - multi-language interactive lightweight dialogue annotator" is summarized in the table. It aims to create a simple interactive platform for dialogue annotation that could be used in automated systems like chatbots. Matilda allows collaboration between annotators and supports annotation in a variety of languages. It employs Natural Language Processing (NLP) strategies and concentrates on evaluating your organizational and note-taking abilities. The tool aims to be user-friendly and is perfect for intricate annotations in dialogs with multiple languages (Table 8) Cucurnia et al. [2021].

Table 8: Matilda-multi-annotator multi-language interactive light-weight dialogue annotator.

Comparison Criteria	**Description**
Title	Matilda-multi-annotator multi-language interactive light-weight dialogue annotator Cucurnia et al. [2021]
Focus	Matilda explanation and development
Context	Chatbots, Matilda classified as a dialogue system
Data	They have a server for databases, and they work with MongoDB, which is not relational-oriented for documents
Methods	NLP (Natural Language Processing)
Evaluation Methods	Assess Matilda's administrative skills and note-taking ability, i.e. performance
Conclusion	Matilda was developed to be user-friendly, allowing multiple languages and multiple annotations.

The table that is shown compares various criteria for a chatbot created to answer frequently asked questions at a university. The chatbot was created using the AIML markup language to interact with

the FAQ database. To give future answers that are more precise and concise, the authors also plan to investigate the use of the latent semantic analysis (LSA) method. AIML, LSA, pattern matching, chatbots, and human-computer interaction (HCI) are some of the key terms. These criteria are crucial for evaluating the efficiency and value of the chatbot in responding appropriately to users' inquiries (Table 9) Ranoliya et al. [2017].

Table 9: Chatbot for University Related FAQs.

Comparison Criteria	**Description**
Title	Chatbot for University Related FAQs Ranoliya et al. [2017]
Focus	Database FAQs chatbots
Context	chatbot in the context of the FAQ database
Data	Database FAQs (frequently asked questions)
Methods	AIML, LSA
Conclusion	Have developed a chatbot using AIML and intend to use LSA in the future, which they believe will have more correct and direct answers
Keywords	AIML (Artificial Intelligence Markup lLanguage), LSA (Latent Semantic Analist), Pattern Matching, chatbot, HCI (Human Computer Interaction)

The table below provides a summary of the study "Leyzer: A Dataset for Multilingual Virtual Assistants." Sowański and Janicki [2020]. The Leyzer dataset and its importance in the creation of chatbot tools are at the center of the paper.

The Leyzer dataset is intended to include a wealth of relevant data required for building effective chatbots. It is intended to be a guide for chatbot implementation rather than concentrating solely on databases. The authors developed this dataset by combining methods from artificial intelligence (AI) and natural language processing (NLP). This made sure the dataset complied with the specifications for multilingual virtual assistants. Using the F1 score for accuracy assessment, the dataset's performance was assessed. The single-BERT, multi-BERT, train-on-target, and zero-shot models were the four types of models tested.

The implementation of the Leyzer dataset proved successful in supporting virtual assistants, bringing the research to a successful conclusion. The authors also state that they intend to add more languages to the dataset, demonstrating its adaptability and potential for future growth. Positive outcomes from the model evaluation further supported the usefulness of the dataset (Table 10) Sowański and Janicki [2020].

Table 10: Leyzer: A Dataset for Multilingual Virtual Assistants.

Comparison Criteria	**Description**
Title	Leyzer: A Dataset for Multilingual Virtual Assistants Sowański and Janicki [2020]
Focus	Leyzer dataset, chatbot
Context	Leyzer is a dataset that contains a lot of relevant information for the development of a tool like a chatbot
Data	Did not focus on databases; they tried to create the most complete dataset to assist the implementation of chatbots
Methods	They had to consult methods based on NLP (natural language processing) and AI
Evaluation Methods	F1-score for accuracy used 4 types of models for testing: single-BERT, multi-BERT, train-on-target, and zero-shot.
Conclusion	Success in implementing a dataset for virtual assistants; they intend to extend the dataset to more languages; the result of the tests on the models is also quite positive
Keywords	Virtual assistant, natural language understanding in a multilingual context, machine translation, corpus text

A research paper titled "A Multi-Language and Extensible Smart Virtual Assistant" Atzeni and Atzori [2019] is summarized in the table that is shown. In particular, a system that can extract user input in natural language and execute it as Java source code is the subject of the paper's discussion of the development of intelligent chatbots and virtual assistants. Two systems used for question answering and semantic analysis are WolframAlpha and the AskCO dataset, and this study compares their performance.

The evaluation used 120 questions, and the findings revealed that AskCO outperformed WolframAlpha in every area, including the handling of questions, providing accurate responses, and overall accuracy. Question-answering, semantic analysis, natural language understanding, and web semantics were among the keywords used in the study.

This paper provides important insights into the field of natural language processing and AI-driven applications, shedding light on the developments in building flexible and language-independent virtual assistants (Table 11) Atzeni and Atzori [2019].

Table 11: A Multi-Language and Extensible Smart Virtual Assistant.

Comparison Criteria	**Description**
Title	A Multi-Language and Extensible Smart Virtual Assistant Atzeni and Atzori [2019]
Focus	Creating Virtual Assistants and Intelligent Chatbots
Context	A system that extracts what the user said (in natural language) and executes that information in Java in source code
Data	Comparison between the AskCO dataset and WolframAlpha
Methods	AskCO
Evaluation Methods	120 questions were evaluated between AskCo and Wolfram: which questions were processed, correct answers, accuracy in particular for each question, and more generally
Conclusion	AskCO had better results in all fields
Keywords	Question answering, semantic analysis, natural language understanding, and web semantics

A comparison of the study "Analysis of User Interaction with Service-Oriented Chatbot Systems" Jenkins et al. [2007] can be seen in the table below. The research focuses on chatbots created under the KIA (Knowledge Interaction Agent) moniker.

The study looked at how people interacted with various chatbots, paying attention to things like the language used, responses, questions and replies, interaction style, and methods of giving users feedback, among other things.

The evaluation techniques utilized AIML (Artificial Intelligence Markup Language), which was also a part of the ALICE system.

The findings underline users' expectations that chatbots will behave and converse like people (Table 12) Jenkins et al. [2007].

Table 12: Analysis of User Interaction with Service Oriented Chatbot Systems.

Comparison Criteria	**Description**
Title	Analysis of User Interaction with Service Oriented Chatbot Systems Jenkins et al. [2007]
Focus	Chatbot

Comparison Criteria	Description
Context	Chatbots developed under the name KIA (Knowledge Interaction Agent)
Methods	Used the same method as in ALICE; they also used AIML
Evaluation Methods	Evaluated: user language, user reaction, questions and answers, style of interaction, and ways of giving feedback to the user, among others.
Conclusion	Users expect the chatbot to behave and communicate like a human being
Keywords	Human-computer interaction, chatbot, question-answering, communication, intelligent system, natural language, dialogue

In this table, a detailed comparison of the Lima et al. study "Real-time Big Data Warehousing: from data collection to data processing" is shown. The goal of the study is to comprehend the role that various technologies play in the implementation of Big Data Warehouses, with a focus on processing capabilities and real-time access to data Lima [2017].

The technologies examined include Kafka for data collection, Spark Streaming for real-time processing, Hive or Cassandra for data storage, and Presto for effective consultations. The evaluation methods focus on the performance of sending messages to Spark Streaming while also discussing the limitations of using technologies from the Hadoop ecosystem and Cassandra's data store.

The findings demonstrate the viability of a real-time Big Data Warehouse that enables analytics with valuable information, real-time data processing, and cross-referencing of current and historical data. To support strategic decisions and keep track of current business data, these features offer useful information (Table 13) Lima [2017].

Table 13: Real-time Big Data Warehousing: from data collection to data processing.

Comparison Criteria	Description
Title	Real-time Big Data Warehousing: from data collection to data processing Lima [2017]
Focus	Understand the role of different components and technologies in the realization of Big Data warehouses understand the role of different components and technologies in the realization of Big Data warehouses

Comparison Criteria	Description
Context	Big Data Warehouses increase the prospects of obtaining data quickly and up-to-date, enhancing access to data in real time. This dissertation also allows us to analyze how several technologies behave, such as Kafka (used for data collection), Spark Streaming (used for data processing), Hive or Cassandra (for data storage), and Presto (for queries).
Data	For the document search, reference databases were defined, namely Scopus, Web of Science, Google Scholar, IEEE Xplore Springer, and RepositóriUM, in which a set of search keywords were used.
Methods	Kafka, Spark Streaming, Hive or Cassandra, Presto
Evaluation Methods	Kafka ensures that messages sent to Sparks Streaming are received as quickly as possible
Issues and Limitations	The technologies of the Hadoop ecosystem and the Cassandra database present limitations in cases where rapid responses are required.
Conclusion	In this dissertation, it was possible to concretize a Big Data Warehouse in real-time, allowing analysis with useful information, real-time data processing, and the crossing of real-time data with historical data, so that decisions are made with knowledge of data patterns and updated business information.
Keywords	Business intelligence, big data, big data warehouse

We present a systematic review of the literature titled "Chatbots in education: a systematic literature review" Kuyven et al. [2018] in the table below. The study investigates the use of chatbots in education with a focus on natural language understanding. The study examined various academic journals and identified articles that addressed the subject in various contexts as well as its use in communication sciences. Artificial Intelligence Markup Language (AIML) and Natural Language Processing (NLP) are the main techniques used to create these chatbots. Machine learning (ML) and deep learning (DL) are the two most common evaluation methodologies.

The development of educational chatbots does, however, come with difficulties, including the need for a strong knowledge base for fruitful conversations and the complexity of the dialogue flow with students. The conclusion emphasizes the potential benefits of combining machine learning and natural language processing, particularly the use of deep learning, to improve the efficacy of these chatbots. The keywords for this review are "education," "artificial intelligence," "conversational agents," and "chatbots" (Table 14)

Kuyven et al. [2018]

Table 14: Chatbots in education: a systematic literature review.

Comparison Criteria	**Description**
Title	Chatbots in education: a systematic literature review Kuyven et al. [2018]
Focus	Natural language recognition
Context	Chatbots are a system that is increasingly found in the health area, and the content of the article focuses on education. However, the area where it appears the most is in the field of communication sciences.
Data	The six databases that gave rise to the publications extracted in the RSL, in descending order, were: Computers & Education(5 -31.3%); Revista Novas Tecnologias na Educação-RENOTE (4 -25.0%); IEEE Transactions on Learning Technologies (2 -12.5%); Brazilian Symposium of Informatics in Education-SBIE(2 -12.5%); International Congress of Educational Informatics -TISE (2 -12.5%); Latin American Journal of Educational Technology -RELATEC-(1 -6.2%).
Methods	AIML (Artificial Intelligence Markup Language); PLN (Natural Language Processing, e.g., pattern matching, Hash tables for keywords).
Evaluation Methods	ML (Machine Learning); DL (Deep Learning);
Issues and Limitations	The need for a considerable knowledge base for a satisfactory conversation; the greater complexity and unpredictability of the dialog flow between the agent and the learner compared to a casual conversation
Conclusion	The machine learning and natural language processing parts show the best results, as does the use of deep learning, which is a type of machine learning, for a deeper search.
Keywords	Chatbots, conversational agents, artificial intelligence, and education

A summary of the research paper "A Classification Model for Named Entity Recognition" Silva [2020] can be found in the table below. The work focuses on employing word embeddings and vector representations to identify named items using contextual distribution and linguistic features. Reputable sources in the fields of educational technology and human-computer interaction provided the data for the evaluation.

BERT (Bidirectional Encoder Representations from Transformers) and BiLSTM (Bidirectional Long-Term Memory), two well-liked neural network designs for named entity recognition, are the techniques employed in the study. As an example, the term "Castelo Branco" can be both a place and a name,

making its classification difficult.

The study's conclusion emphasizes that results were mostly produced through neural networks and that further development and improvement of this area of study is possible (Table 15) Silva [2020].

Table 15: A classification model for the NER.

Comparison Criteria	**Description**
Title	A classification model for the Recognition of Named Entities Silva [2020]
Focus	Exploitation of traces based on the contextual distribution of entities through word embeddings and vector representations associated with linguistic traces
Context	The exploitation of traces based on the contextual distribution of entities through word embeddings and vector representations associated with linguistic traces
Data	Database used: ACM Transactions on Computer-Human Interaction (TOCHI); Congresso Internacional de Informática Educativa (TISE) ; Computers & Education; Creative Education; IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, entre outras
Methods	The use of two neural network architectures such as BERT and LSTM (bidirectional neural networks)
Evaluation Methods	BERT (Bidirectional Encoder Representations from Transformers) and BiLSTM (Bidirectional long short term memory)
Issues and Limitations	Difficulties in the classification of entities hampered the evaluation of the models (e.g. "Castelo Branco" can be both a place and name)
Conclusion	Not talking directly about technologies to be used but about techniques. The results obtained were almost all obtained through the use of neural networks, and research with this type of technique is always open to improvement.
Keywords	Representation of a given language, neural networks, named entity recognition, embedded words

This table provides an outline of the paper "An Overview on Chatbots Technology" Adamopoulou and Moussiades [2020]. The study focuses on the context of virtual assistants as it examines the history, motivation, design, classification, and architecture of chatbots. Chatbots employ knowledge bases to respond to different user inquiries.

The article outlines three main approaches for putting chatbots into practice: the rule-based model, which employs a set of predetermined rules to determine responses; the retrieval-based model, which is

more adaptable and uses APIs to retrieve data; and the generation-based model, which is more sophisticated, employs deep learning, and can deliver better results based on prior user messages.

The article mentions RASA, an open-source platform, as well as closed platforms from Google and IBM that are made available for enterprise use.

Finally, the paper covers key ideas in artificial intelligence, machine learning, and NLU (natural language understanding) to help readers comprehend the fundamentals of chatbot technology. Chatbot, chatbot architecture, artificial intelligence, machine learning, and NLU are all pertinent keywords (Table 16) Adamopoulou and Moussiades [2020].

Table 16: An Overview of Chatbot Technology.

Comparison Criteria	**Description**
Title	An Overview of Chatbot Technology Adamopoulou and Moussiades [2020]
Focus	History, motivation, design, classification, chatbot architecture
Context	Knowledge about virtual assistants
Data	Databases allow the chatbot to answer more types of questions from users (e.g. the knowledge base of the chatbot)
Methods	Rule-based model (most common; choose the system response based on a predefined set of rules such as lexical form recognition); retrieval-based model (less common, more flexible; query and analyze available resources using APIs); and generative-based model (chatbots closer to humans; use deep learning; harder to implement; better than the other two based on users' old and current messages).
Evaluation Methods	RASA (open source), Google e IBM (closed plattforms, offered by enterprise)
Conclusion	Demonstrates information needed to understand the basic principles of a chatbot
Keywords	Chatbot, chatbot architecture, artificial intelligence, machine learning, NLU (natural language understanding)

Chapter 3

Problem and Challenges

Developing a chatbot in a business context, especially in the area of batch processing, presents several complex challenges that require creative approaches and innovative solutions. In this chapter, we will look at some of the main problems and challenges faced during the course of this project, as well as the strategies adopted to overcome them.

One of the most significant challenges encountered was ensuring that the chatbot was globally accessible in the business environment. This accessibility required support for several languages, considering the linguistic diversity that can be found in companies with international operations. The solution adopted to solve this problem was to create a robust and versatile translation system. This system combined Deep Translator, a flexible, free, and unlimited Python tool designed to efficiently translate between different languages using a variety of Organization [2023] translators, with Spacy's core language identification models. Spacy, an open-source library for Natural Language Processing in Python, offers advanced features such as named entity recognition (NER), part-of-speech (POS) tagging, dependency parsing, word vectors, and much more SpaCy [2023]. This combination allowed the chatbot to be effective in different languages and to communicate effectively with a wide range of international audiences.

Another critical challenge we faced was integrating RASA with external databases, which proved crucial in the context of batch processing. To ensure that the chatbot provided relevant and accurate responses to user requests, it was essential to access real-time information from databases such as IBPS and IPE. To achieve this integration, custom actions were defined in RASA, allowing the chatbot to access these databases efficiently and securely. In addition, SQL queries were implemented in Oracle Retail SQL Developer to extract the necessary data, ensuring that the chatbot had access to up-to-date and accurate information. This close integration with the external databases was crucial to the chatbot's effectiveness, as it allowed it to provide real-time information and high-quality contextual responses to users.

In summary, developing a chatbot in a business context with an emphasis on batch processing is a challenging task that involves overcoming obstacles related to global accessibility and effective integration

with external databases. The combination of advanced technologies such as Deep Translator and Spacy, along with the implementation of customized actions and SQL queries, made it possible to create an efficient and highly functional chatbot that met the specific needs of this business project.

Part II

Core of the Dissertation

Chapter 4

Contribution

This dissertation describes the creation and application of a cutting-edge chatbot that makes use of the RASA framework and aims to provide users who are fluent in English, Portuguese, and Spanish with a warm and welcoming interaction experience. The chatbot can access relevant data stored in a database and respond to queries contextually. It also stands out since it exhibits adaptive and continuous learning behaviour by being able to make wise corrections and skip particular steps during the interaction.

The research presented in this dissertation makes a substantial contribution to artificial intelligence, especially in the creation of sophisticated and engaging chatbots. The following is an explanation of the major contributions:

The design and implementation of a chatbot with a friendly and positive focus on user interaction is one of the key accomplishments of this dissertation. Advanced natural language processing (NLP) and a focus on sentiment analysis methods were used to accomplish this goal. The chatbot is careful to ask sometimes how the user is and can modify its responses to sound human depending on whether the user says he is good or sad, so he tries to understand the user's emotions. This strategy tries to enhance the user's interaction experience, boosting satisfaction and system receptivity.

The proposal of an effective method for integrating the chatbot with a database containing pertinent data is another significant contribution of the disseration. The chatbot uses sophisticated data querying and structuring techniques to deliver precise and current responses to users' inquiries. This capacity is especially useful in settings where timely, accurate information is required. The chatbot can assist a range of useful applications, from customer care to the distribution of knowledge, by optimizing this data access.

One of the most creative contributions of this research is the use of intelligent corrections. The chatbot exhibits higher flexibility and adaptability during the engagement process by making intelligent modifications like skipping particular processes, depending on the user input, making it a more dependable and versatile solution.

And finally, the chatbot's support for the three major languages of English, Portuguese, and Spanish

makes a substantial contribution to cross-cultural communication and worldwide user service. Future research on artificial intelligence systems with multilingual assistance can use the strategy used to assure multilingualism's effectiveness as a guide. The chatbot's ability to operate in a variety of languages, which enables it to effectively connect with a wide range of consumers globally, significantly expands its reach and utility.

In conclusion, the contributions of this dissertation represent advancements in the field of chatbot artificial intelligence. The research opens up new possibilities for the use of chatbots in various sectors, from customer service to education, by developing a highly functional and interactive system that can learn and adapt to users' needs and operate in multiple languages, thus propelling the advancement of AI technology on a large scale.

Chapter 5

Applications

5.1 Description Implemation

In this section, we will address aspects related to the implementation of the chatbot, present a broader and more generalized view of what was developed and for that, and address the following:

- Configuration of the development environment;
- Installation and configuration of RASA;
- Creation of domains, intentions, entities, and the definition of conversation stories;
- Designed and trained the chatbot model.

Firstly, we chose Windows for our development work because we are familiar with the operating system made it easier to focus on creating the chatbot rather than having to resolve problems with the environment's configuration. . The use of readily available and accessible tools and resources was made possible by the selection of Windows as the operating system.

Regarding the installation and configuration of the components necessary for the development of this project, we can say that it was an obligation to insert all the mandatory dependencies for the use of TensorFlow. That is, TensorFlow is a dependency of RASA, and to be successfully installed on Windows 7 or a higher version, it was needed to install the C++ redistributable for the specific architecture of my machine, and for its application, it was necessary to restart it RASA [2023b].

After finishing the Tensorflow installation, Anaconda had to be set up. This decision is entirely up to the user, who may also choose to install the required parts directly on the computer. But since I think installing everything in an Anaconda environment will be simpler, I went with that option. For example, if you want to install a specific Python version, which is difficult if you do not use software like this or anything similar. Anaconda installation was nearly default, but to avoid difficulties, I opted to add it to

the environment variables because I didn't have any version of Anaconda installed on this machine RASA [2023b].

Proceeding when the installation was finished, the next step was to open Anaconda's command line and create a virtual environment, mainly to install the Python version more adequate to my needs, because Anaconda's default environment, named as base, according to the installation process I did, came with version 3.9.13 and I didn't want to use that version, I preferred to use an older version, for questions like code compatibility, specific dependencies, stability, among others. In other words, to create the new environment, I used the command *conda create -n multilingua python=3.7*, where multilingua corresponds to the name I decided to give the environment, then I activated the environment with the command *conda activate multilingua*, then already inside multilingua I uninstalled and installed pip, to ensure that it came with the latest version developed and this was done through three commands that we will mention RASA [2023b]:

- For uninstall pip: *python -m pip uninstall pip*;
- To ensure that the pip library can be installed again in the environment: *python -m ensurepip*;
- For install latest version of pip: *python -m pip install -U pip*

The installation of RASA was done quite simply. It was necessary to run in the command line of Anaconda the command *pip install rasa*. However, by choice, I decided to install an older version, so the command was the following *pip install rasa==2.3*. To verify that the installation was successful, I ran the command *rasa -h* RASA [2023b].

Afterwards, and already with the guarantee of having RASA in the system, I ran the command *rasa init*, which, in short, creates a project with example training data presented in the data folder, which are nlu, stories, rules files, and configuration files too RASA [2023b].

The following diagram shows the project structure created based on the command mentioned above. It is relevant to offer a more detailed explanation of some of the files mentioned to better understand the operation of RASA and deepen our understanding of these key concepts. This will also be addressed in the last two topics mentioned at the beginning of this section, which are the creation of domains, intentions, and entities, the definition of conversational stories, and the definition and training of the chatbot model. As we see in the project path, the models folder is empty, but we can generate a model, because we already have example training data, like natural language understanding data (NLU) and stories. To do that, we need to run the command *rasa train*.

```
/path/to/project
├── actions
│   ├── __init__.py
│   └── actions.py
├── data
│   ├── nlu.yml
│   ├── rules.yml
│   └── stories.yml
├── models
├── tests
│   └── test_stories.yml
├── config.yml
├── credentials.yml
├── domain.yml
└── endpoints.yml
```

If someone intends to commence chatbot development using RASA as the framework, I would recommend placing primary emphasis on five essential files: "domain.yml," "config.yml," "nlu.yml," "stories.yml," and "rules.yml."

Furthermore, the domain file assembles the chatbot's elements, while the config file provides the settings for the machine learning models as well as the configuration for the RASA nlu pipeline and the RASA core configuration. RASA processes incoming messages using a series of components. These components are performed sequentially in a processing pipeline described in config.yml. To personalize and finetune the model, I used an NLU pipeline; the pipeline I chose was a minor enlarged component of the basic initialization project, only a spaCY model related to the language element, which we will detail later RASA [2023a].

The nlu file contains examples of intentions and entities and is essential for training the natural language processing (NLP) model. The stories file is responsible for establishing the flow and direction of the conversation, while the rules file contains predefined rules for the dialogue policy. Each of these files has a specific function and requires further analysis.

RASA relies heavily on the domain.yml file, which is responsible for specifying the chatbot's domain. The intents, which represent the actions that the chatbot may execute, the entities that can be retrieved

from the input texts, and the responses that the chatbot can deliver are all included in this file. This information is carefully defined within the domain.yml file, connecting all of the chatbot's critical features RASA [2023a].

The config.yml file contains the necessary settings for the machine-learning models used by RASA. It defines the components of the natural language processing (NLU) pipeline, and from the chatbot dialogue, it is possible to specify different algorithms, such as recurrent neural networks (RNNs) or transformers. In addition, the config.yml file allows us to adjust hyperparameters and define training policies RASA [2023a].

Examples of intents and entities can be found in the nlu.yml (Natural Language Understanding) file. The chatbot's natural language processing (NLP) model is trained using it. This document contains illustrative sentences that users can input and that are linked to particular purposes. Additionally, pertinent sentences can be marked as entities that stand in for particular data that the chatbot needs to retrieve. To teach the chatbot how to comprehend and accurately categorize intents and entities, the nlu.yml file is crucial RASA [2023a].

The stories.yml file describes possible stories, or conversation flows between the user and the chatbot. Each story consists of a sequence of actions in which the user sends messages, and the chatbot responds with specific actions. Stories are used to train the chatbot's dialogue model, allowing it to learn how to handle different conversation scenarios and choose appropriate actions based on the user's messages RASA [2023a].

The rules.yml file includes the chatbot's dialogue policy rules. These rules govern the chatbot's behaviour in specific conditions. Unlike stories, which provide examples of conversation flows, rules are used to define specific behaviours and ensure that the chatbot follows those rules when a certain situation is met. When the chatbot's answer is deterministic and there is no need to train machine learning models, rules can be effective in dealing with simple circumstances RASA [2023a].

As we can see, these five files are fundamental for the development of a functional chatbot using RASA.They offer the foundation required for training conversation and natural language processing (NLP) models, as well as enabling the description of the chatbot's behavior in various scenarios. It is possible to investigate each of these files in greater detail and modify them according to the requirements of the project as the chatbot development advances and shifts towards more specific aspects, such as personalization, improvement of natural language processing (NLP), or integrations with external systems.

Furthermore, when the chatbot's development advances and understanding of the RASA framework is gained, it is possible to grasp how to use another critical file, actions.py. This is an unquestionably strong tool for inserting logic into the chatbot, that is, assigning intelligence to it through the design of custom

action classes that are activated in reaction to particular occurrences throughout the discussion. Such actions can include database queries, requests to other APIs, information processing, and other chatbot specific logic, allowing for the addition of particular behaviours that improve the user experience.

5.2 Chatbot Processing

As we saw in the previous section, each file required to build a chatbot using the RASA framework was defined and explained independently. The connection between all of these, though, was not clarified, and we will illustrate this with the use of a simple example of the work's developing architecture.

When I tested the project automatically generated by the RASA init command, when the user said hello, in the English language, the chatbot would reply with a similar message, and the way this was implemented is split into several parts, as I will show. First of all, in the nlu.yml file, a list or a category of words was defined regarding what the user could say, and that list is inside an intent that is called greet (Figure 4).

```
nlu:
- intent: greet
  examples: |-
    - hey
    - hello
    - hi
    - hello there
    - good morning
    - good evening
    - moin
    - hey there
    - let's go
    - hey dude
    - goodmorning
    - goodevening
    - good afternoon
    - olá
    - oi
    - bom dia
    - boa tarde
    - boa noite
    - hola
    - buenos dias
    - buenas tardes
    - buenas noches
    - buenas
```

Figure 4: NLU greet part.

Next, the stories file will contain a specific direction of the conversation for that intent, and if everything goes well, that is, if the model is well trained, for example, that direction will happen, and the way this is coded is as follows (Figure 5).

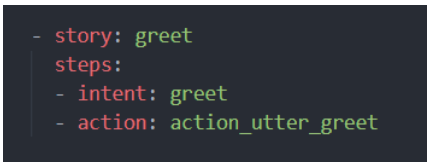


Figure 5: Stories greet part.

To conclude, in the domain file we have a list of intents (Figure 6).

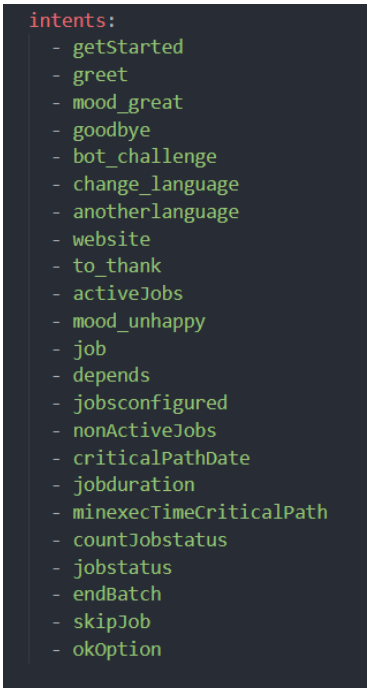


Figure 6: List of intents in the domain file.

and a list of actions I needed to define (Figure 7):

Figure 7: List of actions in the domain file.

As bot developers, we found it vital to convert all responses into actionable steps. If we had continued to use replies up to this point, the response for this specific case would look like this, presenting a visually simpler and more intelligible structure.(Figure 8).

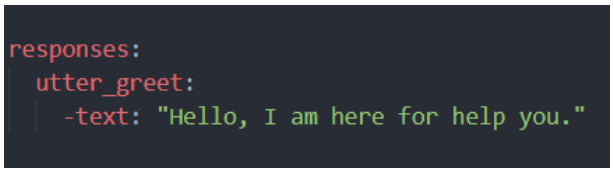


Figure 8: Reponse example in the domain file.

So the way that all of this is processed is like this: When a user greets the chatbot, for example, by saying "hi," RASA will check to see if the word the user mentioned is in the list of words present in the nlu file. If the word is in the list, the greet intention is recognized, and two files are triggered, the domain file and the stories, because both have the same intent.

Starting with the stories file, and as shown in the code above, the name of the story was specified, which is precisely the name of the intention, and that was my choice to highlight the code. Then there are the procedures to follow, which are: when the intention *greet* is raised, the action *action_utter_greet* is performed, which is nothing more than a response like "Hello, how can I help you?" with logic mixed in, as shown further down.

In the domain file, we must define a list of all the intentions and actions defined throughout the chatbot's development so that the model is well-trained and the entire process defined is well executed. All of the actions characterized in the illustrations are functions defined in the file *actions.py*.

Now as we saw in the state-of-the-art case study section, a possible architecture for a chatbot was presented that explains the structure underlying how RASA works. Using architecture and taking into account the previous explanation, I will make an extension of the diagram to help the textual explanation. To demonstrate where the files are embedded and how they are used and activated.

However, in the example I gave about the interaction between the user and the chatbot when both are satisfied, all of this is shown through inputs, i.e., one on the user's side and another on the chatbot's side, and everything is demonstrated in *Botfront RASA Webchat*, which is a chat widget to deploy virtual assistants created with RASA.

To put it another way, it's a react component that enables you to provide the user with a fun and engaging graphical interface where you can communicate with the chatbot and get responses in real-time. The key factor that convinced us to choose "Botfront" was its capacity to supplement RASA's resources by offering new ones and enabling the development, instruction, and use of chatbots. Later on, tests will be offered, allowing you to see all of its capabilities.

Finally, I'd want to present another component: the rules file. As I previously stated, this file is deterministic in the sense that it allows the creator to set certain responses for specific situations, such as user interactions. This file's content may be brief, but it is critical to the interaction.

```yaml
rules:

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: action_utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
  steps:
  - intent: bot_challenge
  - action: action_utter_iambot
```

Figure 9: Rules file.

We can see that configuring in YAML format is precisely the same as defining a tale. The only difference is that we name the rule rather than the story. In addition, rather than a description, we write exactly what we wish to happen. I'll give two examples: when the user says "goodbye," the chatbot will also say "goodbye"; and when the user challenges the chatbot, the action of telling him that he is a bot will be activated to avoid improper replies. In other words, we can think of this action as a backup message that

is triggered when there is an insufficient knowledge base to answer the user's inquiry, particularly if the question is unrelated to the case under consideration.

Another important aspect to mention is that the illustrative example that we see below, it has information about a list of entities and slots, something that was not explained in the greeting example because, for that specific action, it was not necessary to use it. Briefly, the entities serve to capture a specific word from the message provided by the user.

For example, and now speaking more intrinsically in the project, let's imagine that the user wants to know what the functional area of a job is, he can send a message like "What is the functional area of job x, where x represents the name of the job" What the chatbot will do in this situation is the following: it stores the word x and assigns its value to an entity called a job. This entity can store other values that are defined by me. Then the response to the user is sent informing the respective functional area.

To conclude, I would like to point out and reinforce that another architecture diagram was not built, as this corresponds to the process of creating visual representations of the components of the software system, and in the following one components such as rectangles with specific RASA commands and files were added. After the analysis, it is expected that it will be easier to consolidate the knowledge of how everything is interconnected and processed (Figure 10).

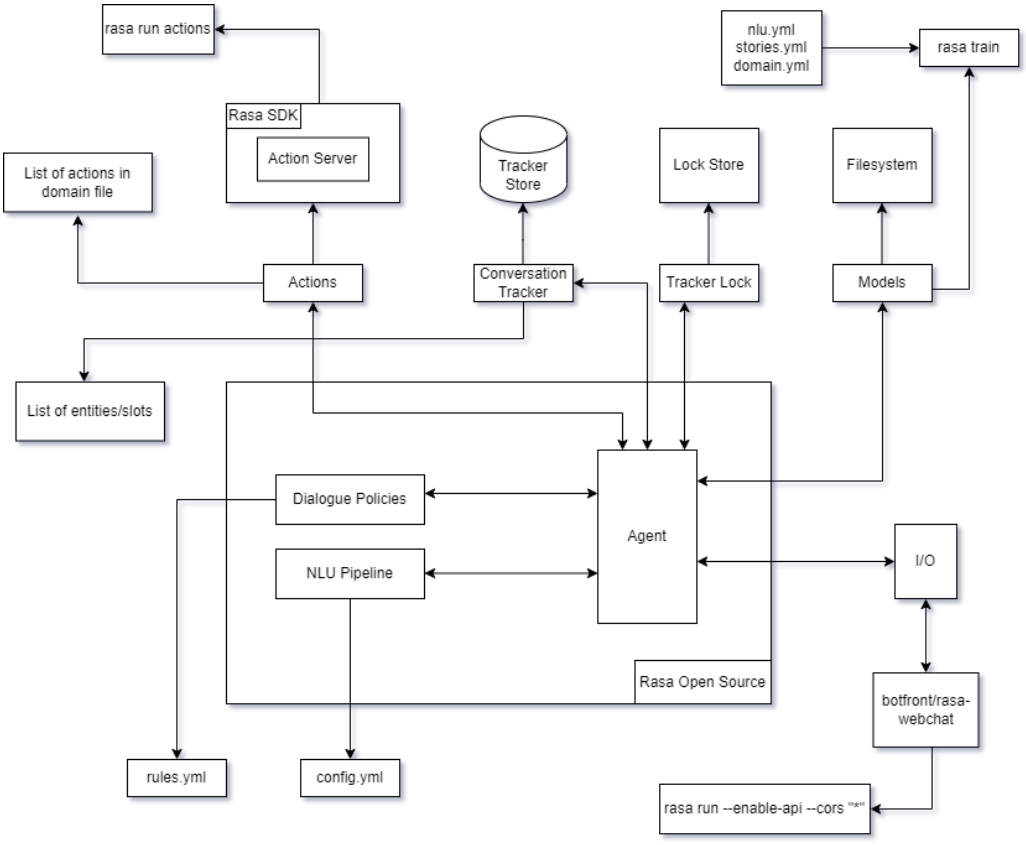


Figure 10: Explanatory diagram to aid writing (based on RASA [2023a]).

5.3 Chatbot Architecture

In addition to using visual diagrams to help explain files and commands, the development of intelligent chatbot systems is an important aspect of modern communication systems. Advanced frameworks and components are used in these systems to enable interactive and efficient user-bot interactions. In the parts that follow, we will look at the high-level architecture of a chatbot system that has been meticulously developed to handle seamless chats, multilingual support, database integration, and integration with external modules. We hope to highlight the underlying components and their relationships by investigating their design, shedding light on the intricate workings of this intelligent communication system.

The chatbot system's high-level architecture is made up of various interconnected components that allow for effective communication between the user and the bot. At the heart of the architecture is a central box dubbed "Chatbot," which houses the main intelligence and logic that drives the bot's activity. To offer a smooth conversational experience, the chatbot interacts with other components that were implemented with the help of RASA used to design and train the bot, which is one of the essential components coupled with the chatbot. RASA supports natural language comprehension and processing, allowing the bot to interpret human input and create suitable answers.

A User Interface (UI) is linked to the chatbot to ease user interaction. The UI represents the chatbot visually, giving users an interface through which they may engage and get replies and is powered by RASA Botfront WebChat.

The "Multilingual" box includes the chatbot's multilingual functionality and he can speak three languages, English, Portuguese and Spanish, being English the main one.

The chatbot system also includes automated responses to frequently asked questions. These responses are saved in databases and can be retrieved by the bot as needed, with selections and corrections dependent on human input. Additionally, connections to the "IBPS" and "IPE" modules, as well as their associated databases, are also included in the architecture. The chatbot can communicate with these modules and get needed data and functionality thanks to these connections.

Overall, this high-level design demonstrates the interconnection of numerous components, allowing the chatbot to interpret user inputs, provide suitable responses, support multiple languages, and effectively interface with external modules and databases (Figure 11). To conclude, I would like to mention that this high-level architecture of the project was defined with the help of the article Faria et al. [2022].

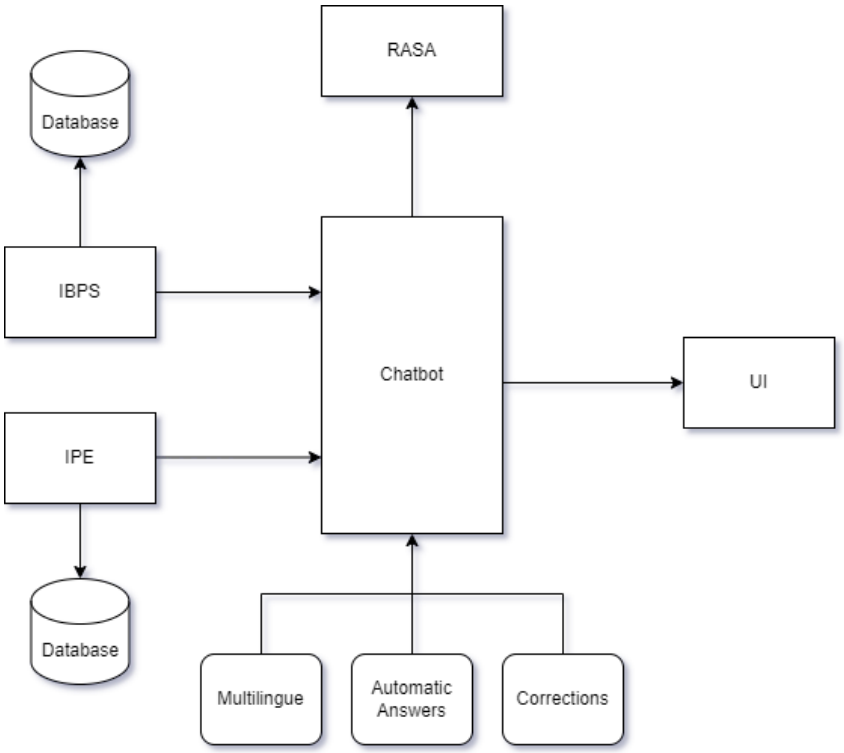


Figure 11: Chatbot high level architecture.

5.4 Chatbot Incorporation

I'll start off by discussing how my chatbot was included into the business context by illustrating this architectural image before moving on to the technical context of the program (Figure 12).

Due to its integration with *Retail Consult* and its incorporation into the Intelligence Batch Processing System (IBPS) module, the chatbot in question plays a crucial role as an internal tool of great value for the organization. In addition, a link is established to the Intelligence Processing Engine (IPE) module, which enables access to the same database.

In terms of architecture, the chatbot is integrated into the application layer, which includes a module with functionalities such as task scheduling and forecasting. The main components are IBPS and IPE. In addition, Oracle Retail, machine learning, and artificial intelligence are all connected to the database, constituting the data layer.

The chatbot acts as an internal assistant, providing information to the company's employees. When requested, it can perform direct queries to the database, making specific "selects". This functionality is incorporated into the chatbot's automatic responses, as highlighted in blue in the figure above.

This ability to obtain information about the data quickly and accurately, without the need to manually query the data, is enabled by the integration of the chatbot into the organizational structure.

To add, one of the main features of the chatbot is its ability to perform correction routines. For example, a user only needs to submit a request to skip a certain job. This functionality of skipping a job, offered by the chatbot, is an option present in the corrections, also highlighted in blue in the image.

This functionality provides better administration and control of the system processes. The chatbot acts as a reliable assistant, providing quick and efficient responses to user requests, including, when necessary, the modification of specific routines, present in the patches.

In short, the chatbot performs tasks and implements changes for the company's employees, as well as providing relevant information. Its integration into *Retail Consult*'s servers extends its usability and accessibility to all users, increasing efficiency and productivity in the work environment.

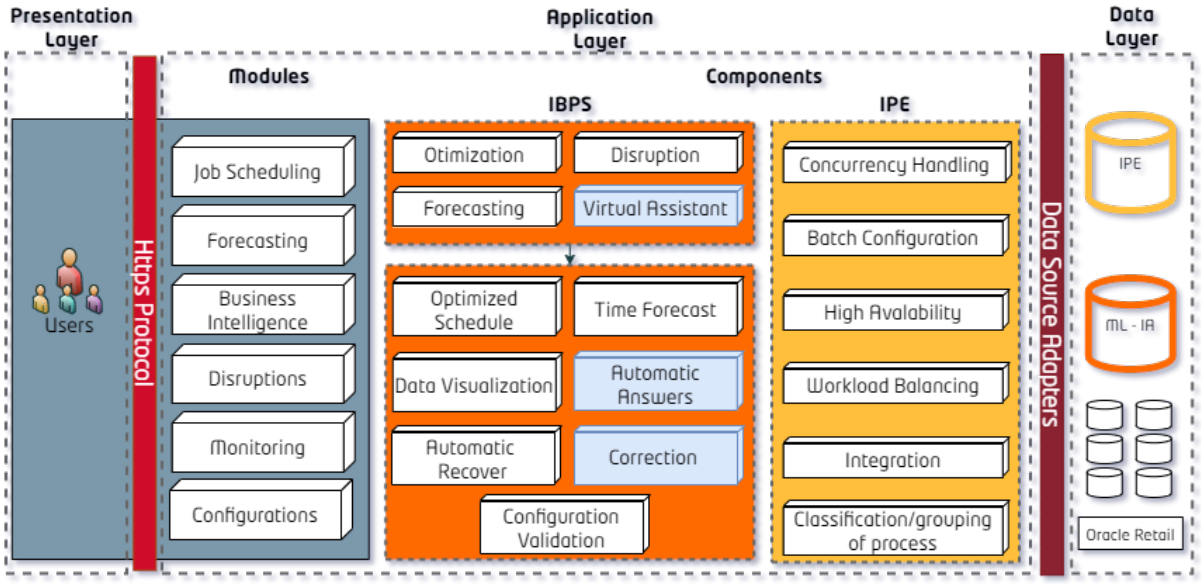


Figure 12: Incorporation of chatbot in the business context.

5.5 Multilingue

Allowing the chatbot to converse in many languages was one of the key problems and needs in the creation of this dissertation. The usage of this in the context of a global firm that works in numerous countries, specifically Portugal, Germany, the United States, Brazil, Mexico, Chile, and China, was a major difficulty and demand in the production of this research.

Although English is the chatbot's native language and is frequently used in international settings, it is a common practice to speak in one's mother tongue when conducting business. This is because cultural, historical, and geographic factors can generate considerable pronunciation and accent differences in English, depending on the country.

Given the company's global reach, it is imperative to comprehend how various regional tongues and dialects may influence the English language, resulting in a variety of accents and communication styles. Understanding that these variances represent rich cultural and linguistic variety rather than a language's shortcomings is crucial for improving communication on a global scale. It was planned to create a chatbot that could understand and respond in several languages for these reasons.

The system was split into two primary components to address this need: identifying the user's language and providing a response in that language.

When the chatbot appears, a welcome message with the following text is displayed: "Welcome! Your virtual helper: I'm RC Bot. This strategy aims to give users a welcoming and approachable experience, regardless of the language they choose to communicate with the chatbot in (Figure 13).

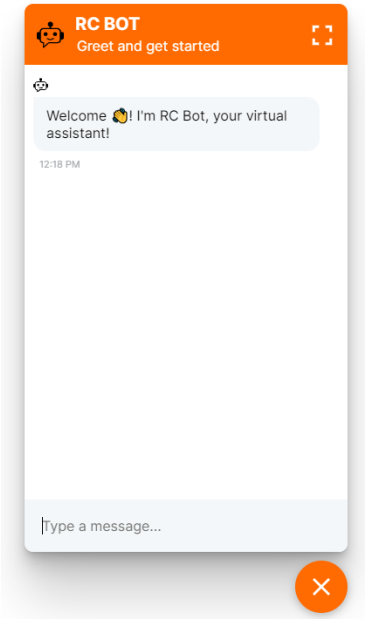


Figure 13: Welcome message to the user.

The user must tell the chatbot in English whatever language they want to speak for the chatbot to change, which is the sole constraint in the context of combining many languages. The chatbot cannot change the language if the request is not made in English. For instance, if you want to speak Portuguese or Spanish, just say "speak Portuguese" or "speak Spanish" in English to indicate your selection. The chatbot will only react in the chosen language moving forward until the user chooses to switch back to it. The future operation of this interaction will subsequently be visually demonstrated (Figure 14).

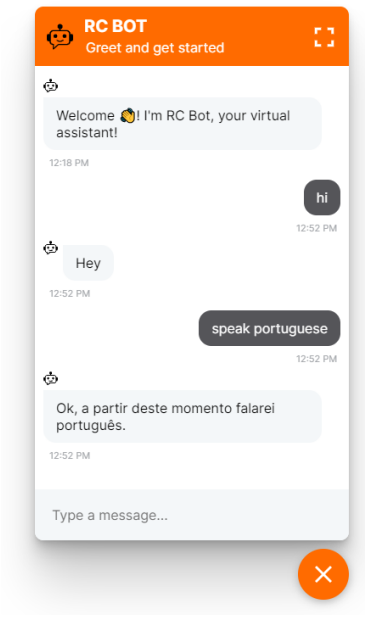


Figure 14: Test witch Language - English to Portuguese.

Users can converse freely and naturally in their favourite language thanks to this tailored approach. A more inclusive and effective work environment is created by the chatbot since it encourages greater comprehension and interaction by speaking the same language as the user. When the user asked the chatbot to speak Portuguese, just as seen in the previous image, it answered by saying that it would, having previously done so. On the other hand, a conversation with the chatbot in Portuguese has not yet been demonstrated; it was only apparent that the language shift had been successfully implemented. The user and the chatbot can then be seen having a conversation in the following picture, considering the Portuguese language as a language and the progression of the conversation in the prior image (Figure 15).

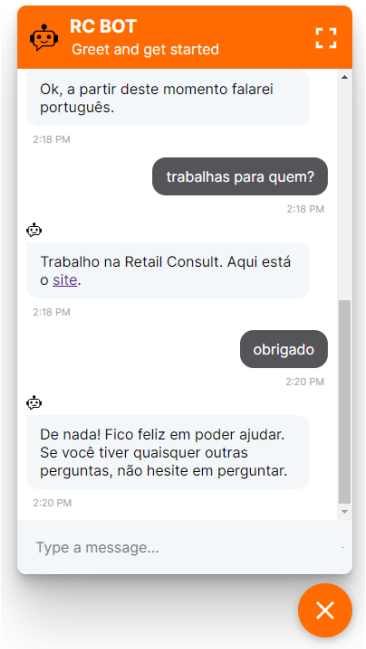


Figure 15: Test user-chatbot conversation in Portuguese.

Additionally, I would want to draw attention to some additional routes that are connected to the multi-lingual area. For each of these paths, I will include an illustration to support the functionalities that will be explained. As we can see, the chatbot also speaks Spanish (Figure 16).

Figure 16: Test user-chatbot conversation in Spanish.

If the user asks the chatbot to speak English and the language is already in English, it will inform them of this information and tell them which languages they can speak (Figure 17).

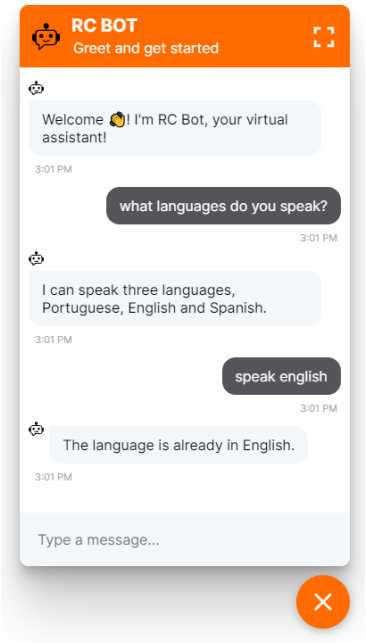


Figure 17: Test user-chatbot conversation in English.

Finally, the user has the option to switch languages as many as they like while speaking (Figure 18).

Figure 18: Test user-chatbot multiple languages.

This system's implementation required considerable consideration to support two-way communication with the user and allow for dynamic language change. The usage of a bespoke pipeline component—which the programmer can create from scratch or choose from preexisting options—enables this functionality. In this instance, it was decided to use Spacy, a library built into the RASA program that has various developed components, as was already discussed in the research section of this dissertation.

The *en_core_web_md* is built with a CPU-optimized pipeline and is intended to handle English text quickly. The existing RASA NLU models can be improved by adding the *en_core_web_md* model to the RASA NLU configuration file. As previously indicated, these models can be altered to meet the unique requirements of the RASA [2023a] project. A portion of the *RASA train* command process will then be displayed, showing RASA attempting to load the model into *rasa.nlu.utils.spacy_utils* which is the component that gives others, in this case, my chatbot, access to the shared loaded SpaCy model. Next, we can see the model being added to the RASA *rasa.nlu.components* (Figure 19).

```
Training NLU model...
2023-07-19 13:20:59 INFO     rasa.nlu.utils.spacy_utils  - Trying to lo
ad spacy model with name 'en_core_web_md'
2023-07-19 13:21:23 INFO     rasa.nlu.components  - Added 'SpacyNLP' to
 component cache. Key 'SpacyNLP-en_core_web_md'.
```

Figure 19: Spacy model 'en_core_web_md' loaded and added.

It's also necessary to note that this component is essential for automatically determining the user's language. Spacy's approach deciphers linguistic elements from the input text and establishes the domi-

nant language based on grammatical patterns, vocabulary, and sentence structure. Real-time language identification makes it possible for the chatbot to respond in an appropriate and individualized way.

Slots were used to implement the task of storing the language value that was discovered. Slots function as the chatbot's internal memory, allowing pertinent information to be saved and accessed as needed. The user's language is regarded as a piece of persistent information in the particular context of this system. The chatbot can modify its response and linguistic behaviour in accordance with the user's preferences and demands by saving the detected language value in the slot designated as *language*.

For instance, a user might say, "Hello, how can I help you?" to begin a conversation in English. The *en_core_web_md* model analyzes and recognizes the input using Spacy's pipeline. Then, during the dialogue, the value "English" is stored and in charge of maintaining the language information. The chatbot can modify its responses based on the language used as the conversation progresses. Spacy's pipeline, for instance, recognizes when a user switches from English to Portuguese in the middle of a chat and modifies the language slot value, for Portuguese.

Therefore, the system can efficiently detect and track the language used by the user in real-time, providing a fluid dialogue tailored to individual linguistic needs. This is done by combining the Spacy pipeline, specifically the *en_core_web_md* model, with the storage of information in slots.

Since the chatbot's knowledge base for user inputs is stored in the nlu file, we have been able to understand that the component can detect and store the language so far. However, it can only do this if the nlu file contains information related to languages. As the bot speaks two other languages in addition to English, i.e. examples in both Spanish and Portuguese were added to the list of each intent. The next figure will display an example (Figure 20).

```
- intent: job
  examples: |-
    - Which is the functional area of [prepost_rpl_pre](job)?
    - Qual é a área functional de  [prepost_start_batch_pre](job)?
    - I have this [priceChangeExtractRetry](job) what is the functional area?
    - Eu tenho [purgeFulfillmentOrders](job) qual é a área functional?
    - Which is the functional area of [purgeAdHocStockCount](job)?
    - Qual é a área funcional de [purgeShelfReplenishment](job)?
    - What is the functional area of [reimaccountworkspacepurge](job)?
    - I have this [resa2dw](job) what is the functional area?
    - I have this [rmse_aip_alloc_in_well](job) what is the functional area?
    - I have this [rmse_rpas_domainPROGRAM_NAME](job) what is the functional area?
    - What is the functional area of [saescheat](job)?
    - What is the functional area of [salesgenrej](job)?
    - Which is the functional area of [alcSnapshotAllocIn](job)?
    - I have this [batch_itmcostcompupd](job) what is the functional area?
    - ¿Cuál es el área funcional de [schedprg](job)?
    - ¿Cuál es el área funcional de [wfordupld](job)?
```

Figure 20: NLU multilingue example.

The chatbot responding to the user in the same language is another feature that needs to be explained. To accomplish this, I had to refactor all the code I had already implemented because, up until this point, all I had were responses. Now, for the chatbot to respond to the user in the language he speaks, I must use actions, specifically custom actions, where I can access the value of the language that is stored in the slot.

To sum up, the user must receive the response, which was created with the use of a Google-powered translation API named *_deep_translator*, specifically "Google translator." For instance, if the language is Portuguese, the answer that was going to be sent in English is translated into Portuguese before being sent. The action below illustrates what was just discussed.

Algorithm 1 Change Set Language Algorithm

1: **procedure** ActionChangeSetLanguage(dispatcher, tracker, domain)
2: $language \leftarrow$ tracker's slot value of "$language$"
3: **if** tracker's latest message contains "speak" or "talk" followed by "portuguese" **then**
4: DispatchMessage("Ok, a partir deste momento falarei português.")
5: **return** slot set "language" to "Portuguese"
6: **else if** tracker's latest message contains "speak" or "talk" followed by "english" **and** $language$ is "English" **then**
7: DispatchMessage("The language is already in English.")
8: **return** slot set "language" to "English"
9: **else if** tracker's latest message contains "speak" or "talk" followed by "english" **and** $language$ is not "English" **then**
10: DispatchMessage("Ok, I will speak in English from now on.")
11: **return** slot set "language" to "English"
12: **else if** tracker's latest message contains "speak" or "talk" followed by "spanish" **then**
13: DispatchMessage("Ok, hablaré en español de ahora en adelante.")
14: **return** slot set "language" to "Spanish"
15: **end if**
16: **return** empty list
17: **end procedure**

This personalized feature allows the chatbot to modify the user's language. The following are the function's specifications:

- `class ActionChangeSetLanguage(Action)`: This line generates the class, which is an in-

heritor of the base class Action. According to this, the unique class `ActionChangeSetLanguage` has particular properties and behaviours that make it suitable for usage as an action in RASA;

- `def name(self):` The action's name is provided by this class method. RASA bases what the user should do on his aim, hence the name is essential. In this instance, the action name is written as `action_change_set_language`;

- `def run(self, dispatcher, tracker, domain):` The run method is required by a RASA action class. A certain action is carried out when it is prompted during the dialogue flow. The processes that take place within the approach are described in full below:

  - `language = tracker.get_slot("language")`: Here, the tracker's "language" slot's current value is recorded. This slot can be used to hold the language that the user desires to utilize;

  - The code then checks the content of the user's most recent message (fourth point) using regular expressions, often known as regex, to ascertain the message's intent concerning the intended language. There were three regex patterns examined:

    1. `re.search(r'\b(speak|talk|in|) portuguese\b'`: This regex examines if the user uses "speak," "talk," or "in" after the word "Portuguese" in any sequence. If this is the case, the chatbot will respond by switching the "language" slot to "Portuguese" and announcing that it will now communicate in that language;
    2. `re.search(r'\b(speak|talk|in|) english\b'`: This regex tests the term "English" immediately after it at the top. There are two potential outcomes:
       * If "English" is already selected as the current language, the chatbot replies in English;
       * The chatbot will respond that it will now speak in English and change the value of the "language" slot to "English" if the language it is speaking in is not "English".
    3. `re.search(r'\b(speak|talk|in|) spanish\b'`: This regex verifies once again whether the user entered "speak," "talk," or "in" after the word "Spanish." If so, the chatbot changes the "language" slot to "Spanish" and responds that it will now speak in the language;
    4. `tracker.latest_message.get("text"), re.IGNORECASE`: The first is for the tracker to get the most recent message from the user so that it may parse it using

a regex, and the second is pattern matching, which ignores the distinction between capital and lowercase characters. In other words, it makes the regular expression case-insensitive, enabling capital and lowercase letters to be regarded equally for pattern detection.

  - The function will return a message to the user if none of the requirements are met, indicating that the chatbot didn't understand the input. The content of the input message is this: "I apologize, but I didn't fully comprehend your previous message. Could you please repeat it? Also, kindly ensure that the words 'Portuguese,' 'English,' or 'Spanish' are correctly written."

- `dispatcher.utter_message(text='...', parse_data=True)`: Sending messages back to the user is the dispatcher object's purpose. Based on the predefined circumstances, the chatbot will respond to the user by using the `utter_message` method. The chatbot will reply with material that is surrounded by quotation marks;

- `return [SlotSet('language', '...')]`: Here, the code modifies the value of the "language" slot to a different language, such as "Portuguese", "English", or "Spanish", using the SlotSet method.

The `ActionChangeSetLanguage` function is in charge of altering the chatbot's language of interaction with the user. Regex, commonly referred to as regular expressions, is used to ascertain the user's intent in this process. It responds following the circumstances tested, modifies the "language" slot's value, and then returns applicable messages.

As was previously noted, the specific action is triggered in both the stories file and the domain file. There are similarities between how the stories are portrayed, with the exception that the names of the intended action and the action to be called have been changed.

The method for that specific activity is different from what has been shown so far, and the user wants to share a picture of the Natural Language Understanding (NLU) file (Figure 21).

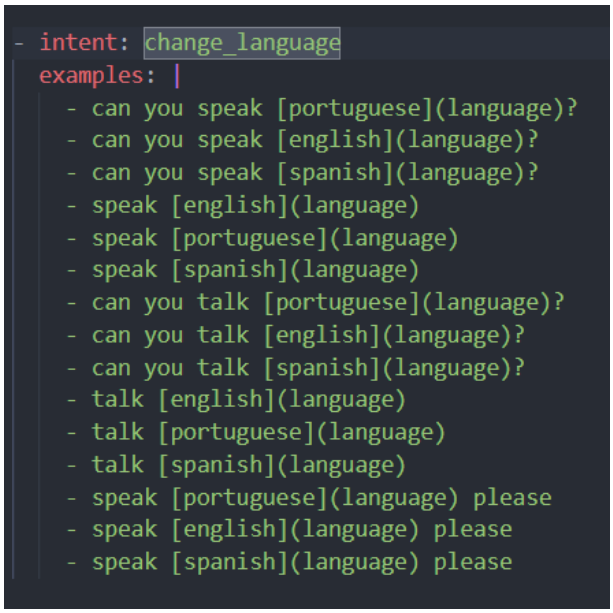


Figure 21: Language nlu file.

It is crucial to make sure that all actions are accurately described in the domain and that the references to them in the stories are accurate as the project moves forward. Additionally, upgrading the NLU archive is essential to enhance user comprehension and interpretation.

The term language is not just a slot but also an entity; we may say that these are pieces of data that were taken from the messages that the user submitted, and RASA represents this as follows: The format is (value) [name], where name is the word "language" and value is either Portuguese, Spanish, or English. The language entity will have the value Portuguese if the user, for instance, sends the chatbot a message with the following content: Speak Portuguese. The chatbot won't be able to understand the user's input if they use any other words outside the first three.

I want to emphasize that every action used in this project is capable of identifying the language being spoken and, if the language change action is activated, translating to the language the user requests. To wrap up, it is important to draw attention to another important action that was an allusion to in this section: the query, "What language do you speak?" This function gives the system more knowledge, enabling the chatbot to answer appropriately by enunciating the number and the languages it understands. The code for this action may be seen in the following image.

To conclude, this action has a list of potential responses, so when the user activates it, one of the list's items is chosen and communicated to the user. Because the user may ask the same question during a discussion or even in a new query, the chatbot's writing option ends up being enhanced, enhancing the text's interest and adding variety and dynamism.

Algorithm 2 Action Another Language Algorithm

procedure ActionAnotherLanguage(dispatcher, tracker, domain)
    $language \leftarrow$ tracker's slot value of "$language$"
    $english\_responses \leftarrow$ ["I can speak three languages, Portuguese, English and Spanish.", "I speak two more languages, Portuguese and Spanish.", "I can speak Portuguese and Spanish, if you want."]
    if $language$ is "Portuguese" **then**
        $translated\_responses \leftarrow$ Translate each response from $english\_responses$ to Portuguese
    else if $language$ is "Spanish" **then**
        $translated\_responses \leftarrow$ Translate each response from $english\_responses$ to Spanish
    else
        $translated\_responses \leftarrow english\_responses$
    end if
    $response \leftarrow$ randomly choose one from $translated\_responses$
    DispatchMessage($response$)
    return empty list
end procedure

5.6 Social Actions

As we know, the chatbot can respond to specific questions by calling dynamic and non-dynamic queries and HTTP requests, making real-time corrections to the database, supporting workers, and providing individualized support by providing clear and unbiased answers.

On the other hand, this section will address the interaction and communication activities of this chatbot, which were an important focus of this study. By establishing a consistent and friendly dialogue with users, they hope to provide a more pleasant and seamless experience to establish a deeper relationship with people, express gratitude and praise, and pose as a transparent bot. This method seeks to improve human-machine interaction, making it more pleasant and efficient in a range of applications. The information that we used to improve the dialogue with the user was this table (Table 17) that contains the action name, example of sentences that the user writes to the chatbot and examples of sentences that the chatbot writes to the user:

Table 17: Protocol Social Actions.

Actions	**User examples phrases**	**Chabot examples phrases**
action_utter_greet	"Hey", "Hello", "Hi"	"Hi! How are you?", "Hi there!", "Hey"
action_utter_happy	"Perfect", "Great","Feeling like a king", "I'm good, thanks"	"Great, carry on.", "Great", "Great, how can I help you?"
action_utter_unhappy	"My day was horrible", "I don't feel very well", "I am sad"	"I'm sorry to hear that.", How can I help you?", "I'm sorry to hear that you're feeling sad.", "Sometimes it can help to share your thoughts and feelings with someone."
action_utter_goodbye	"Bye", "Goodbye", "Have a nice day"	"Bye!", "Goodbye!", "See you soon!", "See you later"
action_utter_iambot	"What are you?", "Are you a bot?", "Are you a human?"	"I am a bot, powered by RASA.", "I'm a chatbot!", "Beep, boop, I'm a bot."

action_ok_option	”Ok”, ”Okey”, ”Okay”	”Can I help you with something more?”, ”May I assist you with more?”, ”Do you need to know more information?”, ”If you need any further assistance, I’m here to help.”
action_utter_thanks	”I appreciate that”, ”Thanks”, ”Thank you”, ”Good Help”	”You are welcome! I’m happy to be able to help. If you have any other questions, don’t hesitate to ask.”

5.7 User Interface

We all know that as technology has advanced, there has been an increasing need for graphical interfaces since they offer a more effective and simple user experience, facilitating access to and perception of more complicated applications. For this reason, the features that were incorporated into the designed graphical interface will be examined and analyzed in this part.

It is feasible to confirm that after the chatbot is closed, the message shows as though it were a notification as seen in the following image. (Figure 22).

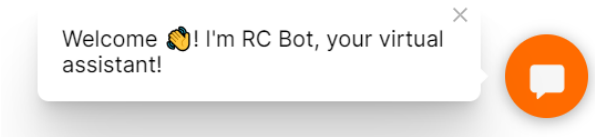


Figure 22: Chatbot close message.

The chat interface displays the user’s messages with a grey backdrop to indicate the message time, while the chatbot’s messages have a white background. By pressing a button, users may go fullscreen, and it stays fullscreen even after closing and reopening. Click X to end fullscreen. The interface is titled ”RC BOT” and includes the subtitle ”Greet and Get Started.” (Figure 35)

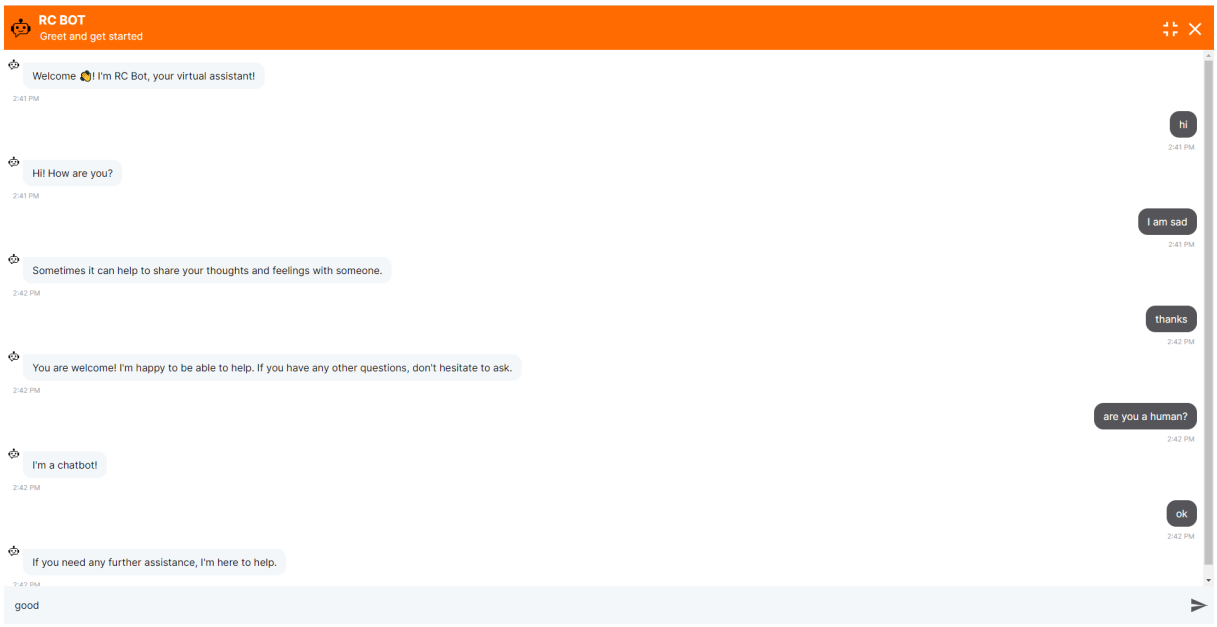


Figure 23: Chatbot fullscreen mode.

Assume that while the user is not writing any messages, an init payload with the message "Type a message..." shows. In this image, we can see that the user is typing or has completed typing everything; here is the word "good," and that message has disappeared; what is now visible is the symbol to send the message.

When you activate the bot, the message icon displays, as seen in the figure below. When we launch the chatbot, we notice an x that implies that tapping it would close the chatbot. The chatbot has an orange backdrop, while the user's messages have a white background that complements the company's color scheme.

5.8 Logic Actions

Other activities implemented in this work that required more logic have not yet been fully discussed, such as joining both the IBPS and IPE database. Furthermore, I will now demonstrate all of the actions that have been developed in this bot, such as demonstrating which and how many jobs are active and not active, which jobs have pre and post-dependencies, which jobs are configured in batch processing, which is the functional area of a specific job, which is the critical path and the minimum execution time of the same for a given day, how long it will take to process a given job, provide information about the status of a job, and so on, for example, a task can be finished, running, submitted, or with an error, and the chatbot can advise you of the number of jobs in each stage, how long it will take to finish the batch if it is running, and ultimately, we can skip work from the database.

The actions that deserve more attention are those with the ability to make dynamic queries, that is, queries that are made based on what the user is asking, and of this type of action, we have some of those that I mentioned above, starting with the one that provides information about the job's status.

When a user asks an inquiry about the status of jobs, the chatbot can perform a specific action called job status. It interprets the user's language and the job status they wish to check using information stored in slots.

Algorithm 3 Determinação de Status de Trabalho

```
1: procedure JobStatus(dispatcher, tracker, domain)
2:     status ← tracker.get_slot('status')
3:     if not status then
4:         dispatcher.utter_message('Desculpe, não entendi...')
5:         return
6:     end if
7:     status ← status.strip().lower()
8:     response ← 'Unknown status.'
9:     status_map ← {"submitted" : "Thesubmittedjobsare : ","working" :
       "Thejobsthatareworkingare : ","finished" : "Thefinishedjobsare : ","error" :
       "Thejobsthatshowederrorsare : "}
10:    if status in status_map then
11:        response ← status_map[status]
12:    end if
13:    results, error ← some_query_function(status)
14:    if not results then
15:        dispatcher.utter_message('O lote não está em execução no momento')
16:        return
17:    end if
18:    if not error then
19:        language ← tracker.get_slot('language')
20:        translated ← response
21:        if language is 'Portuguese' then
22:            translated ← GoogleTranslator(source='en', target='pt').translate(response)
23:        else if language is 'Spanish' then
24:            translated ← GoogleTranslator(source='en', target='es').translate(response)
25:        end if
26:        dispatcher.utter_message(translated)
27:    else
28:        dispatcher.utter_message(error)
29:    end if
30: end procedure
```

The function indicated in the image is the one that will be explained in detail:

The function begins by reading the tracker object's values for the slot's language and status. The slots are variables that hold vital information during the user conversation. The language column most likely contains the user's language preference, and the status slot contains information on the status of the "job" the user is inquiring about.

As previously mentioned, this function handles the task of storing the information entered by the user. In such a scenario, if the user inquires about the status of a job that either doesn't exist, is spelled incorrectly, or if they omit the job's name altogether, the chatbot will respond by expressing that it didn't quite understand the user's request and will politely request them to provide a clearer phrasing.

To ensure that comparisons with status keywords are case-insensitive (meaning it doesn't care about letter case), the code strips away any extra spaces and converts the contents of the status slot to lowercase. This way, users can input the status in any format, and the chatbot will still understand it correctly.

Once the status slot value has been processed, the function checks if it matches any of the known keywords, like "submitted," "working," "finished," "error," and so on. If there's a match, it assigns a status code (S, W, F, or E) and crafts an appropriate response message in English.

If the status slot value doesn't match any of these known keywords, the chatbot will respond in English with "Unknown status" to let the user know that the specified status is not recognized.

The function then searches the database for job names that match the specified status. The results of this query are stored in the results variable, and the error variable helps determine whether any issues arose during the query. It's worth noting that the question is constructed using words supplied by the user, which is what I wanted to highlight. In this scenario, we have an action that can run four different queries, making the query process dynamic.

If the query returns no results (results == []), the chatbot will inform the user that no jobs are currently in progress.

In case there's an error during the database query (non-empty error), the chatbot will notify the user about the technical problem.

If the user specifies a language (like "Portuguese" or "Spanish"), the chatbot will use the Google Translator API to translate the English response into the chosen language. If no language is specified or if it's not supported, the chatbot will reply in English. Using the tabulate library, the chatbot constructs the final response, comprising the translated (or English) response message and the names of the "jobs" in a table format. This response is delivered to the user. To signify that the action has been done, the function returns an empty list [].

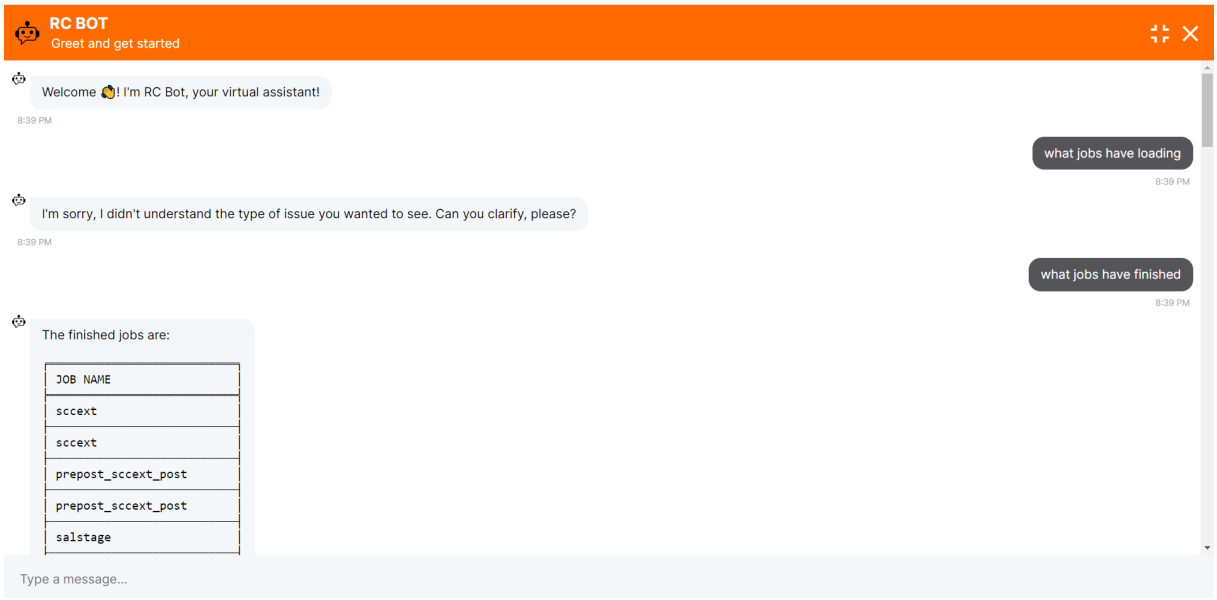


Figure 24: Test job status.

Another function that is unquestionably important to explain is the action that deals with determining the minimum batch execution time for a specific date because it praises the chatbot's error handling, as well as its ability to help and assist the user if the user, for whatever reason, has difficulty typing what he wants.

The verbal action is represented in the image; in addition to the code being discussed in depth, it will also be exhibited in execution.

Algorithm 4 Critical Path Execution Time

```
class CriticalPathExecutionTime(Action):
function name(self) return "action_criticalpath_executiontime"
end function
procedure run(self, dispatcher, tracker, domain)
    language ← tracker.get_slot('language')
    slot_value ← tracker.get_slot('date')
    if not slot_value or not validate_date_format(slot_value) then
        dispatcher.utter_message('I'm sorry, I didn't understand or please mention the date in the format...')
        return []
    end if
    formatted_date ← convert_date_format(slot_value)
    results, error ← run_query_IBPS( SQL_query_with_formatted_date )
    if error then
        dispatcher.utter_message(error)
    else
        if results then
            table ← tabulate(results, headers=..., tablefmt='fancy_grid')
            english_response ← 'For the critical path on the date...'
            translated ← GoogleTranslator(source='en', target=language).translate(english_response)
            dispatcher.utter_message(translated)
        else
            dispatcher.utter_message('No results found for the specified date.')
        end if
    end if
    return []
end procedure
```

Below we have the test against the function represented and explained above, and as we can see the chatbot interpreted the question correctly and answered by providing the time it takes to execute the minimum execution time

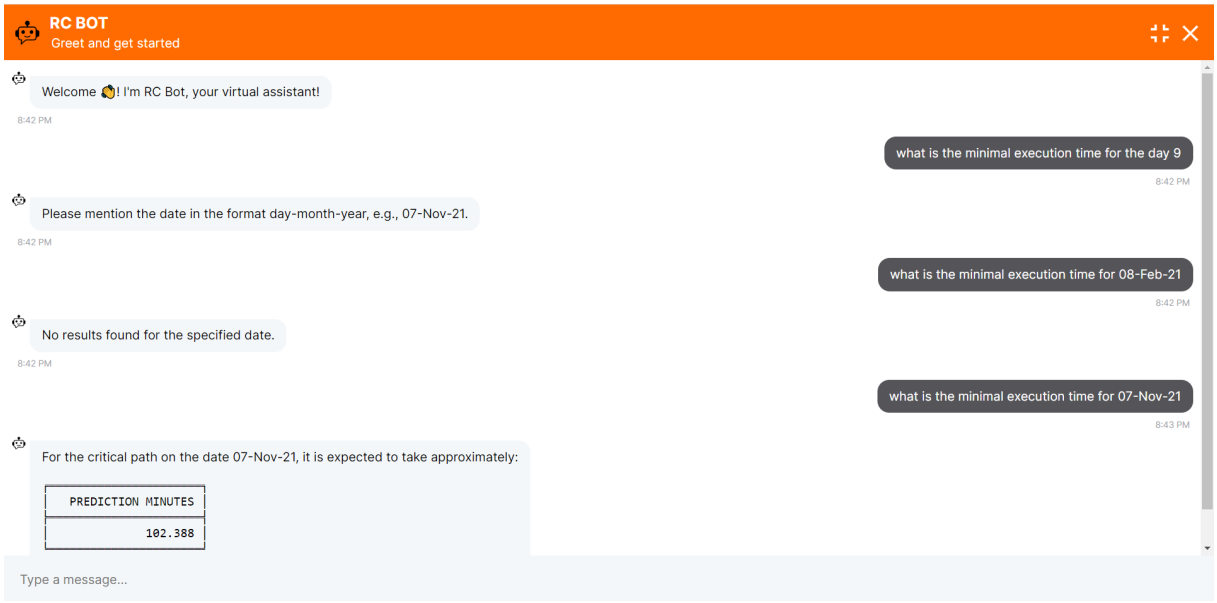


Figure 25: Test critical path minimal execution time date.

Finally, there is an action that allows the user to skip one or more jobs of his choice, with the condition that these jobs be in the submitted or error states; if the user requests to skip a job that is not in one of these two states, an informative message will be displayed. If a job is in one of these states, the user will see a message indicating that the skip to the job was successful. The developed code can be seen in the image below.

Algorithm 5 Skip Job Process

```
1:  class skipJob(Action):
2:  function name(self)
3:      return "action_skip_job"
4:  end function
5:  function run(self, dispatcher, tracker, domain)
6:      language ← tracker.get_slot("Language")
7:      jobName ← tracker.get_slot("jobName")
8:      results, error ← run_query_IPE(SQL query on process instances)
9:      if results is empty then
10:         english_response ← "The mentioned process ID is neither submitted..."
11:     else
12:         if error exists then
13:             dispatcher.utter_message(error)
14:             return
15:         end if
16:         Execute healing command with parameters
17:         english_response ← "The job {jobName} was skipped with success"
18:     end if
19:     if language == "Portuguese" then
20:         translated_responses ← Translate(english_response, to='pt')
21:     else if language == "Spanish" then
22:         translated_responses ← Translate(english_response, to='es')
23:     else
24:         translated_responses ← english_response
25:     end if
26:     dispatcher.utter_message(translated_responses)
27:     return
28: end function
```

The coding approach for all these operations closely resembles what we've discussed earlier. Therefore, I won't delve into the specifics again. I'd just like to emphasize that this particular action stands out from the rest. It's essentially a patch that triggers a function to execute a GET request, which is a type of HTTP request.

The function itself handles the entire process. You simply need to provide it with the necessary arguments, which include the type of correction to be carried out (in this case, a "skip"), the task's name, the date of that specific job, and the thread number, which will always be zero.

Figure 26: Test Skip job.

Chapter 6

Evaluations and Results

Once the chatbot has been created, it is important to evaluate it in terms of performance, and, to do this, various techniques and approaches can be used, such as a manual evaluation which consists of interacting with the chat by asking numerous questions, to test different conversation flows to assess the quality and accuracy of the response.

Another technique applied was automatic evaluation with the Rasa NLU functionality, which allows the efficiency of the NLU model to be classified in terms of precision, recall, and F1 score. Accuracy is the division between correct predictions and total predictions, where the former are made up of true positives and negatives and the total predictions are these plus false positives and negatives. If the division value is equal to 1, all intentions have been successfully identified and no false positives have been detected by the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Figure 27: Accuracy formula.

Recall is the division between the number of true positives (TP) and the sum of the number of true positives (TP) and the number of false negatives (FN). The number of true positives (TP) is the intention that the NLU model has correctly classified as positive. The number of false negatives (FN) is the intention that the model was unable to correctly identify, i.e. if the value of the equation is equal to 1 it means that the model did not identify false negatives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Figure 28: Recall formula.

The division between the multiplication of twice the precision and recall and the sum of the precision and recall. This metric allows for a balanced assessment of the model's performance, i.e. to obtain a high F1 score, both precision and recall need to be high.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 29: F1-score formula.

To carry out this automatic evaluation, you need to run the command, rasa test nlu. During the execution of the command, various pieces of information appear on the console and the most relevant were the accuracy of the intentions, since the NLU model was able to correctly predict all the intentions provided by the user.

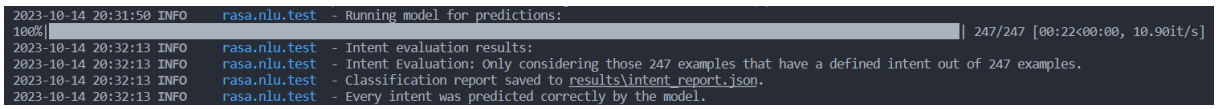


Figure 30: Intent evaluation results.

As we can see in the figure, during the execution of the command a JSON file was generated containing the results of the intentions' ratings. This file is made up of all the intentions that were defined when developing the chatbot and the individual evaluation of each of these intentions, taking into account the metrics explained above. All the intentions had the maximum precision, recall, and f1-score. (Intents JSON file 8.1)

Included in the results folder is a confusion matrix, a tabular representation used to evaluate the performance of a machine-learning model in a classification context. This matrix provides essential metrics, such as true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), which are instrumental in assessing the model's predictive accuracy.

Upon close examination of this matrix, it becomes evident that all values are positive, and notably, there are no occurrences of false positives or false negatives. For example, the first intention "activejobs" was predicted correctly 11 times, and the same goes for the other values in the matrix. This observation signifies that the model has a good performance in its capacity to identify and classify intentions.

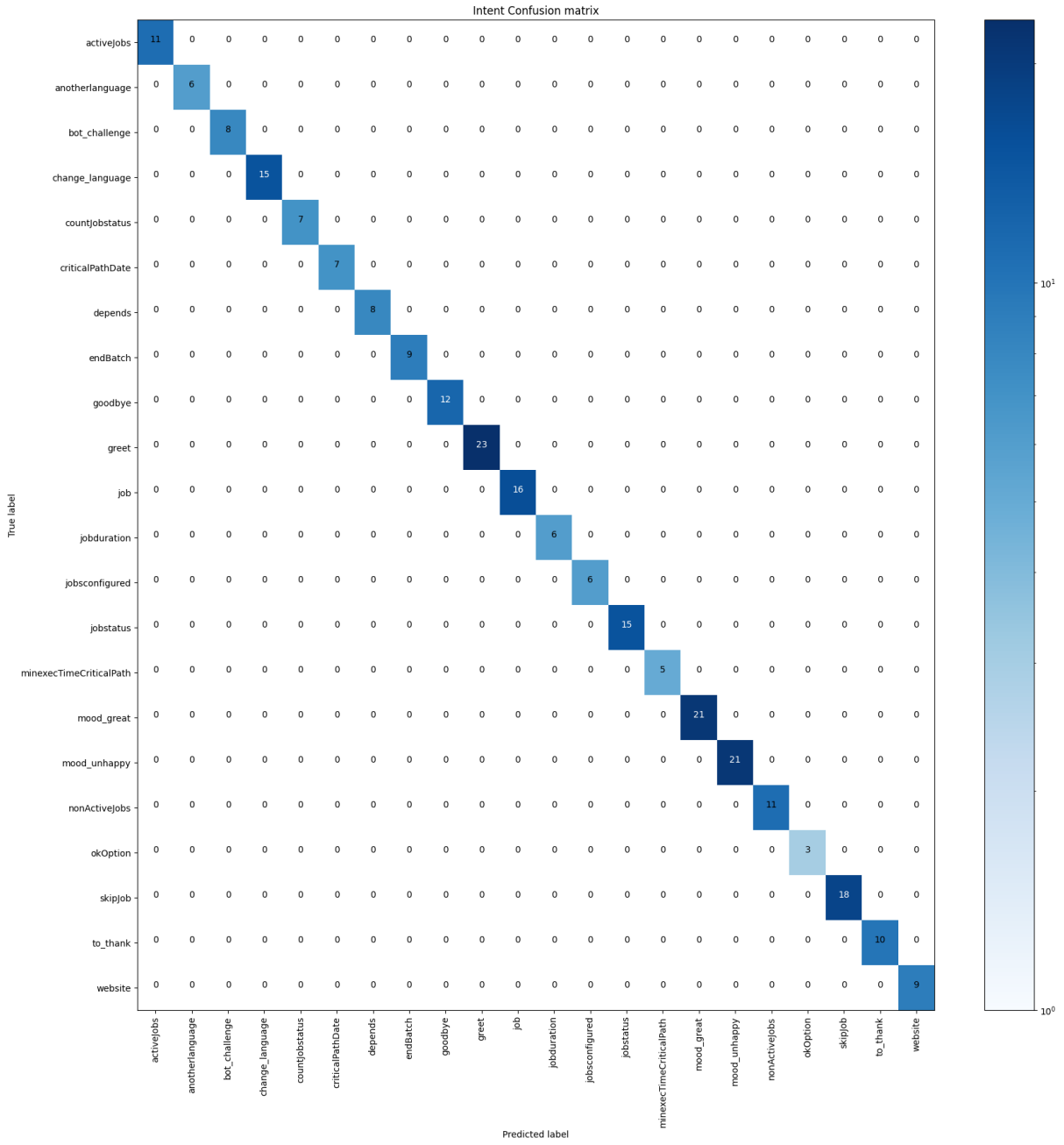


Figure 31: Intents confusion matrix.

The last analysis we can provide is a histogram of intentions which proves once again that we are dealing with a solid NLU model. In this histogram, we can see the direct relationship between the number of samples and the confidence in intentions, which showed good results.

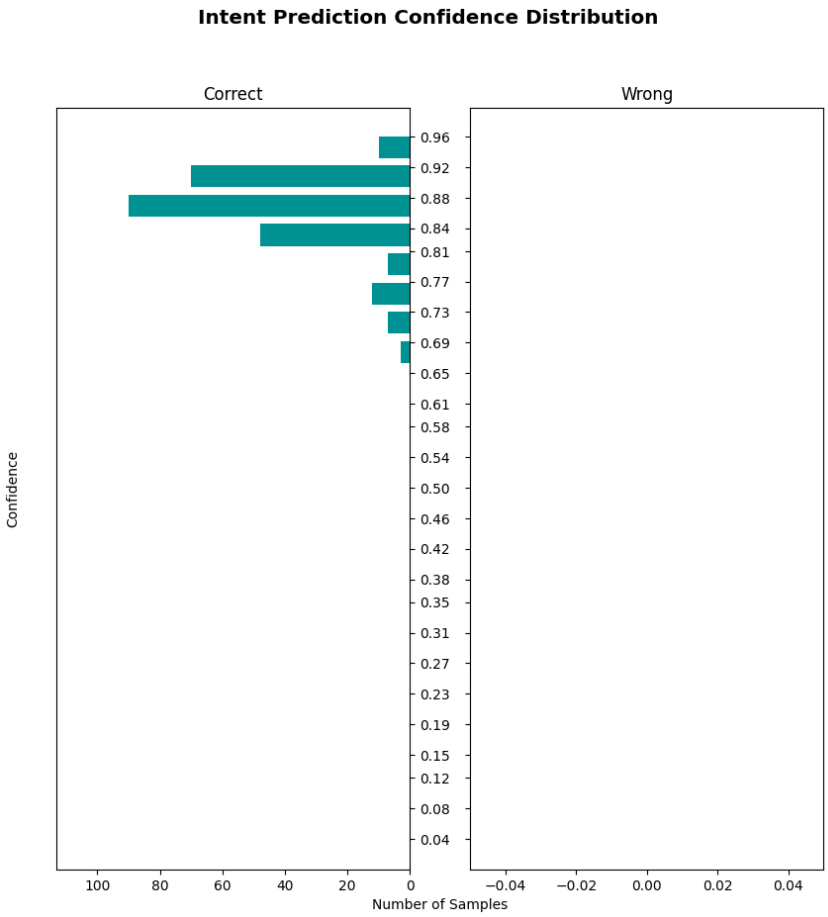


Figure 32: Intent prediction confidence distribution.

In addition, the accuracy of the entities was also evaluated with a positive result, since the chatbot, more precisely the NLU model, accurately identified the specific information that the users provided and as we can see a JSON file was also generated for the entities. (Entities JSON file 8.2)

```
2023-10-14 20:32:16 INFO     rasa.nlu.test  - Entity evaluation results:
2023-10-14 20:32:16 INFO     rasa.nlu.test  - Evaluation for entity extractor: DIETClassifier
2023-10-14 20:32:16 INFO     rasa.nlu.test  - Classification report saved to results\DIETClassifier_report.json.
2023-10-14 20:32:16 INFO     rasa.nlu.test  - Every entity was predicted correctly by the model.
```

Figure 33: Entity predicted results.

The same approach we applied when assessing intentions was also used to analyze entities. This implies that we carried out a series of evaluations, including creating a confusion matrix and generating a histogram representing the distribution of confidence in the entities' predictions.

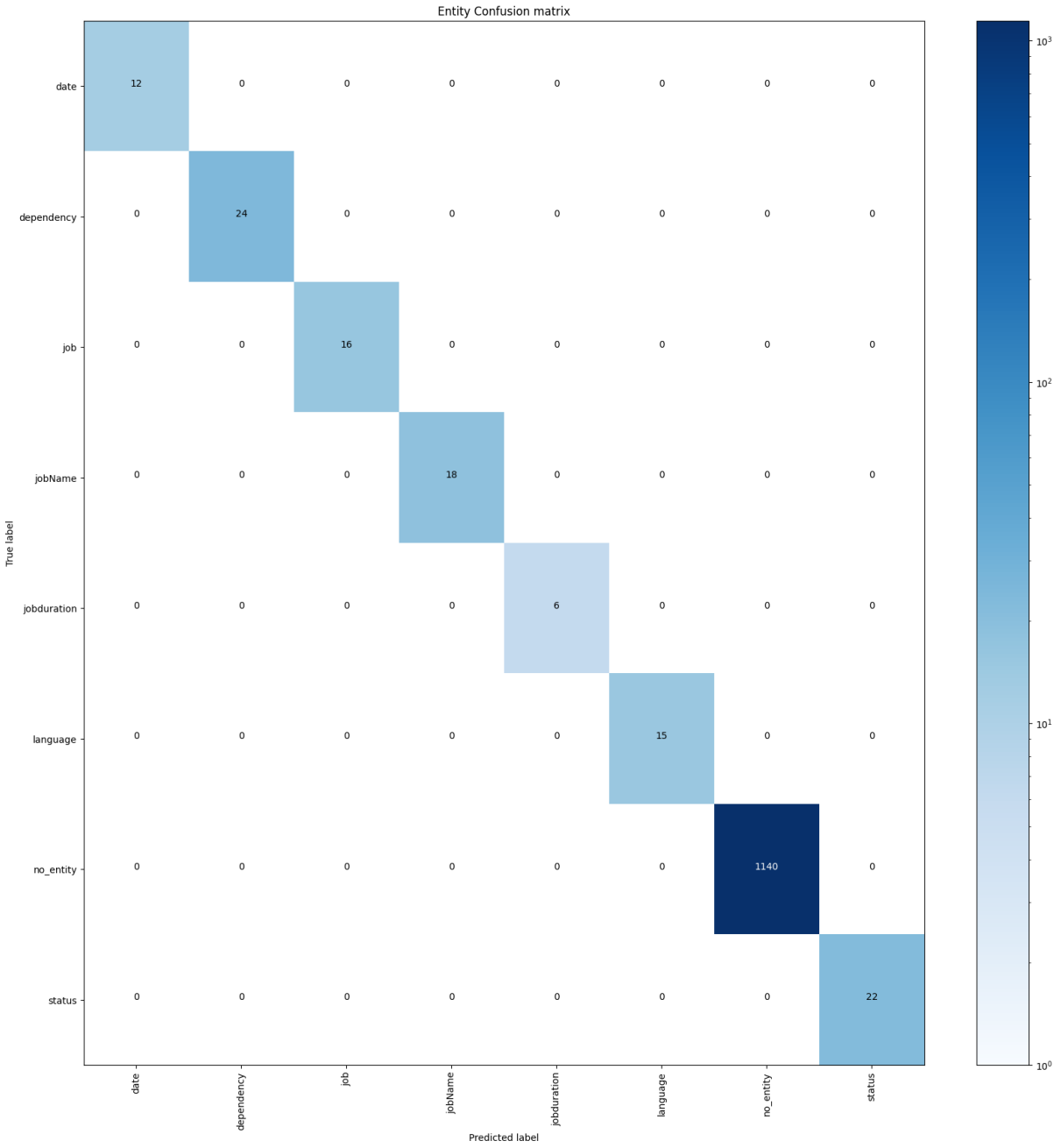


Figure 34: Entity confusion matrix.

Finally, the histogram that has been generated provides a visual representation of the confidence attributed to each entity forecast, taking into account the number of samples and the quality of these forecasts. This helps us to identify trends in the reliability of the forecasts and allows for a more in-depth analysis of the distribution of confidence about the forecasted entities.

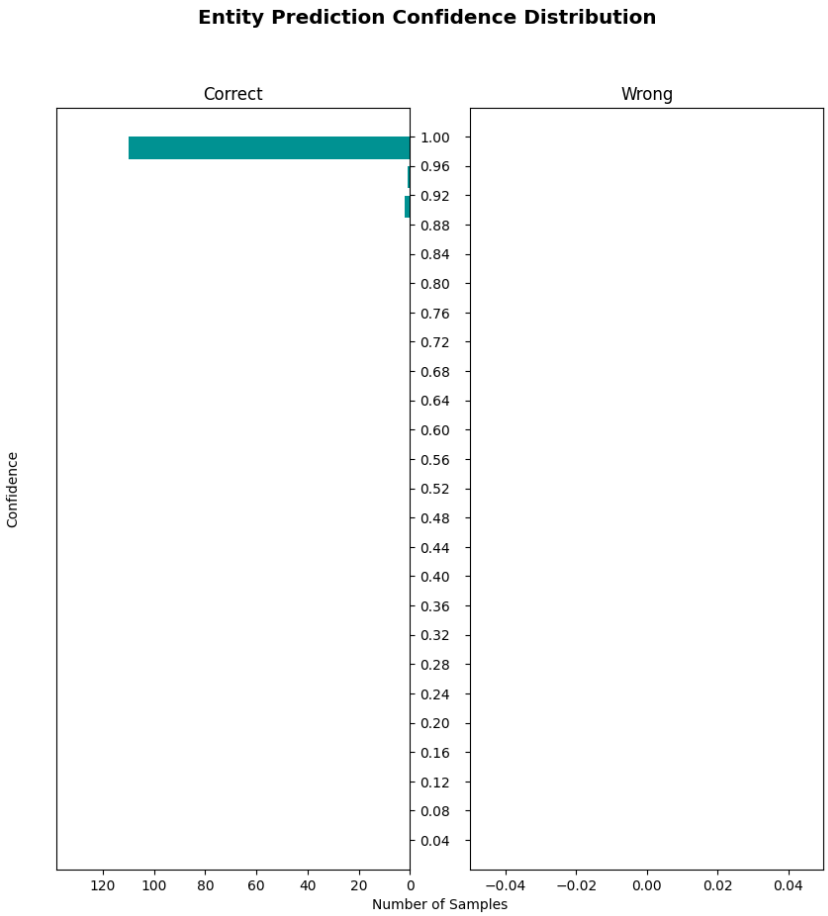


Figure 35: Entity prediction confidence distribution.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Before getting into the specifics of the chatbot's creation and deployment, it is critical to address the constraints encountered both before and during these operations. These constraints were critical in influencing the project's trajectory and outcomes.

During the chatbot's development, it was decided to display the values requested by the user in a list for ease of viewing. Let's suppose the same thing, asking which jobs are active. In this case, some values of these identical jobs will be supplied, but it will also indicate the number of active jobs that exist in the database at the moment. As a result, the user will obtain more information about what he requested, eliminating the need for him to query how many positions are active in the future.

To avoid overloading the user interface and create a more gratifying experience, the list was limited to 10 values. This decision was made to avoid presenting the user with a lengthy list of up to 300 values, which could make the interaction onerous and useless.

It is critical to realize that this option can limit the viewing of all possible values. However, this decision was taken to achieve a compromise between delivering vital information to the user and preserving a clean and easy user interface.

The Rasa chatbot includes a collection of functions that make it an effective and engaging virtual assistant for users. The capacity to detect and reply in different languages, switch languages during a conversation, engage users through natural language processing, answer queries, batch process associated information, and do simple activities such as automated answers and corrections are among the key characteristics.

The chatbot was programmed to recognize and respond to users who spoke Portuguese, Spanish, or English. To do this, a Spacy model was utilized to recognize the language included in the user's message. When the language is detected, the chatbot instructs Deep Translator to translate the text into the chatbot's

primary language. This allows the chatbot to understand and answer effectively regardless of the user's language.

The chatbot's capacity to switch languages throughout a discussion is an intriguing feature. If the user chooses to continue the conversation in a different language, the chatbot rapidly adapts to react in the new language. Users who are multilingual or who want to switch languages to better express themselves will have a more natural and pleasant experience as a result of this.

The bot is intended to mimic the behavior of human assistants, resulting in an engaging conversational experience. To accomplish this, separate lexicons and vocabularies in Portuguese, English, and Spanish were created for each chatbot objective. These lexicons are a set of types of things that users can say to the chatbot during encounters. Natural language processing techniques are used by the system to understand and interpret users' intentions based on these categories and to provide relevant responses.

The conversational system can intelligently respond to user questions and provide batch processing details. The system has access to a large database including important information from the Intelligence Batch Processing System (IBPS) and the Intelligence Processing Engine (IPE) for this purpose. The chatbot employs queries to retrieve specific data from the database, delivering the needed information to the user quickly and accurately.

The chatbot may execute simple tasks such as automatic responses and corrections in addition to answering queries and supplying information in bulk. For example, if a user wishes to skip a specific job, they can use the appropriate syntax to submit a request, and the chatbot will conduct the "skip" action on the desired job.

So, the Rasa-powered chatbot is a solution for multilingual interactions with users. It can adapt to different languages during a conversation, provide an engaging virtual assistant experience through natural language processing, answer users' questions provide relevant information based on trustworthy databases, and perform simple tasks to streamline user interaction. Because of this mix of qualities, the chatbot is a useful tool for a variety of applications, such as customer service, technical assistance, and others. All of the objectives listed above were presented.

However, another main aim of the dissertation was to improve the experience of users who engage with the generated product by acting as a liaison between batch processing and the individual or group of individuals who use it daily. I attempted to provide the chatbot's users with a natural and intuitive interaction by combining the aforementioned elements. I expect that this technology will tremendously benefit *Retail Consult* and help streamline everyday activities and communication.

7.2 Prospect for future work

This dissertation describes how to use the Rasa platform to create a chatbot that can communicate in Portuguese, English, and Spanish. However, in order to widen the scope and usage of the chatbot, we recommend the following subjects for further research and implementation:

One of the most major improvements would be to add Mandarin, German, and French to the chatbot's linguistic skills. In addition to serving *Retail Consult*'s Chinese clients, adding German and French support would allow for a more direct and effective engagement with customers in German and French-speaking nations, increasing the chatbot's reach and making it into a truly global tool.

To improve the chatbot's language capability, the lexicon and terminology employed should be expanded. By adding more terms and synonyms to the chatbot's knowledge base, which is contained in the natural language understanding (NLU) file, it will be able to understand and respond to a broader range of expressions and questions asked, making the chatbot more powerful and reliable tool for user interactions.

The chatbot currently offers a capability that allows the user to bypass specific tasks contained in the Intelligent Batch Processing System (IBPS) database. However, it would be fascinating to provide the option to start or stop certain jobs, which would be conducted by HTTP requests, as was the skip, allowing users to further control batch processing activities.

To facilitate the scalability and deployment of the chatbot in diverse contexts, all produced code should be packaged in Docker containers. It is suggested that three containers be used: one to run the Rasa actions server and install the required dependencies, another to run the React application through npm, and a third to run Rasa with the API enabled. This strategy will result in a simpler and more efficient chatbot deployment, mostly by avoiding dependence issues on other machines.

To improve the user experience, more actions that implement various inquiries and capabilities related to the chatbot environment can be added, offering more accurate and full replies to a wide range of questions and requests.

The incorporation of pagination functionality is a tempting area for advancement. To avoid undue strain on the user interface, the chatbot currently displays a short list of ten values. A smart improvement would be to include a "Show More" button, allowing consumers to access other options. Users can properly assess the values by wisely using paging, avoiding the unnecessary imposition of an exhaustive list all at once. This technique fosters a greater sense of flexibility and personalization, effectively catering to a wide range of user preferences and needs.

Furthermore, if we want to improve the system, one of the aspects that we will take into account

will be the user's expectations from the system and interactions and the quality of the chatbot in terms of appropriateness of language level, continuity of conversation, and success in task performance de Trabalho [2022].

By adopting these future works, the Rasa chatbot will become an even more valuable tool for *Retail Consult* and its users, improving the experience by increasing interaction efficiency and expanding its worldwide reach. Each of these enhancements will add to a more smart, adaptable, and valuable chatbot that will satisfy the demands of an ever-changing market.

Bibliography

Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 373–383. Springer, 2020.

Helena Isabel Oliveira Alves. *Estudo e implementação de modelos de gestão de stocks em sistemas ERP para empresas retalhistas*. PhD thesis, 2015.

Mattia Atzeni and Maurizio Atzori. Askco: A multi-language and extensible smart virtual assistant. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 111–112. IEEE, 2019.

Nicolas Audebert, Catherine Herold, Kuider Slimani, and Cédric Vidal. Multimodal deep networks for text and image-based document classification. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 427–443. Springer, 2020.

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*, 2017.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S Lam. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, pages 341–350, 2017.

Davide Cucurnia, Nikolai Rozanov, Irene Sucameli, Augusto Ciuffoletti, and Maria Simi. Matilda-multi-annotator multi-language interactivelight-weight dialogue annotator. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 32–39, 2021.

Plano de Trabalho. Calendarização e objetivos plano de trabalho. https://dcmei.di.uminho.pt/web#id=588&model=dissertation_admission.dissertation&view_type=form&cids=1&menu_id=100, 2022.

Equipe editorial de Conceito.de. Retalhista - o que é, conceito e definição. https://conceito.de/retalhista, 2020. accessed 10.01.2023.

Hugo Faria, Maria Araújo Barbosa, Bruno Veloso, Francisco S Marcondes, Celso Lima, Dalila Durães, and Paulo Novais. Edubot: A proof-of-concept for a high school motivational agent. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 223–232. Springer, 2022.

Felipe Ramos Feitoza et al. Cecílio: um chatbot para automação do atendimento aos usuários em instituições federais de ensino superior. 2021.

Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

Marie-Claire Jenkins, Richard Churchill, Stephen Cox, and Dan Smith. Analysis of user interaction with service oriented chatbot systems. In *International conference on human-computer interaction*, pages 76–83. Springer, 2007.

Veton Kepuska and Gamal Bohouta. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, pages 99–103. IEEE, 2018.

Neiva Larisane Kuyven, Carlos André Antunes, Vinicius João de Barros Vanzin, João Luis Tavares da Silva, Aliane Loureiro Krassmann, and Liane Margarida Rockenbach Tarouco. Chatbots na educação: uma revisão sistemática da literatura. *RENOTE*, 16(1), 2018.

Francisca Leitão Gonçalves do Vale Lima. Big data warehousing em tempo real: da recolha ao processamento de dados, 2017.

Trang Nguyen Thi Mai and Shcherbakov Maxim. Enhancing rasa nlu model for vietnamese chatbot. *International Journal of Open Information Technologies*, 9(1):31–36, 2021.

Martech. Pros and cons future facebook chatbots. https://martech.org/pros-cons-future-facebook-chatbots/, 2022. accessed: 02.11.2022.

Medium. Advantages and disadvantges of siri. https://macbook.medium.com/the-advantages-and-disadvantages-of-siri-1cc83604a08f, 2022. accessed: 07.12.2022.

Microsoft. What can you do with cortana in windows. https://support.microsoft.com/en-us/topic/what-can-you-do-with-cortana-in-windows-f57ef46f-716a-9940-a8fc-09b3433d05ea, 2022. accessed: 26.10.2022.

MyAyan. Advantages and disadvantages of amazon alexa. https://www.myayan.com/advantages-and-disadvantages-of-amazon-alexa, 2022. accessed: 25.10.2022.

Eduardo Nacimiento-García, Carina S González-González, and Francisco L Gutiérrez-Vela. Free software virtual assistants for designing pervasive gaming experiences to promote active aging: State of the art. *CEUR-WS. org*, 2747, 2020.

S Nithuna and CA Laseena. Review on implementation techniques of chatbot. In *2020 International Conference on Communication and Signal Processing (ICCSP)*, pages 0157–0161. IEEE, 2020.

Oracle. Batch schedule. https://docs.oracle.com/cd/E79623_01/rms/pdf/16021/html/batch_schedule/merch-batch.htm, 2017. accessed: 17.12.2022.

Python Organization. Deep translator from google translator. https://pypi.org/project/deep-translator/, 2023. accessed: 15.03.2023.

PandaSecurity. Cortana security flaw. https://www.pandasecurity.com/en/mediacenter/mobile-news/cortana-security-flaw/, 2022. accessed: 25.10.2022.

PeopleBank. The pros and cons of google home. https://www.peoplebank.com.au/blog/2018/04/the-pros-and-cons-of-google-home?source=google.com, 2018. accessed: 05.12.2022.

Bhavika R Ranoliya, Nidhi Raghuwanshi, and Sanjay Singh. Chatbot for university related faqs. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1525–1530. IEEE, 2017.

RASA. Conversational ai with rasa: Introduction to rasa. https://www.youtube.com/watch?v=Ap62n_YAVZ8&t=3s, 2021. accessed: 07.3.2023.

RASA. Conversational ai with rasa: Introduction to rasa. https://rasa.com/, 2023a. accessed: 07.03.2023.

RASA. Rasa installations procedures. https://www.youtube.com/watch?v=RVoFqxmG8p0, 2023b. accessed: 12.07.2023.

RetailConsult. Oracle retail consult. https://www.retail-consult.com, 2023. accessed: 16.01.2023.

Reuters. Us apple siri. https://www.reuters.com/article/us-apple-siri-idUSKBN16G0H3, 2022. accessed: 18.11.2022.

Sakshi Sachdeva. Scrum methodology. *Int. J. Eng. Comput. Sci*, 5(16792):16792–16800, 2016.

Andressa Vieira Silva. *Um modelo de classificação para o Reconhecimento de Entidades Nomeadas*. PhD thesis, Universidade de São Paulo, 2020.

Abhishek Singh, Karthik Ramasubramanian, and Shrey Shivam. Introduction to microsoft bot, rasa, and google dialogflow. In *Building an Enterprise Chatbot*, pages 281–302. Springer, 2019.

Pavel Smutny and Petra Schreiberova. Chatbots for learning: A review of educational chatbots for the facebook messenger. *Computers & Education*, 151:103862, 2020.

Marcin Sowański and Artur Janicki. Leyzer: A dataset for multilingual virtual assistants. In *International Conference on Text, Speech, and Dialogue*, pages 477–486. Springer, 2020.

SpaCy. Industrial-strength natural language processing. https://spacy.io/, 2023. accessed: 22.04.2023.

Spring. Spring batch documentation. https://docs.spring.io/spring-batch/docs/current/reference/html/index-single.html, 2023. accessed: 24.03.2023.

Techtarget. Named entity recognition - ner. https://www.techtarget.com/whatis/definition/named-entity-recognition-NER, 2023. accessed: 02.03.2023.

U THIRUPALU and M SURESH. Virtual personal assistant using artificial intelligence. *Journal of Engineering Sciences*, 14(08), 2023.

Wikipedia. Batch processing. https://en.wikipedia.org/wiki/Batch_processing, 2023. accessed: 13.11.2022.

Chapter 8
Listings

8.1 Attached 1

Listing 8.1: Classification intents report


```
{
  "skipJob": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 18,
    "confused_with": {}
  },
  "criticalPathDate": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 7,
    "confused_with": {}
  },
  "jobstatus": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 15,
    "confused_with": {}
  },
  "website": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 9,
    "confused_with": {}
  },
  "mood_unhappy": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 21,
```

```
    "confused_with": {}
  },
  "nonActiveJobs": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 11,
    "confused_with": {}
  },
  "to_thank": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 10,
    "confused_with": {}
  },
  "depends": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "job": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 16,
    "confused_with": {}
  },
  "bot_challenge": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "anotherlanguage": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 6,
    "confused_with": {}
  },
  "change_language": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 15,
    "confused_with": {}
  },
```

```
  "jobsconfigured": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 6,
    "confused_with": {}
  },
  "endBatch": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 9,
    "confused_with": {}
  },
  "activeJobs": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 11,
    "confused_with": {}
  },
  "okOption": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 3,
    "confused_with": {}
  },
  "greet": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 23,
    "confused_with": {}
  },
  "minexecTimeCriticalPath": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 5,
    "confused_with": {}
  },
  "goodbye": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 12,
    "confused_with": {}
  },
  "mood_great": {
    "precision": 1.0,
```

```
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 21,
    "confused_with": {}
  },
  "countJobstatus": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 7,
    "confused_with": {}
  },
  "jobduration": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 6,
    "confused_with": {}
  },
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 247
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 247
  }
}
```


8.2 Attached 2

Listing 8.2: Classification entities report


```
    {
  "status": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 22,
    "confused_with": {}
  },
  "date": {
    "precision": 1.0,
```

```
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 12,
    "confused_with": {}
  },
  "dependency": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 24,
    "confused_with": {}
  },
  "jobName": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 18,
    "confused_with": {}
  },
  "jobduration": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 6,
    "confused_with": {}
  },
  "job": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 16,
    "confused_with": {}
  },
  "language": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 15,
    "confused_with": {}
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 113
  },
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 113
```

```
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 113
  }
}
```