Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Diploma Thesis

Cognitive RPA: an open source solution for user support

Alket Shabani

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

MSc. Alket Shabani

Systems Engineering and Informatics
Informatics

Thesis title

Cognitive RPA: an open source solution for user support

Objectives of thesis

The main objective of the thesis is to implement a cognitive RPA solution for user support in multiple information systems in an organization.

The partial goals of the thesis are such as following:
- to provide a current overview of state of the art of RPA cognitive computing and to identify main issues with RPA implementations in user support processes;
- to design an optimal open source software solution for user support in multiple systems in an organization;
- to evaluate the proposed solution and make recommendations.

Methodology

The methodology of this work is based on reviewing current literature on cognitive computing and RPA. After that, an open source software will be selected according to the given criteria. In the practical part we will try to design and implement full automation of the user support process based on cognitive computing and an RPA bot by using the combination of selected software. By synthesising and contrasting findings in the literature review and practical part, final recommendations will be made and conclusion will be formulated.

The proposed extent of the thesis

60 – 80 pages

Keywords

RPA, Cognitive Automation, Cognitive Computing, Open Source, user support, chat bot.

Recommended information sources

AGUIRRE, Santiago; RODRIGUEZ, Alejandro. Automation of business process using robotic process automation (rpa): A case study. In: Workshop on Engineering Applications . Springer, Cham, 2017. pp. 65-71.

BONACCORSI, Andrea; ROSSI, Cristina. Why open source software can succeed. Research policy, 2003, 32.7: 1243-1258.

KELLY III, John E.; HAMM, Steve. Smart machines: IBM's Watson and the era of cognitive computing. Columbia University Press, 2013.

LACITY, Mary C.; WILLCOCKS, Leslie P. A new approach to automating services. MIT Sloan Management Review, 2017.

LERNER, Josh; TIROLE, Jean. The open source movement: Key research questions. European economic review, 2001, 45.4-6: 819-826.

MADAKAM, Somayya; HOLMUKHE, Rajesh M.; JAISWAL, Durgesh Kumar. The Future Digital Work Force: Robotic Process Automation (RPA). JISTEM-Journal of Information Systems and Technology Management, 2019, 16.

POOSAPATI, Vijaya Ramaraju; KATNENI, Vedavathi; MANDA, Vijaya Killu. Cognitive Automation in Industry–Design Principles and Case Study. International Journal of Applied Engineering Research, 2018, 13.20: 14827-14831.

STALLMAN, Richard. Viewpoint Why open source misses the point of free software. Communications of the ACM, 2009, 52.6: 31-33.

WILLCOCKS, Leslie P.; LACITY, Mary; CRAIG, Andrew. The IT function and robotic process automation. 2015.

Expected date of thesis defence

2019/20 SS – FEM

The Diploma Thesis Supervisor

Ing. Miloš Ulman, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 26. 8. 2019

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 14. 10. 2019

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 05. 04. 2020

Declaration

I declare that I have worked on my diploma thesis titled "Cognitive RPA: an open source solution for user support" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 05/04/2020 _____

Acknowledgement

I would like to thank **Ing. Milos Ulman, Ph.D.** and my **family** for the support throughout this project.

Cognitive RPA: an open source solution for user support

Abstract

In today's technological world every day, new technologies emerge. Robotic process automation is one of the technologies that empowers companies in automating their processes. On the other hand, is up to the organizations to find which processes are best to automate. What empowers RPA even more is the cooperation with Artificial Intelligence in creating cognitive applications. This makes RPA more flexible and more extensible.

The aim of this thesis is to combine the power of these two technologies in providing an end to end automation. The best part is that we are trying to build this based on Open Source tools. A lot of research will be needed to find the tools available and to integrate two of them in automating a user support process.

This study aims to provide an insight of the tools available from open source community and their capabilities. What is more, the implementation can be extended in other fields or more processes.

Keywords: Open source software, robotic process automation, cognitive computing, cognitive automation, TagUI, Rasa NLU, natural language understanding

Table of content

List of pictures

List of tables

List of abbreviations

- RPA – Robotic Process Automation
- AI – Artificial Intelligence
- ERP – Enterprise Resource Planning
- CRM – Customer Relationship Management
- FOSS – Free and Open Source Software
- GNU – GNU's Not Unix
- NLU – Natural Language Understanding
- NLP – Natural Language Processing
- ITIC – Information Technology Intelligence Consulting
- OS – Operating System

1 Introduction

Repetitive business tasks like copying, pasting, merging and moving are part of most of companies' employees especially those who work with ERP, CRM and Excel.

Some of these tasks are highly structured and manual so they could be handled by a robot and the worker would have valuable time to take on some more valuable tasks. Automating these processes is what RPA does best.

Working with structured data is a process that RPA companies seem to have improved more and more the recent years. Recent case studies have been carried to show the benefits of RPA in different business processes and there are also a number of actors who carried these studies that have some tips on what processes are best to automate.

On the other hand, nowadays, RPA has a new partner AI (Artificial Intelligence) to form cognitive automation. This "partnership" AI+RPA is still in its early days, but companies seem to put more and more focus on this direction.

What is to be mentioned is that RPA is not based on classic idea of robots which are hardware devices that are programmed to do specific tasks. RPA is based solely on software applications which are programmed to do work which previously was done by people.

2 Objectives and Methodology

2.1 Objectives

This thesis aims to firstly give an overview of how Robotic Process Automation and cognitive computing can work together. Many software companies are working in this direction and we will provide some case studies where these solutions have been applied. This not only works as a general idea of how RPA has been merged with cognitive computing but also as a learning path for other companies trying to figure out which processes to automate.

Based on this a high-level architecture of how these two components are linked together will be provided.

In the technology world, RPA is one of the most hyped words in 2019 and so is cognitive computing but we want to give some basic information regarding the two, so the readers can be more familiarized and to better understand the specific technologies.

Most of the companies providing RPA solutions or RPA together with cognitive computing to form intelligent automation have their released software proprietary so our goal is to do thorough research and to find some open source alternatives. The open source alternatives should be on both RPA and cognitive automation. Also, the research that we will make will be necessary for our final objective in this thesis which is to create an open source solution of intelligent automation.

Our last but not least objective will be to evaluate an automated process comprised mostly of open source software which includes RPA and cognitive computing. Also, we will make recommendations based on our practical part.

2.2 Methodology

The first section of this thesis will be literature review where we define some key concepts like RPA, open source and cognitive computing.
There will be a comparison of tools that we can use in our thesis and finally we will explain why we chose the tool that will be used for the practical part.
Through literature review we want to let the reader know what we will do in this thesis and explain the basic concepts that will be further used in this work.
After we are familiar with the concepts there will be some introduction to our core technologies that will be used in the practical part. Before we discuss about the practical part some key tips of dos and don’ts will be provided. Also, a general architectural overview of how we think that RPA and cognitive services can be used will be further explained in our work.
The final part of this thesis will be the practical part which is web scraping of images from a website with RPA tool and then using cognitive computing to find which of these images are those of human.

3 Literature Review

The literature review section will cover the necessary information that will help in practical part. The foundation of open source software will be covered, starting with a general information about this initiative and then covering areas such as why choose open source and comparison of open source tools with commercial ones. Furthermore, an introduction to RPA and intelligent automation will be covered, including open source RPA tools. The impact of robotic process automation in area of user support will be covered in the literature review and will help us understand the need or not for RPA in user support systems.

3.1 Open Source

Open source has attracted a substantial attention in the recent years [4]. Whether open source is indeed 'faster, cheaper, better' is still an object of controversy, but there is little doubt that open source projects have been tremendously successful in recent years.[4]. But what is open source software? Roughly, being open source requires that the source code, and not only the object code (the sequence of 1's and 0's that computers actually use), be made available to everyone, and that the modifications made by its users also be turned back to the community. The details vary with the license adopted for the program. Some of the key criteria included in the Open Source Definition are (a) the royalty free redistribution of the program, (b) the release of the source code, and (c) the requirement that all modifications be distributed under the same terms as the license of the original software [1].

What open source includes and what is the most important among its features is free software and by free software we do not mean pricing but like, "free speech," not "free beer" [2].

3.1.1 History of Open Source

Open source dates back to the year 1970 when Richard M. Stallman created some code for the printer in order not to get stuck. When MIT was equipped with a new printer the source code was not available for change, so this shook Stallman and his culture as a hacker (back in the days it had positive meaning). During this year's selling licenses was the main profit the companies had, and Stallman came up with the idea of creating a new operating system which would not deprive users from freedom of how the OS worked and the users could change it based on their preferences. This marked the start of open software movement.

The open software movement started with what was then called by Stallman GNU (pronounced "guh-NEW" in this case) and this was his operating system.

Based on Stallman definition the open source software should have these four freedoms:

- The user has the freedom and the rights to run the program for any specific purpose
- The user can change the source code of the software application in the way that he finds suitable
- Also, the user has the right to redistribute copies for free
- The user has the right to redistribute modified copies of the software so the community can benefit from the changes

The problem that was encountered by Stallman was linked with the release of the software. He could not just release the software because someone could patent it on his name. And the other problem was that he did not want to close the code so the whole purpose of open source software and user reviews would be lost.

These two issues that Stallman faced led to another breakthrough for the open source software and this was GNU General Public License which was released in 1989 and encapsulated the concept of copyleft. Copyleft means you can use the software as you wish, you can also distribute and modify it but if you modify this should be shared with the community so everyone can benefit.

The birth of Linux marked another significant moment in open source history. Until the 90's the missing part from the open source operating system was the kernel and in 1991 Linus Torvalds released the Linux kernel version 0.02. Torvalds was already integrating some of GNU components into its kernel and Stallman says that Linux should be called GNU-Linux.

What Torvalds wanted to emphasize about Linux is that its power lies in the community and the code itself.

There are several arguments on why a company, or an individual should choose to use FOSS (free and open source software). We will try to list several features that we consider to be useful whether to choose or not an open source software.

3.1.2 Reliability

The first feature that we will list is **reliability (ISO 25010: Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.)** as it is one of the core qualities that not just a software but everything should have. according to a Fuzz study in a paper called "Fuzz Revised" characteristic as crashing and freeze-ups of software were tested and it turned out that FOSS applications are more reliable [3].

It is also interesting to compare results of testing the commercial systems to the results from testing “freeware” GNU and Linux. The seven commercial systems in the 1995 study have an average failure rate of 23%, while Linux has a failure rate of 9% and the GNU utilities have a failure rate of only 6%.

Also if we refer to most recent figures as described in the annually ITIC report the most reliable enterprise server OS: IBM Power Systems, Lenovo System x and ThinkSystem, and Huawei KunLun servers running ***Linux*** and ***open source distributions*** each delivered the least amount of unplanned downtime associated with system or OS technical problems, less than 12 minutes per server/per annum [4]. Taking into consideration these figures we can see an ongoing domination and consistency of the open source software.

It is reasonable to ask why a globally scattered group of programmers, with no formal testing support or software engineering standards can produce code that is more reliable (at least, by our measure) than commercially produced code. Even if you consider only the utilities

that were available from GNU or Linux, the failure rates for these two systems are better than the other systems.

What is more 80% of top 10 most reliable hosting providers ran open source software in their systems. This is pointed out in a survey made by Netcraft [5]which found out that out of 10 hosting providers 4 used Linux 4 used FreeBSD and only 2 proprietary software like Microsoft Windows.

3.1.3 Scalability

After reliability what we think that is necessary to take into consideration when a software is chosen is to know how it can **scale** with your infrastructure and your business needs. There rare data gathered from Forbes [6]about the top 500 computers in the world and if they are running on open source software or proprietary one. The results are that 391 of them are running on Linux 189 on Unix and 2 on FreeBSD. This gives an insight that FOSS can run either on normal hardware but also drive the supercomputers of the world.

What most of companies and people can do is to start small by using FOSS and then scale as per their needs.

3.1.4 Security

Security is the third feature that will be covered, and needless to say it is one of the most important ones. In today world where the information and data are so important, we cannot leave without mentioning the security of FOSS and how open source software compares to proprietary one. This part is difficult to measure in a quantitative way, but we will do our best to give some arguments.

We believe open source software is a necessary requirement to build systems that are more secure.... opening the source of existing systems will at first increase their exposure... However, this exposure (and the associated risk of using the system) can now be determined publicly. With closed source systems the perceived exposure may appear to be low, while the actual exposure... may be much higher. Moreover, because the source is open... the period of increased exposure is short. In the long run, openness of the source will increase its security... [and] it allows users to make a more informed choice about the security of a system....” [7]
In the following sections will be quantitative studies that back the idea that FOSS offers better security at least at some areas:
[8]In this paper it is pointed out that OSS and proprietary software did not differ in security. And thus, we can say that the security of open source is not a myth. What is interesting to point out is that the OSS vulnerabilities in terms of severity tended to be less severe.
"When we determine the medians of medians of open source software and closed source software and also the corresponding medians of the proportions of highly severe vulnerabilities (30.28% and 45.95%, respectively), the first impression is that open source software is more secure in terms of the severity level. However, applying statistical analysis (Mann-Whitney U-test) on the medians, no statistically significant differences can be found: the two-tailed test provides a high number for p ($p=0.11$). Applying the same test to the

proportion figures, the test, again, does not indicate that the samples are significantly different at the 0.05 level (p=0.06)" [8].

3.1.5 Total cost of ownership

We think that when it comes to choosing a product especially a software product total cost of ownership is an important criterion. It is important because it doesn't matter if you start free with a software what is important is the cost in the long run.
There are a number of factors to be considered when calculating TCO and all these factors should be identified and in the end a cost calculation is necessary. Some "hidden" costs should be taken into account such as technical support, upgrade costs, end user costs and more. [3] Identifies the TCO factors that according to them are necessary to evaluate with FOSS and proprietary software: hardware costs, direct costs of software, indirect costs, staffing costs, support and downtime.
Sometimes it can be misleading when talking about TCO and proprietary software as the end user does not own the software rather than he has the right to use it based on a "lease".

3.1.6 Community

One of the things that people who create open source software pay attention to is having a user base and not for the purpose of proving that the things that you have done are good and useful but most of them are hackers as well and so they can become co-developers. Since the source code is available most of them will try it and probably will find some bugs in your code or give you valuable feedback on how some things could have been done better.

3.2 Open source RPA tools

In this section we will focus on the tools that are available under open source licenses. We will go through the details of some of the tools and finally one of the tools will be used for the practical purposes of this thesis.

The truth is that there are no too much open source RPA tools and this is mainly because robotic process automation is not consolidated properly in the market. RPA is still a growing industry with its benefits and drawbacks at the same time which are to be resolved in the future.

3.2.1 Robot Framework

The first tool that we will go through is Robot Framework which is an open source tool licensed under the Apache license 2.0. Robot framework is an RPA tool which is extensible giving it the possibility to be integrated with a whole number of other tools. Moreover, it is also used for acceptance testing and acceptance test driven development (ATDD) so the user base is not just strict to RPA usage.

Robot framework first was developed by Nokia Networks to be then sponsored by Robot Framework Foundation. The foundation which is a non-profit organization has its main goal

to provide the growth of Robot Framework and it is backed up by a number of companies that have the same scope.

What robot framework offers as stated before is its extensibility and the community can use either Python programming language or Java. The Robot Framework community develops and maintains a number of libraries that provide functionality such as optical image recognition, HTTP APIs, iOS and Android application support, database access, and remote execution. The list of libraries can be found on the GitHub page or their own website.

Another key advantage of Robot Framework is that it is operating system and application independent. In this way you are not bound to specific OS for running the tool.

3.2.2 Roro

Another open source tool that is free to use for RPA purposes is called RORO [9]. It is created by Arvie Delegado in collaboration with two other contributors. The main idea was from Arvie and the two other contributors joined to add some extra functionalities like add-in for Fody (another open source project) and the compiler in .NET, since Roro is made to run only on Windows operating system.

3.2.3 Open RPA

Another tool [10] included in our list is country of origin of which is Denmark. It is not difficult to tell that open RPA is open source. The code is publicly available in GitHub under the profile open-rpa/openrpa/. Open RPA is available on windows machines and automates a number of tasks. As with other RPA tools OpenRPA offers the possibility to record the user interactions with windows application and then can be replicated. OpenRPA has an UI that the users can use in order to program their robot. There are predefined building blocks that can be used to create certain tasks. Below we will introduce some of the tasks that OpenRPA offers to its users.

As we mentioned before one of the tasks that most RPA tools offer is to record the steps that users do in their desktop. OpenRPA adds the step to the tool which will show the steps one by one related to the designers matched.

What are more basic activities such as working with workbooks, downloading and uploading files, working with images using open source image processing libraries such as OpenCV.

Since OpenRPA is open source as the other tools in this section addition of extra code or combine it with other tools to extend the possibilities of the tool.

3.2.4 Automagica

Automagica [11] is the next tool in our open source RPA list. It claims to be not only robotic process automation but smart RPA, meaning that Automagica has prebuilt cognitive abilities.
The activities that Automagica support are numerous. We will go through a list of them to show the capabilities of the tool.

It comes without saying that the tool supports desktop automation by interacting with applications on users' desktop. Also, possibility to automate business applications such as SAP, Oracle, Salesforce is included. Business applications are the ones that take much of manual work in numerous companies so Automagica solves that issue pretty well. Web automation is part of what the tool offers to its community including tasks such as web scraping and website management without any user interaction. OCR or optical character recognition makes for the smart part in their claim. Also, computer vision is used in order to be able to process images on the screen and to offer the best user experience.
Moreover, what makes Automagica special is the support for Citrix automation because Citrix runs as a separate desktop service in the user's computer and adds and extra layer of abstraction to the automation process.
In order to support the claims of Automagica we need also to verify the parts comprising this tool. In this way the users can get also an overview of the range of offerings that

3.2.5 TagUI

TagUI [12] is another open source tool that offers RPA capabilities. At first TagUI was developed by Ken Soh then it was backed by AI Singapore. AI Singapore is a government funded initiative that is focused on building local artificial intelligence capabilities.

What TagUI offers is automating cognitive actions that users do in their desktops or web browsers. This is done by integrating a number of tools which as well are open source. The tools that TagUI integrates are:

- SikuliX - Raimund Hocke from Germany [13]
- CasperJS - Nicolas Perriault from France
- PhantomJS - Ariya Hidayat from Indonesia [14]
- SlimerJS - Laurent Jouanneau from France [15]

Another feature that TagUI offers to its users/customers is the possibility to write "code" in natural like language, meaning that there is no programming required. It offers JavaScript as well for advanced users and extended code creation. As far as language-like syntax is concerned support is available for 20 languages.

As with other RPA tools in this section TagUI offers the feature of cross OS, meaning that the tools can be use in Windows, MacOS and Linux. For the data science users TagUI brings Python and R integration.

3.2.6 Taskt

Formerly known as sharpRPA, Taskt [16] is the last tool in our list of RPA tools from the open source community. It is based on .NET framework, so it means it is not a cross platform tool, but it works only in Windows OS which supports .NET framework. Browser and desktop automation are the main things that Taskt does well.

3.3 Robotic Process Automation

When people hear about robots immediately comes in mind the physical robot doing some tasks, but in the case of RPA there is just a software robot. When it comes to business processes with RPA we mean configuring and programming the software 'robot' to perform some task that previously was carried out by people [17].
Another definition by [18] is "RPA is an umbrella term for tools that operate on the user interface of other computer systems in the way a human would do."
What RPA tends to automate are tasks that people do on a daily basis which are highly structured, time consuming and include manual tasks like copy pasting, moving data from one platform to another. [18]
When it comes to the IT side the RPA bot is equal to one software license when it comes to vendors that sell their software. The RPA bots have the ability to communicate and interact with third party software in our IT systems via front-end in contrary with traditional software that tend to communicate in the backend. This makes it possible to integrate RPA with almost every other software that can be used by a human being [19].
There are also some other features that distinguish RPA from traditional software like business management processes. These features include:

- The way that RPA accesses the third-party platforms on their application layer [20]
- Most software vendors of RPA tend to make the tools that they create easy to use and no programming skills are required as most of them are with building blocks

As we stated above and in different papers has been studied that not everything can be automated not just with RPA but also with other tools. Below there is a picture showing how RPA stands in the market and what is a general idea of what can be a potential task that can be automated.

As in every other field what drives innovation is business and, in our case, what makes business processes easier is RPA. The problem with some companies that try to implement automation into their processes, is that they fail to follow some critical guidelines. The primary focus of automation should not be reducing cost but customer satisfaction as that is every business ultimate goal [21]

3.3.1 Benefits of RPA

Software robotics as we stated in the previous paragraph has come to change the way repeating business tasks are done and enabling people to do more cognitive tasks.
In this section we will focus on certain areas that RPA claims to improve in current business processes. We will focus on some case studies to see how companies have implemented RPA and what they got in return in terms of different indicators.

There is different way we can look into the benefits of RPA. We can look into the benefits of the business and by this we mean the ROI or return of investment. This is one of the key promises of the RPA and RPA implementations in general. Usually the companies take back their investment in one year or in one year and a half. It is often mentioned that companies should focus on the long-term gains from RPA and not just on the short term. Looking in the

long term makes also the RPA implementation more useful for the processes that are automated.
Another key point regarding the return of investment is that companies should not focus only in in this component, but we will talk into details about what companies should avoid in the next paragraph "Common issues with RPA". [22]
Except for the return of investment what businesses are interested is cutting costs. In different companies' robotic process automation has made it possible to have more than 10% of cost reduction also depending on the process that is automated. [23]

In terms of time factor businesses have seen a decrease in processing time and a large possibility in scaling up and down according to the workload. This is quite the advantage for the business as they can change the "robots" number based on the workload that is available. In O_2 company which embraced RPA they saw an improvement of processes time from days to minutes or hours. [24].

In terms of IT department RPA provides easiness in implementation as it does not require for the current IT processes or systems to change. RPA can work on top of those processes and it does not require any API or backend integration. Even though IT teams need to be convinced before implementing some tool into their environment the case studies have shown that in the end IT teams have embraced robotic process automation. There are certain suggestions if the robotics should be part of other systems of IT departments or in separate environments, but this is out of the scope of this section and this thesis.

What is more robotic process automation requiring little or no time compared to ERP implementations. Usually the setting up of the environment takes several weeks compared to years that an ERP system could take to be implemented [19]

Another key point that is worth discussing is the relation between robotic process automation and outsourcing of repetitive tasks to low cost labour countries. In the case of Opus Capita [19] the company saw it more fruitful to have an inhouse RPA solution rather than delegate the work to other countries. In this way the company has the full control over its processes and the clients are more satisfied in this way.

One of the key concerns regarding the RPA implementations is that the workers will be fired if the company takes on robotic automation of some tasks that used to be done by employees. But this is not the case as with the robotic automation people that used to do this task can oversee the robots work. Also, the employees can focus on more cognitive tasks rather than do manual processes as they used to [19]. So far there is no indication of job losses in the companies that implemented robotic process automation. The fear still remains from the employees' point of view as they think that their jobs are at risk, but with RPA in house the employees can be more valuable to the company than they used to be by just doing manual work.

3.3.2 Common issues with RPA

The first to keep in mind when thinking of using RPA in your infrastructure is to evaluate if the task is repetitive or not and if it requires cognitive capabilities from third parties.

If the task is not highly repetitive and it requires extra cognitive abilities from your employees such as critical thinking than this is not the type of task you should consider automating.

Another necessary thing to consider is that you should be able to write down every step of the process and define all the possible outcomes. In this way you are sure that the process is clearly defined, and you can automate all of it or certain parts of it which are repetitive. [19] What is more to consider, is, if a company is willing and prepared to replace workers with "robots".

According to [22] one of the common issues with RPA implementations is that teams that create the PoC[1] are ready to go in production. The problem is that the implementation must be scalable and resilient.

Also, in [22] is stated that the companies fail to think about the processes after they have been automated. This step requires a thinking from the implementation side on how the bots will go live, who will be in charge of the robot workforce and who will run it.

3.4 Cognitive Computing

Cognition is defined as "The mental action or process of acquiring knowledge and understanding thought, experience, and senses." [25].
The advancement of artificial intelligence is giving a lot of advantage to different fields of technology and companies can now build human like machines, which think and act like humans would do [25].
As we said a cognitive system is a system that can perform some functions that a person would normally do such as learning, planning, deciding and making decisions. Nowadays cognitive systems are equipped with abilities like pattern recognition, NLP (natural language processing), learning and other functions which normally in the past were performed by humans [26].
The main advantage of artificial intelligence is that it can work with unstructured data, which is what a human would do. A cognitive system should be capable to work with large amount of data coming from multiple sources and generate some meaning out of this.
Big companies like Google, IBM, Amazon, Facebook and others are among the ones that provide cognitive computing services.
As we said previously a cognitive system should work with unstructured data as a human would do, and for this a cognitive system should communicate with various sources to get the data. Natural language processing and text to speech conversion are widely used in such systems. In order to have a reliable cognitive system we should take into consideration the level of confidence of our system. Improving the confidence level of one's system is also a trivial and important task.
Cognitive computing has the ability to learn making these systems different from traditional ones. These new systems can learn and adapt and what is more important they can evolve over time which allows them to act independently.

PoC – Proof of concept is a demonstration, the purpose of which is to verify that certain concepts or theories have the potential for real-world application.

What is to be cleared is that cognitive computing should not be thought as being artificial intelligence, as cognitive computing is the umbrella that has the specific technologies used by AI. On the other hand, artificial intelligence puts into practice by the use of machines, cognitive computing capabilities.

3.4.1 Applications of cognitive computing

In the next section we will go into depth of how cognitive computing is linked to RPA and how it empowers the future of robotic process automation, but in this section, we will give an overview of other applications of cognitive computing.

The earliest cognitive systems started as systems for tutoring as described in "Cognitive Tutors" 1995. These models were primarily used to help students solve problems in different areas of studies such as geometry and algebra. [27]

Problem solving systems are another area that cognitive computing found its use by solving geometry questions for SAT tests in the US. The system is capable of understanding text and diagrams. The system is called GEOS and it scored 49% on the test. [28]

Healthcare is the area of life that has a lot of data that need to be processed and sometimes systems can process and work with this data in better speed than a human would do. IBM has created a version of its cognitive system Watson for healthcare and it is called Watson Health. This product is used for diabetes diagnoses and new treatments. The way it will try to achieve new breakthroughs in diabetes is by feeding the system with 125 million data from different patients.
Another hospital the Massachusetts General is planning in using graphical processing units such as Nvidia in order to detect anomalies in CT scans. To achieve this up to 10 billions of actual images will be used to train the cognitive computing system and will help doctors in better identifying diseases such as cancer from the early stages.

IBM with Watson is trying to enter the financial industry. Nevertheless, there are critics regarding how Watson or tools like Watson perform when their cognitive abilities are put to the test. According to an ex IBM engineer `IBM Watson has great AI...It's like having great shoes but not knowing how to walk — they have to figure out how to use it` [29].
Systems like Watson health and Watson financial services are backed with some of the best AI capabilities in the market, but they fail to perform in certain cases. [30]
One issue with both systems is the large amount of data required by subject matter experts in order to have the correct results. If the domain that these systems work on is complex than the greater is the level of training needed and greater is the risk of having incorrect results.
As was noted above the more difficult the domain the harder is for cognitive tools to perform, but this should not have been the case for Watson health. Nevertheless, experts argue the success of Watson in this field as in the year 2018 IBM fired up to 300 people from three companies that they had acquired and had brought IBM a substantial amount of data to train Watson.

Another important thing to point out is that these integrations require a lot of consulting work in order to achieve the desired curated data which the late has led some companies from stepping from IBM.
According to [31] one client suspended its project with IBM after having spent an amount of approximately $60 million.
The last note serves as a reminder to companies that in order to build such integrations, not only with IBM, but with other third parties as well, a lot of thought must be given. [30]

Taking into consideration the above arguments it is necessary to point out that as with other systems, implementing AI is not the solution to every problem. A lot of prework needs to be done in order to evaluate the compatibility of AI systems with underlying infrastructure and business needs. There are cases where the companies have stopped the projects after they spent money and time, just to find out that the solution is not viable.

3.5 Intelligent Automation

Since artificial intelligence is the buzzword of the moment, we can see its applications in almost every field of technology and life. RPA vendors are trying to add cognitive capabilities to their existing robotic process automation solutions to create intelligent automation. In this section we will see some areas that intelligent automation is applied. [32]
According to IBM "The biggest payoff occurs when robotic process automation (RPA) and artificial intelligence (AI), or cognitive computing—two complementary forms of automation—are integrated to drive cognitive automation."

The banking sector offers a lot of processes that companies can standardize through RPA but are the complex ones that need the help of cognitive solutions like cases where there need to be made a decision. The banking sector strives to achieve user satisfaction, increased security and fast service.
Some other fields that cognitive automation could come in handy could include financial advice, claim and complaint handling or customer service.
In general theory combining RPA and cognitive abilities is not a difficult concept to think about, it is necessary that digital service levels and requirements are met. Also, there are some other problems regarding the implementations which we will mention some of them here and companies or individuals that want to automate can take into account.

We will try to define some related terms to intelligent automation which will guide us through the practical part of this thesis. For the terms we will cite IEEE Guide for Terms and Concepts in Intelligent Process Automation. [33]

Robotic Process Automation (RPA)- A preconfigured software instance that uses business rules and predefined activity choreography to complete the autonomous execution of a combination of processes, activities, transactions, and tasks in one or more unrelated software systems to deliver a result or service with human exception management.

Cognitive Automation (CA)- The identification, assessment, and application of available machine learning algorithms for the purpose of leveraging domain knowledge and reasoning to further automate the machine learning already present in a manner that may be thought of as cognitive. With cognitive automation, the system performs corrective actions driven by knowledge of the underlying analytics tool itself, iterates its own automation approaches and algorithms for more expansive or more thorough analysis, and is thereby able to fulfil its purpose.

Intelligent Process Automation (IPA)- A preconfigured software instance that com-bines business rules, experience-based context determination logic, and decision criteria to initiate and execute multiple interrelated human and automated processes in a dynamic context. The goal is to complete the execution of a combination of processes, activities, and tasks in one or more unrelated software systems that deliver a result or service with minimal or no human intervention.

Another key point to note is that for us to have a cognitive implementation there should be some properties such as:

1. The sources of data should be various and should be of structured or unstructured data
2. There should be an interaction with the user through one of cognitive computing fields such as NLP and generate results based on these interactions

3.6 NLU

"In simple terms NLU is a subset of a bigger picture of NLP, just like machine learning, deep learning, NLP, and data mining are a subset of a bigger picture of Artificial Intelligence(AI), which is an umbrella term for any computer program that does something smart" [34]
"A good rule of thumb is to use the term NLU to express a machine's ability to understand the natural language in a form provided by humans" [34]

3.6.1 NLU terminologies in chatbots

In this section some information about the terminologies used in chatbots will be explained. This will make easier the transition to the practical part where Rasa NLU will be used to create the chatbot.

The first term that we will come across is intent. When a user interacts with a chatbot there will be a request by the side of the user which should be understood by the chatbot, this is intent. Explaining intent with an example would look like this:

```
1. User: I want to order food
2. Chatbot: the intent is food or order_food.
```

Another term which comes together with intent is entity. In the above example that we used to explain the intent it was clear that the intent is ordering food, while food on its own is an entity for the chatbot. This entity can be of anything such as flight, movie tickets or anything that a user can ask for.

The last term is utterance. Utterance is the different ways a user can express the question or the intent. This is one of the hardest problems in chatbots as users can use synonyms for words and the chatbot should come up with the same intent. [34]

3.6.2 NLU evaluation

The following terminology will serve as the foundation of the benchmarking of our NLU and core models of the tool that will be chosen for the conversation management.
The important task in machine learning is that the data are transformed to the models properly and the model is representing the system with an accuracy that can be accepted.

- **Confusion Matrix**

Model validation techniques are various but one of the most used and reliable is the one called N-fold cross-validation technique. "This technique divides the dataset into N number of parts, and each of them consists of an equal number of samples from the original dataset. For each part, training is performed with (N-1) number of parts and the test is done with that part." [35]
The output of this technique is what is called a confusion matrix which is a summarization of a classifier with respect to some testing data [36]. The confusion matrix will be represented in:

Table 1: Confusion Matrix

		Assigned class	
Actual class		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

From Table 1 the output will contain results that are:

- ***True negatives (TN)*** *which* are the negative examples that are correctly classified by a classification model. [36]
- ***True positives (TP)*** are the positive examples that are correctly classified by a classification model. [36]
- ***False positive (FP)*** is an example of a negative class that has been incorrectly classified as positive [36]
- ***False negative* (FN)** is an example of positive class that has been incorrectly classified as negative. [36]

The above data can be used to have a number of classification performance indicators:

$$Recall = \frac{TP}{TN + FN} \; also\; called\; the\; probability\; of\; detection$$

$$Precision = \frac{TP}{TP + FP} \quad the\ data\ points\ that\ were\ relevant$$

Another metric used for the evaluation of prediction system is the F-measure.

$$F - measure = \frac{2 * (recall + Precision)}{recall + Precision}$$

3.6.3 Rasa

Rasa is a tool that is comprised out of two python libraries called Rasa NLU and Rasa Core which are used to create conversational software. Both libraries are open source and free to use by anyone. [37]
As far as the architecture of Rasa is concerned the approach followed by the creators is to make the tool modular. Making it modular gives Rasa users the ability to integrate easily with other tools rather than Rasa. One common example would be to use Rasa core for dialogue management and use another NLU tool for the language part and not using Rasa NLU. [37]

3.6.3.1 Architecture

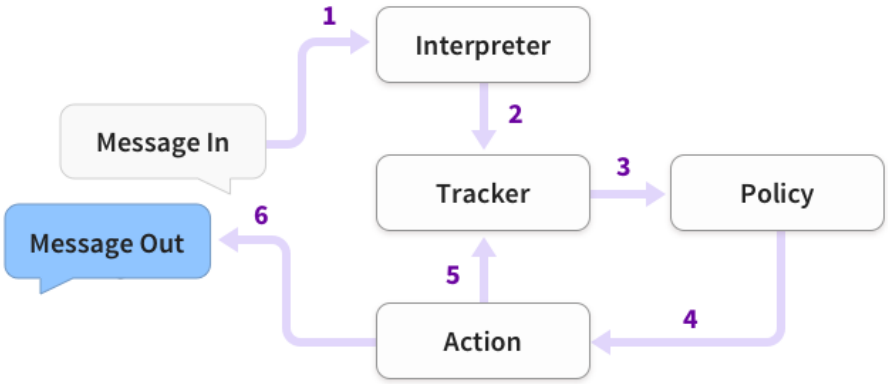


Figure 1: Rasa conversation management

The workflow in *Figure 1* represents how Rasa manages the conversation. The message that comes in from the user is interpreted by the NLU part, which in our case is Rasa NLU. In this step the NLU component will try to extract the intent of the user, what the entities are and any other structured content.
The tracker is the component that maintains the state of the conversation. There is one tracker object responsible for every conversation session. The trackers duty is to store slots and all the steps that led to that state.
In the next step the policy receives the current state of the tracker and based on that the policy is responsible for choosing the next action to be taken. This action as well is logged by the tracker.

The user receives some output from Rasa. The action predicted by Rasa could also be a listen action, which excepts some user input.

3.6.3.2 Actions

Rasa approaches the problem of dialogue management in the form of classification problem. Rasa core is the part of the tool which manages the dialogue and in the case of actions from a list of predefined actions the next action is predicted to be sent to the user. When an action is executed it is passed to the tracker which in turn keeps the state of the conversation.

3.6.3.3 Natural language understanding

This part is managed by the component called Rasa NLU. It is a combination of machine learning and natural language libraries that are served through an API. The data used for the training can be provided by the so-called pipelines which offer pretrained data sets. One of the recommended pipelines is spacy_sklearn. How spacy sklearn works is by pretrained word vectors. The information regarding how pipelines work in Rasa will be explained in detail in the section 3.6.3.6.

3.6.3.4 Policies

What a policy does is to select the next action to be passed to the tracker. A policy is instantiated along with a featurizer[2], which creates a vector representation of the current dialogue state given to the tracker. On the other hand, the core module of Rasa uses markdown format to specify dialogue training or as we mentioned the stories.

3.6.3.5 Training data

Both modules of Rasa work with human-readable training data. The NLU module requires a list of utterances annotated with intents and entities. This data can be feed to the system either in markdown or json format.

3.6.3.6 Pipelines

“In Rasa NLU, incoming messages are processed by a sequence of components. These components are executed one after another in a so-called processing pipeline. There are components for entity extraction, for intent classification, response selection, pre-processing, and others. “ [38]
The three most important part of a Rasa pipeline are [38]:

1. Tokenization
2. Featurization
3. Entity Recognition / Intent Classification / Response Selectors

[2]In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon being observed

Tokenization makes possible to split the text into tokens. If the text that the user is working with is not whitespace separated or the users' needs to separate the intent into labels this can be done with Rasa. Moreover, there are languages which are not separated by whitespace and this must be taken into consideration as well.

Regarding featurization Rasa divides it in two categories: sparse and dense. Sparse return feature vectors with missing values such as zero which would consume a lot of memory, so in Rasa these are stored as sparse ones. Sparse features store only the values that are only non-zero and where their position in the vector is. This gives the ability to train larges datasets and reduce the memory consumption. In this thesis in the practical part will be selected one featurizer and the reasons will be explained, as Rasa provides a number of featurizers for its users. [38]

The role of the intent classifier is to assign one of the intents defined in the domain file to the users incoming message, while entity recognition does the job of extracting entities from the received messages. [38]

3.7 Current state of RPA in user support systems

In this thesis our selected area of automation is user support system which is part of back office operations. According to Investopedia "The back office is the portion of a company made up of administration and support personnel who are not client-facing ". The idea is to automate a process through NLU and RPA. Firstly, it is necessary to highlight what the current trend of RPA is in this area and what is the need of organizations for automation.
In the year 2010 Telefonica O2 thought of transforming its office operations by outsourcing it. Even with the added workforce the cost went up and there were still place for improvement in the processes. The company at that point thought of adding RPA automation to their suite of applications. At first, they just started with 2 processes and just a proof of concept in order to verify the RPA success. After that Telefonica O2 continued to add more services to being automated by RPA and also advised their outsourcing company to do the same. [24]
Based on another study done by Capgemini `` Process candidates for RPA can be found in all back office areas ``. What is more important is that when the study was done the companies part of the survey were expressing that in the next 3-5 years, they would be using RPA in a scale of 81% out of 100% of the participating companies.
Based on a research [39] there are several processes in Human Resources that can be automated with RPA. Task such as candidate status notification, interview scheduling and more. Even though HR is not a back office operation tasks that are mentioned above are. This are manual tasks that require a human to accomplish. Using RPA in such tasks can free people on doing interviews and scheduling their time much better than waste it with tasks that can be automated.

Based on a Gartner report made in 2019 [40] in the next two years there will be an increase of importance for bots. This is backed by 68% of the service leaders. Living in a time where customer support and satisfaction is really crucial RPA can be a game changer.
Implementing RPA in user support gives the company a coverage of 24/7 without break. Also, a minimization of human errors and increased customer satisfaction. [41]

According to a research made by AIMultiple, [42] which is a company focused mainly in Artificial intelligence, there are numerous back office and user support processes that are automated. In their research they list the companies that were part of their survey and the processes that these companies have automated using RPA.
A food and beverage company has used a commercial product called Automation Anywhere to automate their help desk processes.
Part of the list are more companies that have automated other back office processes by using RPA.

4 Practical Part

RPA has taken a boost the recent years but so has the open source community and the usage of open source software [10]. Taking into consideration this two facts we will try to provide a solution with two components: RPA and NLU.
We are also interested in finding out if user support systems are compatible for RPA automation and what is the current trend in this field as well.
As mentioned earlier a software solution will be provided to tackle a process and automate it. In order to achieve this a set of tools need to be compared and after the results we will build the automation.
Based on our solution and the current state of not open source solutions we will conclude if using open source in this field is optimal and can achieve the desired business value.

The following three sections will include all the findings and results on the topics stated above.

4.1 Designing an open source solution of cognitive automation

The idea of cognitive automation in user support comprises of two critical parts. The cognitive part and RPA part. For both of these parts there will be needed two different tools that will communicate together to deliver the desired result. In literature review are described all available open source tools that offer the capabilities of RPA. In this section we will compare these tools based on certain criteria. After the comparison one of the tools that matches our requirements the best will be selected, and the RPA part will be implemented.

4.2 Comparison of tools

As noted above it is necessary for the purpose of this thesis that comparison takes place between the various open source tools that were described in literature review. Since we are talking about open source tools and the code is free to use by anyone being that a person or company, we will use GitHub as our main of reference to benchmark our tools.
Based on the criteria described below the tool for the practical part will be selected.

4.2.1 Technologies that are used to create the tools

a. Robot framework – is mainly based in the libraries that are around the tool. The creators divide the libraries in three categories: standard, external and other libraries. Most of the libraries are browser driven such as selenium, http requests library and much more. Anyone interested in using the tool can refer to the documentation and get the full overview of the libraries that are used.
b. Roro – the creator of roro has based it on C# which makes it Windows OS compatible only.
c. Automagica – the tool is built by a concentration of open source Python libraries. The information regarding these libraries is not shared. To discover those libraries,

it is necessary going through the python code of Automagica. The goal of the tool is to create wrappers around other tools.

d. TagUI – is made by a compound of other open source technologies such as SikuliX which is used for desktop automation powered by another open source tool which is OpenCV. Other tools include PhantomJs which provides scriptable headless browser and Slimmerjs which is used for browser automation.
e. Taskt – there is little information on the technologies that Taskt uses but is written in .NET framework
f. OpenRPA – The information regarding the technologies used are missing, but it is mainly built to support Microsoft Workflow Foundation.

4.2.2 Programming languages supported

Having an overview of the extensibility of the tool gives the users and companies a clear image of the capabilities of the tool and what is more allows the end users to know if what they currently know will suit with the new tool.

a. Robot framework is written in python and can be extended by using libraries in other languages such as Java.
b. Roro – since is open source anyone can add feature to the tool, but at the moment there is no documentation as what the extensibility of the tool is.
c. Automagica – is written in python and as with the other tools can be extended by using other libraries from external sources
d. TagUI – is written in python but offer large extensibility and also is possible to include Python and R for machine learning
e. Taskt – the creators of the tools are really open on the features that someone thinks they are missing so it is possible to require a feature addition or use the code in your own responsibility
f. OpenRPA – offers no information regarding the extensibility. It is built in C# as it runs on Windows OS only.

4.2.3 Platforms supported

Platforms supported gives information where the tool can be used. The more operating systems supported the better is for the users of the tool. Being platform dependent adds extra costs to the final bill of the company of users.

a. Robot framework is cross platform which means it can be used in any operating system
b. Roro – is only available in Windows operating system
c. Automagica – is officially supported for Windows 10 operating system, but can be installed by using python packages
d. TagUI is cross platform which as we noted before means that it works on any operating system
e. Taskt – is only supported in windows 10 machines and .NET framework 4.6
f. OpenRPA – the only platform supported is Windows operating system because as we mentioned it uses Microsoft Workflow Foundation.

4.2.4 Available features

a. Robot framework in terms of RPA is mainly focused on application testing but can be connected to any other tools that the user needs to fulfil the automation.
b. Roro – the tool at the moment supports OCR which stands for optical character recognition as part of its cognitive features
c. Automagica – provides a range of features such as web scraping in chrome browser, excel automation by reading and writing data, PowerPoint presentations and lastly using OCR to navigate by images
d. TagUI – the tool offers the possibility to automate chrome in visible and invisible mode. Visual automation of websites and desktop. There is also a chrome extension that offers extra features to the users that would like to make Chrome automation.
e. Taskt – includes features such as image recognition, OCR and custom code execution.
f. OpenRPA – provides to its users' visual automation for desktop applications. Also, browser automation as well. OCR is part of the suite of features as well.

4.2.5 Learning curve

What is meant by learning curve is how hard new users will find working with one of the below tools and if there is documentation in place to help them overcome certain difficulties. This is really important measure when we talk about open source software as there is no proper support compared to commercial offerings.

a. Robot framework - In order to use the tool no programming is needed, so the test cases can be written in plain human language. This makes the learning curve not so steep for new users.
b. Roro - The learning curve is not so steep as the tool does not offer too many features and support for too many platforms as well.
c. Automagica – the documentation is detailed and users can refer to it in order to achieve their RPA automation. Having documentation in place makes it easy to get familiar with the tool
d. TagUI – offers the possibility to write your "code" in twenty plus human languages and JavaScript
e. Taskt – there is documentation regarding the possible commands, so the users have where to find how to achieve a specific action. In general, is fairly easy to do a simple automation
f. OpenRPA – getting familiar with OpenRPA seems to be easy as with the other tools. There is no need for programming and the documentation in place makes it easier for new users. The tool is using workflow method of creating the automations rather than scripting

4.2.6 Documentation

In the section above we mentioned the learning curve needed in order to use these tools and we mentioned that documentation is necessary in order to have a smoother learning curve. In this section it will be mentioned if the tools have documentation in place and if it is comprehensive.

a. Robot framework - There is documentation available which is hosted in the tools website as well as in GitHub. Also, there are examples which the users can refer too.
b. Roro – lacks in the part of documentation as there is no documentation except for one example which is not very useful
c. Automagica – the documentation is available and can provide users with what the tool is capable
d. TagUI – there is documentation available in github and also there is a forum explaining different use cases for the tool
e. Taskt - The documentation includes a list of commands that can be used in order to automate different tasks
f. OpenRPA – the tool has its wiki pages where the users can learn to do basic tasks. The creators and community have also a Slack channel that anyone can join.

4.2.7 GitHub users and contributors

Having an idea of how big the community behind the tool is gives some insight of the scale that the tool has. Furthermore, new features will be available quickly if the contributors are more in number than a few people. Apart from the Github users and community we will try to find additional information if the tool is backed up by some organization which will impact in a direct manner the speed that the tool is developing. The number of forks gives an overview of how many people have taken the source code to use it for their special needs.

a. Robot framework – has 99 contributors 1.3 k forks. The project has 86 releases
b. Roro – has only 1 contributor which explains also the results above such as missing documentation. The project has zero releases
c. Automagica – has 6 contributors 232 forks
d. TagUI – has 10 contributors apart from AI Singapore that supports it. Has 286 forks and there are 18 releases of the tool
e. Taskt – has 6 contributors 89 forks. The project has 44 releases.
f. OpenRPA – has 4 contributors and the project has been forked 71 times. Number of releases is 142.

4.2.8 Open and closed issues

This “metric” gives us the ability to see how active the community behind the tool is. By having too many open issues means also that the tool is buggy and there is no workaround now for the specific case.

Table 2: Open vs Closed issues

Tool Name	Open Issues	Closed Issues
Robot Framework	108	3024
Roro	0	0
Automagica	18	52
Taskt	4	122
TagUI	3	625
OpenRPA	2	35

Based on the Table 2 above the conclusion that we get is that TagUI seems to have the most active community. The number of open issues compared to the closed ones is quite low. On the other hand, Roro is not widely used as a tool as there are no issues at all.

4.3 Implementation

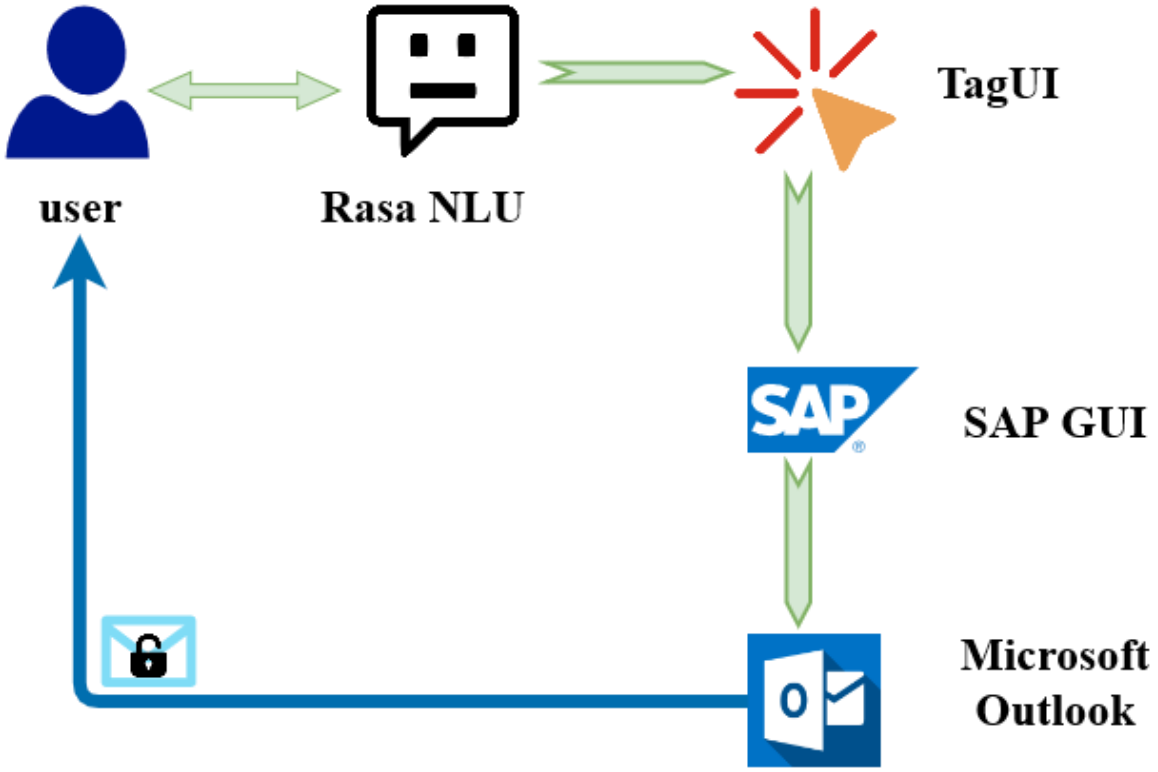


Figure 2: Architecture overview of proposed solution

In *Figure 2* is displayed an architecture overview of our proposed solution. This architecture will lay the foundation for the following sections and the selection of tools.
In the previous section several criteria were used to compare the open source software that are available to create robotic process automation. Based on the above criteria the tool that is going to be used for the implementation part is TagUI.
TagUI was chosen as the tool for the RPA part because the user does not need to have any previous coding experience. Furthermore, it is available in more than 20 languages which makes it possible for anyone even if you do not speak English.
The documentation is in place in order to find answers to the questions or regarding different commands. The number of issues closed in GitHub compared to those that are open is really slow which means the community behind it is really passionate and proving what open source software really means.
For our purposes we need to use the feature of image recognition which is provided by SikuliX integration that TagUI provides out of the box. We will not provide details about SikuliX or how image recognition works as it is out of the scope of this thesis.
Furthermore, apart from the RPA part there will be an integration with NLU tool called Rasa NLU. TagUI is really useful for this integration as it includes a python package called *tagui*.

Another independent contribution from the community and based on TagUI is called *rpa* and is available in Python pip package manager.
The details listed above provide clear reasons why TagUI was selected as the tool of choice for building the RPA.

The process that we will select to implement our cognitive automation is comprised of two parts. The first part is the user part where the chatbot will understand the intent of the user and the second part is the RPA part that will accomplish what the user asked. As mentioned before the process that will be automated is part of user support systems and specifically password change in an ERP system such as SAP GUI.
The scenario is that the user has forgotten the password to login to the system and for this reason will use the chatbot. On the other hand, after the chatbot has understood what the user wants it will trigger the RPA to actually change the password in the system for the specific user. After the user's password is changed the user should receive an email that the password was changed.
For the communication with the user the tool called Rasa NLU will be used. Rasa as mentioned in the literature review has the capability to understand user intent and for the RPA part after the results that we got from tools comparison TagUI will be used.

In the following two sections the part of NLU and RPA will be explained in detail and all the necessary configurations to make the project work.

4.3.1 Preparing the experimental environment

Creating the environment for the experiment is important so that we can reach to conclusions and make final recommendations.
For the purposes of this thesis a Windows 10 machine will be used. The main reason for selecting a Windows OS is that SAP GUI, a client to connect to SAP server, is much easier installed. The version of the client that will be used in our thesis is 7.3.1 but there are newer versions as well. If the user owns a newer version of SAP GUI then the images used by TagUI should be of that system.
The two other components needed for the experiment are TagUI and RASA NLU. Since both of the tools have python pip packages (Python Package Index https://pypi.org/) we will use pip to install.
Also in order to keep our operating system clean and in case that something goes wrong during the setup Python virtual environments (Virtualenv https://virtualenv.pypa.io/en/latest/) will be used. This gives the flexibility of having multiple environments that do not conflict with each other or different Python versions that the users could have in their machine.

The first step is to confirm that python version in your machine is greater that 2.7.0. In order to achieve this write in you Windows PowerShell (https://docs.microsoft.com/en-us/powershell/):

```
1. python --version
```

In this thesis the output of the above command is:

```
PS C:\Users\Diploma> python --version
Python 3.6.7
```

After the python version is verified two packages need to be installed python pip and *virtualenv* which will be installed by the pip package. In order to install *pip* the following procedure will be used:

First *pip* should be downloaded and then python will be used to install the file that was downloaded:

```
python get-pip.py
```

Next it is necessary to have the latest pip installed so:

```
pip --version
```

At the time of this thesis pip latest version is: ***pip 20.0.2.*** If pip is not the latest version than upgrading it is quite is by running:

```
python -m pip install --upgrade pip
```

The below command confirms that the pip version installed in Windows is the latest:

```
PS C:\Users\Diploma> pip --version
pip 20.0.2 from \\path\\ (python 3.6)
```

The next step after pip package is installed would be to use it in order to install python virtualenv. Use the following command to install *virtualenv* in your machine:

```
pip install virtualenv
```

At this step it would be possible to create the virtual environment where TagUI and Rasa NLU will be installed.

To create a virtual environment in python, go to your desired directory in command line, and run:

```
virtualenv name_of_your_venv
```

a new directory called `name_of_your_venv` will be created. In order to use the virtual environment, you should activate the environment. What we mean by activation is going into the **script** directory in your **virtualenv** and running activate, or by putting the path of **activate** file in your shell.

```
PS C:\Users\Diploma\Desktop\virtualenv_diploma> .\env\Scripts\activate
(env) PS C:\Users\Diploma\Desktop\virtualenv_diploma>
```

(env) In your case would be **`name_of_your_venv`**

The above output indicates that you are inside the python virtual environment and any change will be only visible inside that environment.

After the python packages installation is completed successfully, we should continue with the installation of the two main tools needed for this experimental part.

As it was mentioned previously for the installation of TagUI will be used the pip package provided by the community. The package can be found under the name **RPA** in pip repository.
To install TagUI pip package as it is with other packages one command is necessary:

```
1.  pip install rpa
```

Please keep in mind that the installation of the tools should be done while inside the virtual environment created in the previous step.

In order to verify that TagUI is installed successfully we will create a short script for testing. In the directory that you are, or in a directory of your choice create a file called *test.py*.
In the contents of this file there should be the following script:

```
1.  import rpa as r
2.
3.  r.init()
4.  r.url('https://www.google.com')
5.  wait 2 sec
6.  r.close()
```

The above script should open Chrome browser and type in the URL field https://www.google.com.

The first time that the automation runs TagUI is setup. This includes creating the environment with all the necessary tools for OCR, Visual and web automation. The package size is 200 Mb. After the package is downloaded the automation should be completed.

The final step in creating the experimental environment would be the installation of RasaNLU. Installing Rasa is quite straightforward, and it requires only one command:

```
1.  pip install rasa
```

Checking if rasa was installed and what version:

```
1.  (env) PS C:\Users\Diploma\Desktop\virtualenv_diploma> rasa --version
2.  Rasa 1.8.0
3.  (env) PS C:\Users\Diploma\Desktop\virtualenv_diploma>
```

The finalization of this step means that our testing environment is ready. In the next sections will be explained in detail how each tool is configured to achieve the desired result.

4.3.2 NLU

Rasa NLU is an open source tool used for intent classification, response retrieval and entity extraction in chatbots. [43]

In order for the chatbot to communicate with the user it is necessary to have a domain where the chatbot works. Moreover, the chatbot must understand the responses of the user meaning that the chatbot should be trained.
In this thesis it will not be used a pipeline because as described now the users can select their components for their *config.yml*.
Throughout the time Rasa has removed the predefined templates such as *supervised_embeddigns*. The reasoning behind is that the user can now select all the components that fit their needs such as featurizer and tokenizer.

In order to have a full working chatbot a data file which will be used for training should be provided to Rasa NLU. Also, stories should be created in order for the chatbot to communicate with the users.
The domain file, training data and stories used to power the chatbot will be included in the appendix section of the thesis.

Apart from training the chatbot to communicate with the user for the purpose of this project it is needed that the chatbot takes some action. That action is triggering the RPA part of the project.

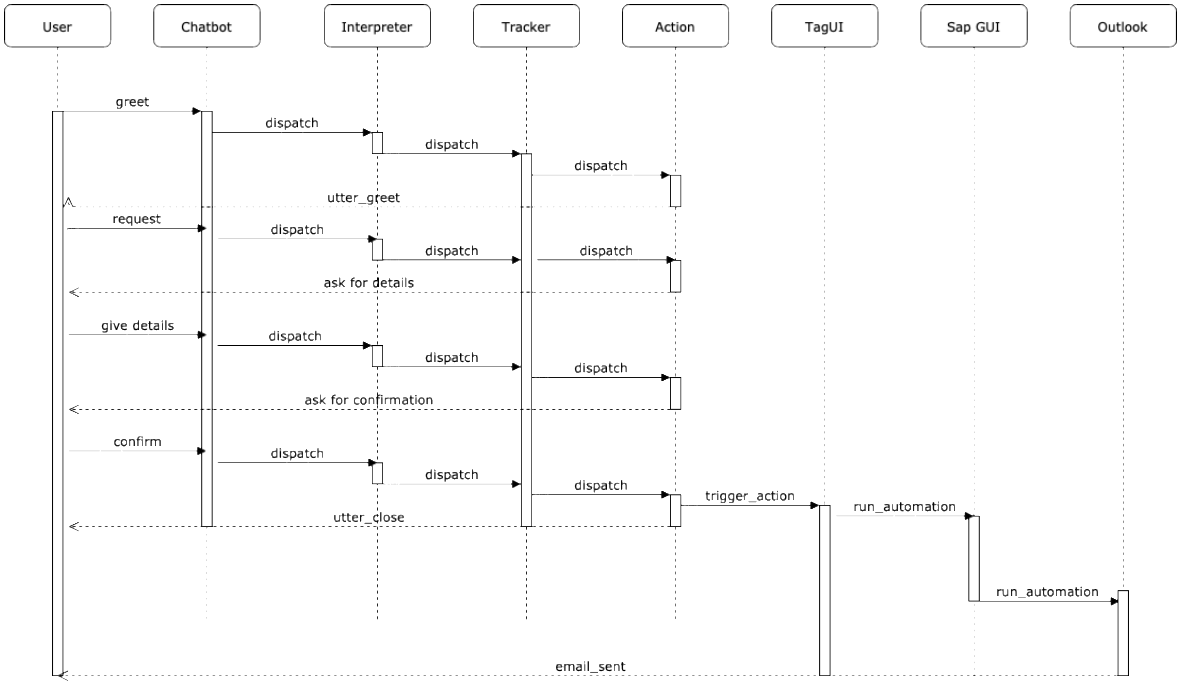


Figure 3: Sequential Diagram

As it is shown in *Figure 3* the chatbot will extract the username, email and system name from the user (ask for details). These three details are really important as the RPA needs to know the username that the password is lost the email address of the recipient after the password change has happened and lastly what the system name is. Being an ERP user means that the user works in more than one system, so it is necessary for the RPA part to know in which system to login and change the password.
After the extraction is complete the chatbot should trigger the RPA bot to proceed with the automation. The information extracted above should be passed to the RPA bot from the chatbot. In order to achieve this a python custom action will be created in Rasa NLU.

In the beginning of this section the foundations of what the NLU part will do was outlined. Below will be explained the steps to create and train your chatbot for your domain. This thesis use case domain is the change of password in an ERP system.

The first step would be to create a new Rasa project which will include all the necessary files to continue with the project. To initiate a new project the user needs to be inside the virtual environment as explained in section 4.3.1.

```
rasa init --no-prompt
```

this command will create a new Rasa project for you. In case you do not want to create an environment as we did, you could use Rasa online environments.
The following files will be created.

```
__init__.py             an empty file that helps python find your actions
actions.py              code for your custom actions
config.yml ‘*’          configuration of your NLU and Core models
credentials.yml         details for connecting to other services
data/nlu.md ‘*’         your NLU training data
data/stories.md ‘*’     your stories
domain.yml ‘*’          your assistant’s domain
endpoints.yml           details for connecting to channels like fb messenger
models/<timestamp>.tar.gz   your initial model
```

As it was mentioned previously it is important that our bot understands what the user’s requests are. In order to achieve that a dataset must be created, and the bot must be trained with this dataset.
In this thesis we will use rasa-nlu-trainer that is an online tool to create your data in json format. In the moment of writing this thesis the Rasa creators are suggesting using Rasa X for editing your training data. Using Rasa X would be a future step for this thesis, but at the moment this thesis was started rasa-nlu-trainer(*Figure 4*) was the suggested tool.

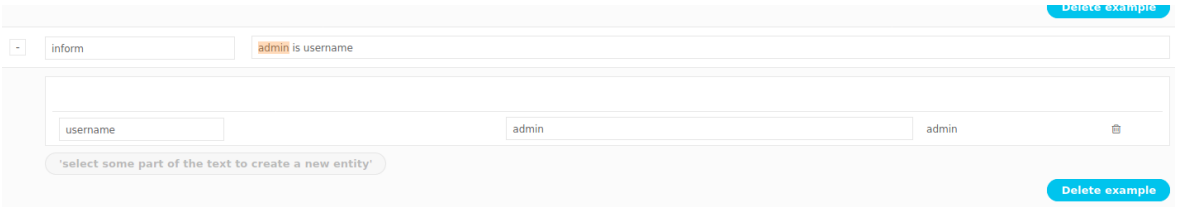


Figure 4: rasa-nlu-trainer

From *Figure 4* we can see that admin is the entity and the intent is inform. This information will be used by the bot to understand when the user will be chatting.

```
{
    "rasa_nlu_data": {
      "common_examples": [
        {
          "text": "hey",
          "intent": "greet",
          "entities": []
        },
        {
          "text": "hello",
```

```
            "intent": "greet",
            "entities": []
        },
        {
            "text": "hi",
            "intent": "greet",
            "entities": []
        },
        {
            "text": "i want to change the password for the login",
            "intent": "password",
            "entities": []
        },
        {
            "text": "login forget",
            "intent": "password",
            "entities": []
          },
          {
            "text": "my login alket",
            "intent": "inform",
            "entities": [
              {
                "start": 9,
                "end": 14,
                "value": "alket",
                "entity": "username"
              }
            ]
          },
          {
            "text": "my email address is alketcami@gmail.com",
            "intent": "inform",
            "entities": [
              {
                "start": 20,
                "end": 39,
                "value": "alketcami@gmail.com",
                "entity": "email"
              }
            ]
          }
        ]
    }
}
```

Above is an extract from the data that will be used in our use case. The complete data used for training the bot will be added to the Appendix for the user's reference. The data format used in this case is json.

The next step after the training data are created is to set up the configuration file which in turn will decide which NLU and core components the model will use. The file that needs to be setup is called **config.yml**. There are two section to be configured, the NLU and core section. For the NLU section a pipeline should be selected to train our data. Since our domain is specific and we will use our own training data the pipeline to be used will be the *supervised_embeddings* pipeline which is deprecated at the Rasa version that it is being used in this thesis. Nevertheless, the components of a pipeline can now be defined separately in the *config* file and our approach will be explained in this section. In section 3.6.3.6 we explained what a pipeline is and how Rasa makes use of them. Depending on the use case

there are other pipelines to be used. Below will be the *config.yaml* that will be used in this use case.

```yaml
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy
```

As explained at the beginning of this section Rasa has deprecated the pipeline templates, so all components of a pipeline can be changed in the new version. The components above can be found as part of a default pipeline or can be changed to fit the needs of the users based on their domain and their language.
Apart from user requirements there are also other prerequisites that prevent the usage of some components. This would be *tensorflow_text* which is used by several tokenizers. This component is not available for **Windows OS** machines, just for Linux ones. This is a problem that we encountered with the implementation and in that case other components will be used. We suppose that this will not pose a threat to our implementation as our domain is specific and the training data will be enough.

Since our data and words will be separated by whitespace, then the selected tokenizer fits our needs. In the case that the language is not white space tokenized there are other tokenizers which can be found on the official documentation of Rasa or other sources. As it is visible from the *config.yml* code above our selected tokenizer is `WhitespaceTokenizer`.

After the steps to prepare the data to train the bot are completed what is important is to have a conversation, and in order to achieve this, stories need to be generated. Stories are the bot's response to the captured intents and entities. This part the part that is being managed by Rasa core module.

The next step in creating the Rasa bot is to prepare the domain. As stated earlier the domain is where the bot will operate which in our case is the password change.
In the domain file should be clear what the bot will expect meaning intents, what actions it should predict and respond with actions.
In the domain file there should be listed three parts as shown in (Table 3):

Table 3: Domain.yml

intents	things you expect users to say
actions	things your assistant can do and say
templates	template strings for the things your assistant can say

```
slots:
  username:
    type: text
  email:
    type: text
  system:
    type: text

intents:
  - greet
  - goodbye
  - affirm
  - deny
  - mood_great
  - mood_unhappy
  - inform
  - password
  - system_name
  - email
entities:
  - username
  - email
  - system

actions:
- utter_greet
- utter_did_that_help
- utter_happy
- utter_goodbye
- utter_password
- action_confirm
- action_tagui
- utter_repeat
- write_username
- utter_system_name
- utter_email
- utter_confirm

responses:
  utter_greet:
  - text: "Hey! How can i help you?"


  utter_did_that_help:
  - text: "Did that help you?"

  utter_happy:
```

```
48.   - text: "Great carry on!"
49.
50.   utter_goodbye:
51.   - text: "Bye"
52.
53.   utter_password:
54.   - text: "I see you want to change the password. Can you provide me with your us
    ername?"
55.
56.   utter_email:
57.   - text: "Can you provide your email address?"
58.
59.   utter_repeat:
60.   - text: "Can you repeat?"
61.
62.   utter_system_name:
63.   - text: "Thanks.Can you provide the name of the system please?"
64.
65.   utter_confirm:
66.   - text: "Is the information correct?"
```

The domain file used in our case is the listed above. From the domain file we can distinguish that the desired intents are *inform*, *password*, *system_name* and *email*. There are also other intents that will be part of the conversation as well, but the prior ones are the ones that our automation is interested in.

Finally, all the parts have been set up. The last step is to train our two models with the data that we have provided in the different files that were configured above. The commands used to train Rasa are three:

1. rasa train – both models (NLU and core)
2. rasa train nlu - for the NLU part
3. rasa train core - for the core part.

The newly trained model will be stored under the models/ directory in the Rasa project that was created. The usage of the models/ directory is that when newly made changes will be trained Rasa will train only the differences not the whole model from the beginning.

```
1.  class Action_confirm(Action):
2.      def name(self):
3.          return "action_confirm"
4.
5.
6.      def run(self, dispatcher, tracker, domain):
7.          username = tracker.get_slot('username')
8.          email = tracker.get_slot('email')
9.          system = tracker.get_slot('system')
10.         response = """ we will reset the password for username {}, to the system
    {} and send an email to {}""".format(username,system, email)
11.         dispatcher.utter_message(response)
```

- Interacting with the chatbot

For the purposes of our implementation the tool created will not be as fancy as a production ready. In this case it will be used the terminal to communicate with the chatbot. Further integrations need to be done in order to have Rasa as part of a website or to be linked to other tools such as: Slack, Microsoft teams etc.
In order to chat with Rasa a command needs to be run in the directory that you created your project, as it is important that Rasa reads you trained model and other configuration files.
Moreover, since we are running custom actions it is important that Rasa server should be running at the same time in the background. To configure Rasa server, it is necessary to edit the *endpoints.yml* and if your actions are running locally as in our case than the file should look like this:

```
action_endpoint:
  url: "http://localhost:5055/webhook"
```

These lines need to be commented out in order for Rasa to run custom python actions. This configuration creates a webhoom locally in the specified URL. This url can be modified as per your needs.
In order to run the action server the below command can be used:

```
> rasa run actions
```

After the command is run this is what will be logged in the terminal:

```
020-03-
25 19:20:42.498016: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore a
bove cudart dlerror if you do not have a GPU set up on your machine.
2020-03-
25 19:20:45 INFO     rasa_sdk.endpoint  - Starting action endpoint server...
2020-03-
25 19:20:45 INFO     rasa_sdk.executor  - Registered function for 'action_confirm
'.
2020-03-
25 19:20:45 INFO     rasa_sdk.executor  - Registered function for 'action_tagui'.

Exception occurred while handling uri: 'http://localhost:5055/webhook'
```

The output ensures that the action server is running and we can proceed with chatting with Rasa.

To start the chatbot the command used in the terminal would be:

```
rasa shell
```

which will initiate the chatbot and we can communicate with it. Below the communication that will take place between a user and the chatbot will be visualized.

```
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  hello
Hey! How can i help you?
```

```
Your input ->  i want to change my pasword
I see you want to change the password. Can you provide me with your username?
Your input ->  alket
Thanks.Can you provide the name of the system please?
Your input ->  NPL
Can you provide your email address?
Your input ->  alketshabani@outlook.com
 we will reset the password for username alket, to the system NLP and send an ema
il to alketshabani@outlook.com. Can you confirm?
Your input ->  yes
Bye
```

As can be seen from the above output in terminal the chatbot understood all our provided information, and afer the user confirmed the validity of the information the bot is configured to triger the RPA part as it was mentioned earlier.
What is interesting to be noted is that in the first line it was written ***pasword*** intentionally and the bot was able to pick up what the user intended. The reason for this is that usually typos will be part of a conversation.

4.3.3 RPA

It is necessary that before the implementation of the RPA is completed the process should be broken down in all the steps necessary. Breaking down the process is really crucial because as it is mentioned in the literature review having a process that could be automated end to end is very important.

There are two applications that TagUI will interact with: SAP Gui and Microsoft outlook. The first one is the tool that the user works and has lost access to and the mail application will be used to notify the user that his request was completed.

In the following two use cases the steps that TagUI will replicate are shown. As mentioned in literature review breaking down the process is critical for the success of RPA automation. Another thing to point out is that the process is manual or not and if it requires human interaction along the way.

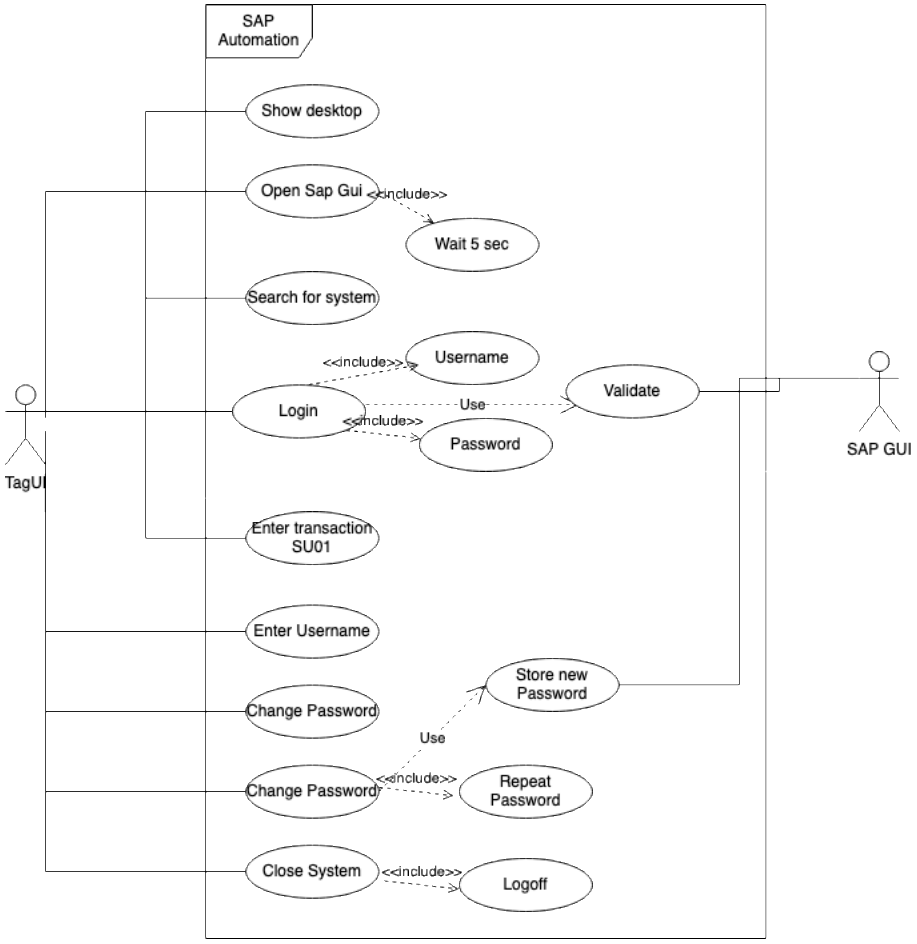


Figure 5: Sap GUI use case

Based on the above use case diagram we can tell that the process that we have selected fulfils the criteria to be automated.
Our process is related to SAP GUI (*Figure 6*)lost access, but in user support systems there are a number of systems that the users might lose access to. Our implementation is covering just a small fraction of what is possible through RPA. The only thing to keep in mind is that every process should be well thought end to end in order to have a successful implementation of RPA.

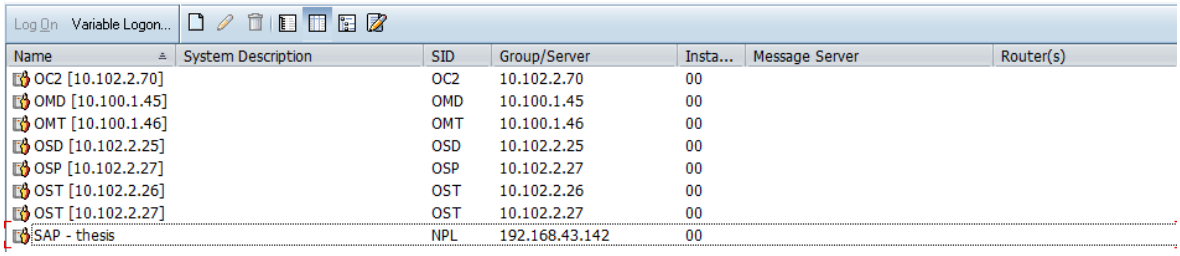


Figure 6: Sap GUI

The same rules apply also to Outlook application automation through TagUI. The below use case *Figure 7* is breaking down the processing of sending the email to the user after the automation is finished. From the diagram it is visible that every interaction with the application is defined properly.

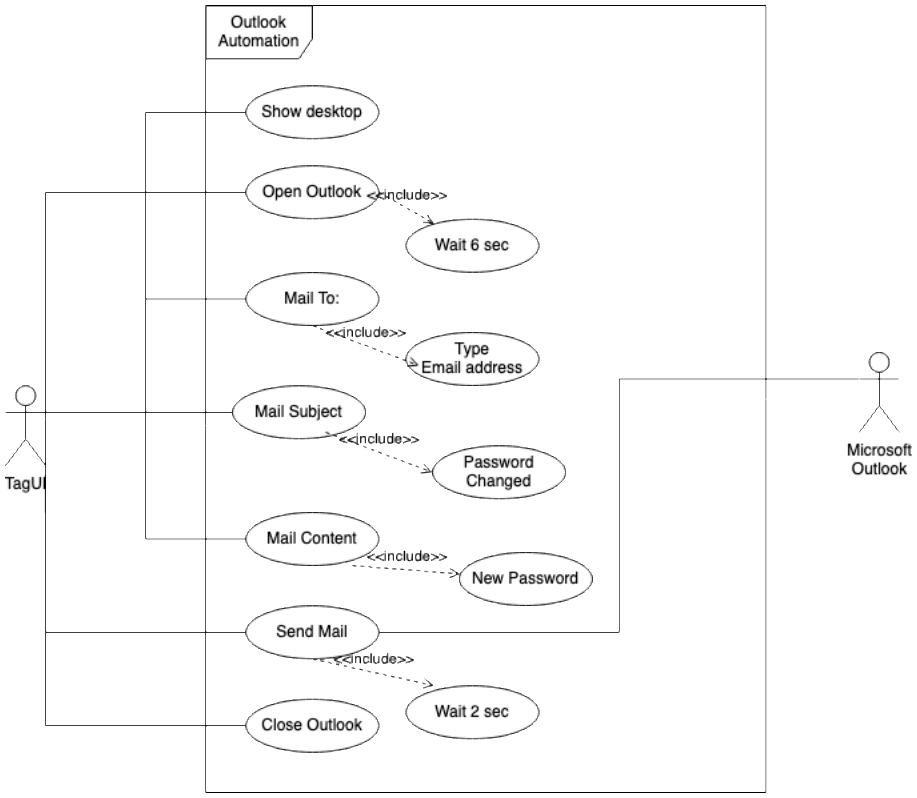


Figure 7: Outlook use case

After the steps for the automation are clear they should be followed by creating the code and using TagUI for the RPA part. In both cases we will use visual automation within TagUI. TagUI leverages SikuliX integration that is based on OpenCV and has the ability to detect GUI components. This is done by image recognition capabilities built into SikuliX. In order for us to acheive the automation we need to guide the tool through the components that we need to use.

Every component that will be clicked should be captured in the form of image and then will be supplied to TagUI through the file path. After this step if we run the automation TagUI should be capable of finding the elements in the screen.

It was previously mentioned that to create the automation no programming is needed and commands in TagUI are similar to **click image.PNG,** where **image.PNG** is the image that we captured and want to be clicked by TagUI. The other commands are similar to the one described above.

```
#Automation done for sap and outlook together

class Action_tagui(Action):
    def name(self):
        return "action_tagui"

    def run(self, dispatcher, tracker, domain):
        username = tracker.get_slot('username')
        email = tracker.get_slot('email')
        system = tracker.get_slot('system_name')

#initiate visual automation
```

```
13.         r.init(visual_automation = True, chrome_browser = False)
14. #show desktop
15.         r.rclick ('start.PNG')
16.         r.click ('showDesktop.PNG')
17. #open SAP GUI
18.         r.dclick ('sap-icon.PNG')
19.         r.wait(5)
20. # variable from RASA
21.         r.type ('system_field.PNG', system)
22.         r.keyboard('[enter]')
23.         r.wait(5)
24. #login into the system - static values
25.         r.type ('user.png', 'SAP*[tab]')
26.         r.keyboard('[shift]a')
27.         r.keyboard('ppl1ance[enter]')
28. #enter the transaction
29.         r.click ('transaction.PNG')
30.         r.wait (1)
31.         r.keyboard('suo1[enter]')
32.         r.wait(2)
33. #Rasa username variable
34.         r.type('user-field.PNG', username)
35.         r.click('change-user.PNG')
36.         r.wait(3)
37.         r.click('logon-data')
38. #new password
39.         r.type('new-password.PNG', 'Password001!')
40. #repeat Password
41.         r.type('repeatPassword.PNG', 'Password001!')
42. # put automation for save button, in our case we do not save
43. # as we do a lot of tests and everytime it is necessary to reset the password
44.
45.         r.wait (2)
46. # Logoff from the system
47.         r.click('system.PNG')
48.         r.click('close.PNG')
49.         r.click('yesLOGOFF.PNG')
50.         r.click('close2.PNG')
51. # Automate Outlook Application
52. #open the Application
53.         r.dclick('outlook.PNG')
54.         r.wait(6)
55. #new mail
56.         r.click ('newMail.PNG')
57.         r.wait(2)
58. #send mail to variable mail from RASA
59.         r.type('mailTo.PNG', email)
60.         r.type('mailSubject.PNG', 'Password Reset[tab]')
61.         r.keyboard('Hello[enter]' 'Your new password is: P@ssword[enter]''Best Re
    gards[enter]''Your Bot' )
62.         r.click ('sendMail.PNG')
63. #click minimize.PNG
64.         r.wait(7)
65. #close outlook
66.         r.click ('outlookClose.PNG')
67.         r.close()
```

The script above is the script that is triggered by the NLU action from Rasa. In the code there are two sections. The first section automates SAP GUI and the second one automates Outlook (line 51). The code from line 51 is in correlation with the use case diagram that was presented in Figure 7.

After the script is started by the chatbot is TagUI which should run and do the automation in the backend. Below will be presented an output from the Outlook automation part and how TagUI handles the steps of the script.

```
1.  [tagui] START  - listening for inputs
2.
3.  [tagui] INPUT  - [1] present C:/Users/Diploma/Desktop/virtualenv_diploma/outlook.
    PNG
4.  [tagui] ACTION - present C:/Users/Diploma/Desktop/virtualenv_diploma/outlook.PNG

5.  [tagui] OUTPUT - [1] SUCCESS
6.
7.  [tagui] INPUT  - [2] dclick C:/Users/Diploma/Desktop/virtualenv_diploma/outlook.P
    NG
8.  [tagui] ACTION - dclick C:/Users/Diploma/Desktop/virtualenv_diploma/outlook.PNG
9.  [log] DOUBLE CLICK on L[189,275]@S(0) (1111 msec)
10. [tagui] OUTPUT - [2] SUCCESS
11.
12. [tagui] INPUT  - [3] present C:/Users/Diploma/Desktop/virtualenv_diploma/newMail.
    PNG
13. [tagui] ACTION - present C:/Users/Diploma/Desktop/virtualenv_diploma/newMail.PNG

14. [tagui] OUTPUT - [3] SUCCESS
15.
16. [tagui] INPUT  - [4] click C:/Users/Diploma/Desktop/virtualenv_diploma/newMail.PN
    G
17. [tagui] ACTION - click C:/Users/Diploma/Desktop/virtualenv_diploma/newMail.PNG
18. [log] CLICK on L[224,212]@S(0) (604 msec)
19. [tagui] OUTPUT - [4] SUCCESS
```

The above script is an extract of the first steps running through TagUI. As it can be seen from the output TagUI tries to find the elements identified in the screen through images and than through our commands will perform those actions in the image section. Moreover it can be seen that if a step failed or not and the time it takes for a step to be completed.

We provided an output of TagUI running rather than images of the automation as it would not be visible from the images how the automation is running.

When the automation will finish either with success or error in the directory where all your TagUI content is located will be generated a file called:

name_of_your_script.log

```
1.  [RPA][0] - listening for inputs
2.  [RPA][1] - exist_result = exist('outlook.PNG').toString()
3.  [RPA][1] - listening for inputs
4.  [RPA][2] - dump exist_result to rpa_python.txt
5.  [RPA][2] - listening for inputs
6.  [RPA][3] - dclick outlook.PNG
```

The content of the *log* file will store all the steps and the commands that were run during the automation and where the automation might have failed if that is the case.

In the case of troubleshooting your automation from TagUI perspective a feature called live mode comes really handy. In live mode is possible to run commands one by one and see how TagUI is performing and where it is failing.

4.4 Benchmarking

The aim of this section is to test how our NLU model behaves with the training data and the selected components. In order to achieve this, we will use some proprietary commands that Rasa provides through their tool. Moreover, other benchmarking made on different datasets will be used in order to see how Rasa performs compared to other tools in the market.

Through the evaluation process of Rasa, a lot can be achieved such as comparing the pipelines [44]. We will mainly focus on intent classification and response selection. In our working Rasa directory, there will be created a new folder that will contain the results of the evaluation.
The command that will be used to get Rasa model evaluation is:

```
1. rasa test
```

4.4.1 NLU model

In this section the natural language understanding model will be tested in order to verify its performance. There will be two subsections of interest the intent classification and response selection.

What is important to note is that the method of testing is by splitting our data and creating a subset of the data in our *json* file. To achieve this please run: [45]

```
1. rasa data split nlu
```

the above command will create a folder with test data that is used to test our model.

To test the NLU model the following command will be used: [45]

```
1. rasa test nlu -u train_test_split/test_data.md --model models/nlu-20200401-
   145833.tar.gz
```

- Intent classification

Running the above command will generate a report that contains a confusion matrix and confidence histogram for our model.
The report contains precision of the model, recall and f1 measure for each intent and entity. The confusion matrix will give us an overview which intents are confused for others and an error file in *json* format will be generated that will contain the incorrectly predicted samples.

Running the above command created a terminal output and *results* that we mentioned previously.
For our model no file called *errors.json* was generated which means that there are no samples incorrectly predicted.
In Figure 8 it is shown the histogram that was generated in the results folder. This histogram as it is indicated on the top of the picture shows the confidence distribution of the intent prediction. It is clear that there were no misses in our predictions and the confidence is above 95%.

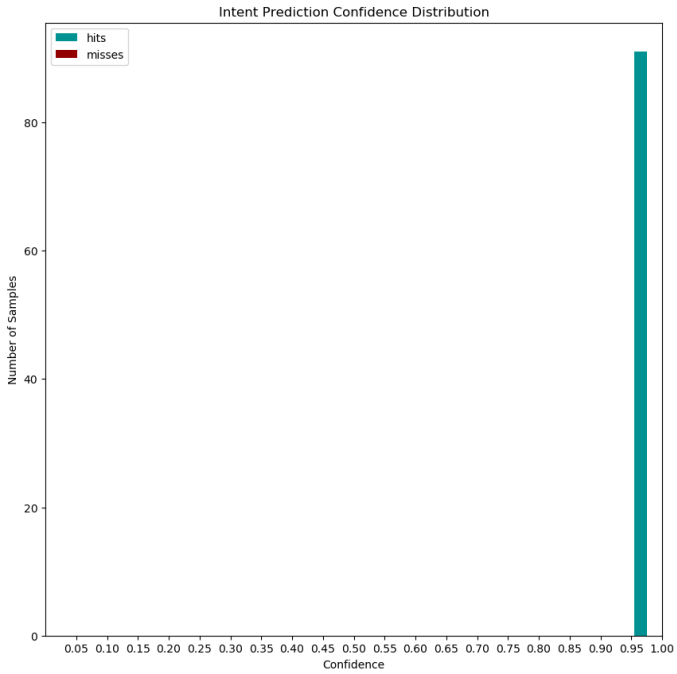


Figure 8: Histogram

Moreover, the above histogram is represented also in json format with all the predicted intents. This is represented in the *json* below:


```
{
  "password": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 6,
    "confused_with": {}
  },
  "system_name": {
    "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": 5,
```

```
    "confused_with": {}
  },
  "greet": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 5,
```

The above *json* is also used in order to create the confusion matrix that is in Figure 9. The output shown above is not complete and the rest will be available for cross checking in the Apendix section.

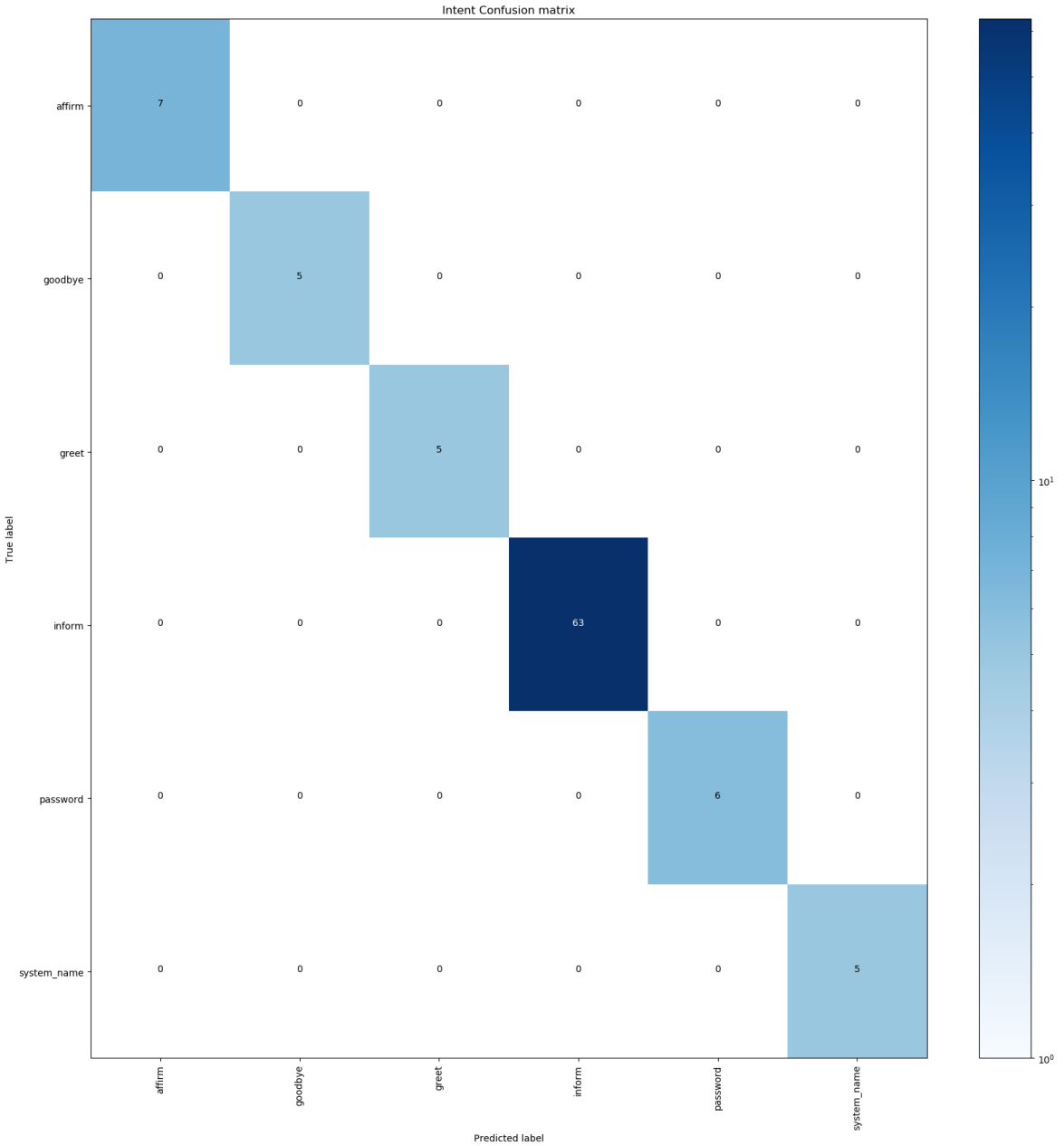


Figure 9: Confusion Matrix NLU Model

- Response Selection

Another output of the evaluation proces is the response selection. This can also be generated in *json* format by appending the flag *–report* to the evaluation command run in the previous step.
As it was visible from the json in the previous step the output includes a number of indicators:

1. F1 score – which in our evaluation is 1 for all of the intents
2. Precision – which is 1 in our evaluation
3. Recall – having the value of 1 in our evaluation

Apart from the three indicators above there is also calculated the weighted average of the model evaluated as can be seen from the *json* below.

```
"accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 91
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 91
  }
}
```

4.4.2 Core model

The second part of benchmarking will include the evaluation of our core model which is responsible for the dialogue management as it is previously mentioned.
The procedure followed is the same but in this case it is important to select our core model to evaluate. This can be done by running the following command:

```
rasa test core --stories test_stories.md --out results
```

the above command will create a file called failed stories. This will count if a story failed if at least one of the actions was predicted incorrectly.
The output received in our case for the *failed_stories* file is shown in Figure 10. From the file it is clear that all of our stories were able to pass successfully meaning that all our actions can be predicted.

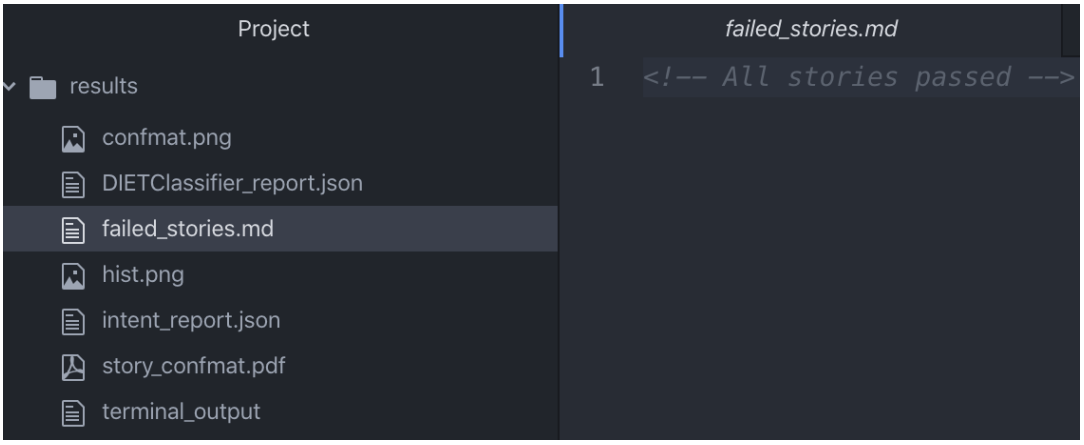


Figure 10: Failed Stories

Apart from the failed stories, as in the previous step with NLU in the core model there is also a confusion matrix generated.
The output of terminal creates which is displayed in Figure 11.

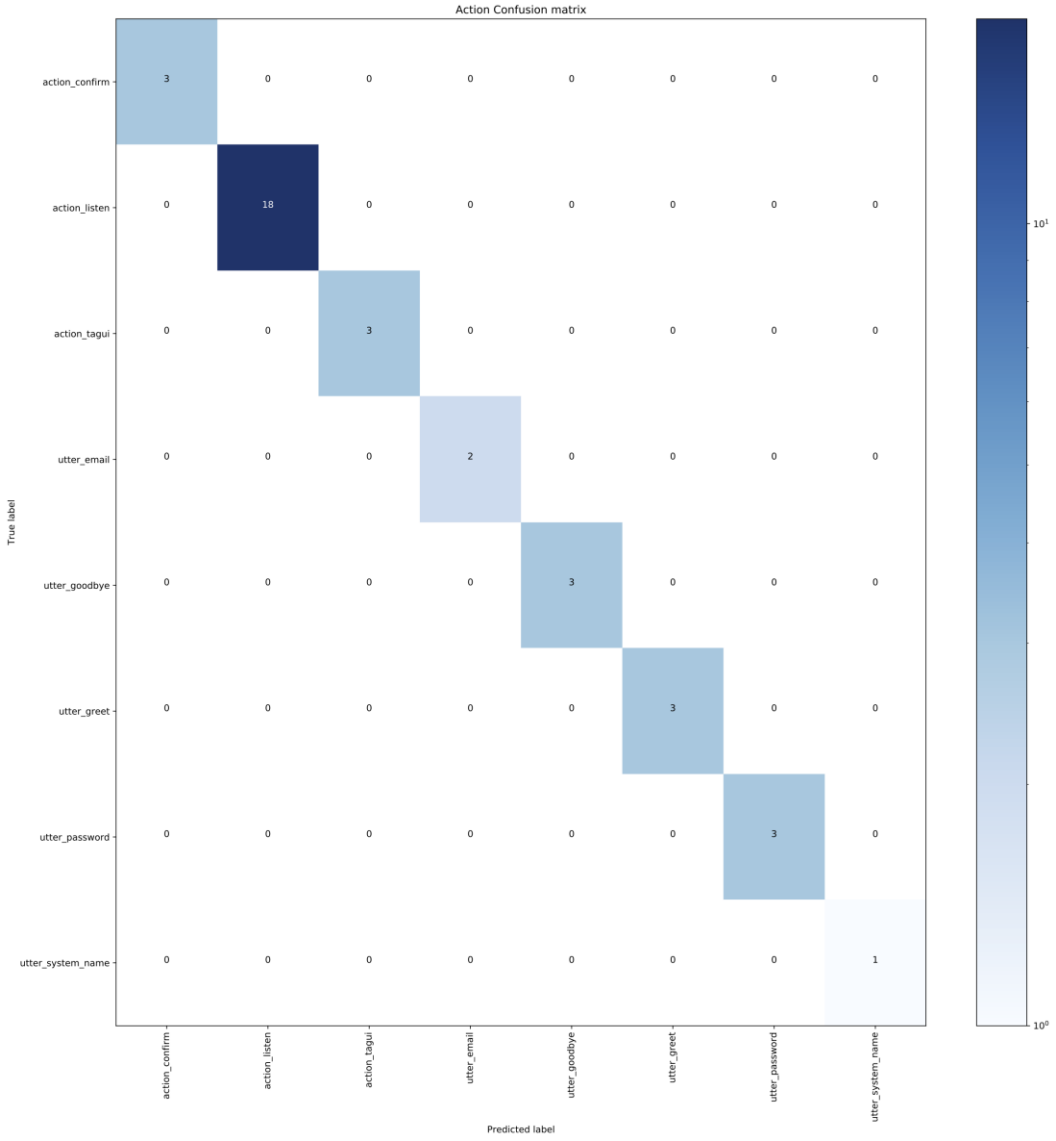


Figure 11: Confusion Matrix Core Model

From the confusion matrix Figure 11 we can come to conclusions about our core model. The important values would be:

Precision of 1 meaning that there are no false negatives, recall of value 1 means that there are no false positives leading to having a F1 score of 1.

As it was explained from the literature review the f1 score gives the accuracy of our testing. In our case we could say that the testing is good as the result for the f1 score is 1.

5 Results and Discussion

In practical part we were able to demonstrate the implementation that includes the cooperation of two tools, one for conversation management and the other for automation. Through the experiments that were made is was possible to prove that the two tools could be merged and achieve the end result.
Our domain was user support and specifically password change of a system such as SAP GUI, so this narrows down the domain that the NLU tool was operating and this can justify the results in the benchmarking part. What is useful from this implementation is that the NLU part can be extended to include a lot of other processes inside the same domain.

It can be noted that the combination of RPA and NLU is possible and what is more this combination can be achieved using Open Source software. The benefit from this combination can be really handy to a lot of companies who want to increase customer satisfaction and the speed of their processes.

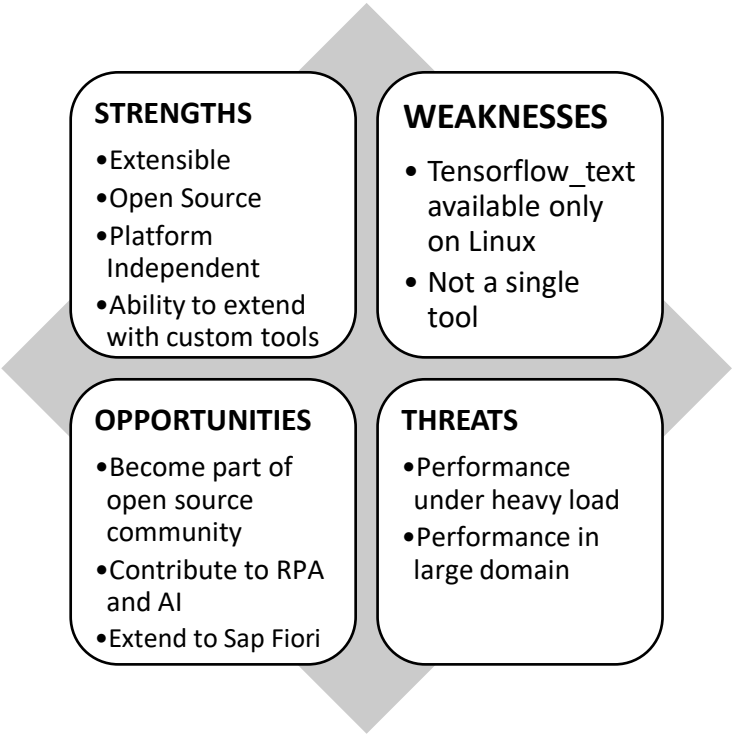
Throughout literature review was stressed out that in order to achieve good results with RPA the process needs to be clearly defined. In practical part the process that we selected was broken down in small pieces and every step was clear. This gives the ability to define the RPA steps efficiently and the RPA will not fail. Nevertheless, RPA is part of other systems and automates other systems and everything needs to be taken into account in order to achieve the end result.
As mentioned, the provided solution is a proof of concept to prove that the integration of open source tools only can achieve the end result. The product that came out of this thesis has the necessity to be developed some more, in terms of user interface, not as functionality.

5.1 SWOT Analysis

This section will be dedicated towards the pros and cons of the tool and the implementation. This will be detailed through a SWOT analysis, which will look into the implementation in four different areas. The areas of the analysis are:

- Strength
- Weaknesses
- Opportunities
- Threats

The following areas with bullet points are presented in the chart below. All the bullet points for every section will be explained in order to give the broader audience what the author means with each one of them.



- **Strengths**

What we see as a strength for the implementation is that it can be extended with other domains, or with other tasks inside the same domain. Both of the tools are able to work together in different automations including browser and other UI tools not dependent on the operating system.
The other strength that we take pride on is the Open Source foundation of the tools. This gives the ability to cut down costs and be backed by a community passionate on what they are doing. This also leads to the next strength which is extension with other tools or libraries that may be custom made or open source as well.
We mentioned before that the tools are platform independent a feature which is useful especially since nowadays companies are not strictly linked to one platform or Operating System.

- **Opportunities**

What we see as an opportunity is being part of the open source community. Helping in the community helps also the users of the tool as in this way the tools improve, and possible bugs are fixed. Moreover, the contribution apart from the improvement of the tools will help the emerging fields of RPA and AI which are much bigger than TagUI and Rasa.
SAP as a company has a new version of SAP GUI, not only for desktop but also a browser based one. In order to automate the desktop version no change needs to be done in terms of the scripts just the images need to be updated as the SikuliX tool will not be able to recognize the fields.
Extending the implementation to SAP Fiori which is browser based could leverage also the potential that TagUI has to automate web-based applications. TagUI is really powerful tool

also in the domain of web automations. This implementation would also overcome the limitation of *tensorflow_text* not being available for Windows OS.

- **Weaknesses**

It was mentioned previously that the tools are cross platform and it is true. Nevertheless, there are underlying components which are not compatible with every operating system. Such component is *tensorflow_text* which is not available in windows operating system. This can be a bottleneck in case that your domain is large and you want to use other pipelines which are based on TensorFlow. This would pose a threat on TagUI Windows based automations.
Another weakness of the implementation is that it is based on two different tools. It would be better if the whole implementation was a single tool.

- **Threats**

What is not tested as part of this thesis is how the tools perform under heavy load, and how this will be overcome in real production environments. For Rasa there are deployments based on distributed systems which might not work as well for TagUI or at least not for every case. When in production multiple users might contact the chatbot and the RPA will have to be ready to process all the requests.

We stated earlier that the domain where Rasa works can be extended not to include just password change, but also other automations included in user support systems. This poses a threat on the dataset and the performance of the NLU part. A lot of preparatory work needs to be done in order to achieve successful implementations and an increased user satisfaction.

5.2 Comparison to other tools

In literature review was pointed out the importance of AI and the speed this sector is growing the recent years along with RPA. There are e number of articles describing the usage of conversational RPA or RPA + AI in user support systems. As of the moment that this diploma is being written there is no production ready tool that offers the same experience. It is certain that there are companies that leverage the power of open source community and have built their in-house automations based on TagUI or Rasa for the NLU part.

Throughout this thesis the author had the chance to speak to the founder of TagUI, and one of the main contributors to the community of TagUI. The information that he provided was that are companies which use Rasa and TagUI in their daily operations, but still these companies are not sharing the automations they have made with the general public. It could be possible that in the near future their product will be generally available but at the moment it is not.
As a result, comparison with other tools that offer a full suite combination is nearly impossible. The author does not take into consideration premium tools that are available when is referring to other cognitive RPA tools as there is a number of tools that nowadays offer cognitive RPA as part of their features.

6 Conclusion

Throughout the previous five sections starting with literature review and going on with practical part we tried to prove and demonstrate if open source software provides the tools to integrate RPA with AI and build conversational RPA. Out of two tools, Rasa and TagUI, it was managed to create a full working pipeline for user support systems to automate password change in systems such as SAP GUI.
The performance of the tools was very good and the community behind really helpful. The author had the chance to communicate with some of the creators and regarding questions the community was really helpful and fast.
After months of work with both tools a final PoC was created. What is to be taken into consideration is the SWOT analysis in section 5.1.

Furthermore, user support systems provide a suitable ecosystem for RPA automation, as long as the processes are simple and detailed in every step so there are no failures in design.
The process that was automated in this work was simple and straightforward, but there is space for extensibility in both domains RPA and NLU.

Even though, modifications in the code of the tools were not made we hope that this is extended by bigger companies or other contributors to make both tools in a single application.

7 Bibliography

[1] LERNER, Josh a Jean. TIROLE. The open source movement: Key research questions. European economic review,. *European Economic Review*. Elsevier, 2001, , 819-826.

[2] STALLMAN, Richard. *Viewpoint Why open source misses the point of free software. Communications of the ACM, .* 2009.

[3] WHEELER, David. *Why Open Source Software*. 2015. [Online] https://dwheeler.com/oss_fs_why.html

[4] ITIC. *ITIC 2019 Global Server Hardware, Server OS Reliability Report*. 2019.

[5] Web Server Survey. *Netcraft*. 2020. [Online] https://news.netcraft.com/archives/category/web-server-survey

[6] FORBES. Linux Rules Supercomputers. *Forbes*. 2005. [Online] https://www.forbes.com/2005/03/15/cz_dl_0315linux.html#5f8879bd65fe

[7] JAAP-HENK HOEPMAN, Bart. Increased security through open source. *Communications of the ACM*. ACM New York, NY, USA, 2007, **50**(1), 79--83.

[8] SCHRYEN, Guido. *Is open source security a myth*. 2011.

[9] DELGADO, Arvie. *Roro - Free RPA Software*. 2018. [Online] https://www.roroscript.com/

[10] *OpenExpoEurope*. 2018. [Online] https://openexpoeurope.com/the-future-of-the-open-source-survey-the-new-stack-the-linux-foundation-and-the-todo-group/

[11] Open Source Smart RPA . *Automagica*. b.r. [Online] https://automagica.com/

[12] *TagUI*. 2018. [Online] https://www.aisingapore.org/resources/tagui/

[13] HOCKE, Raimund. *SikuliX*. 2017. [Online] http://sikulix.com/

[14] PhantomJS - Scriptable Headless Browser . *PhantomJS*. 2018. [Online] https://phantomjs.org/

[15] JOUANNEAU, Laurent. *SlimerJS*. 2018. [Online] https://slimerjs.org/

[16] BAYLDON, Jason. *TaskT*. 2018. [Online] http://www.taskt.net/

[17] WILLCOCKS, Leslie, Mary LACITY a Andrew. CRAIG. *The IT function and robotic process automation*. 2015.

[18] VAN DER AALST, Wil, Martin BICHLER a Armin. HEINZL. *Robotic process automation. .* 2018.

[19] ASATIANI, Aleksandre a Esko. PENTTINEN. Turning robotic process automation into commercial success–Case OpusCapita. Journal of Information Technology Teaching Cases,. *Journal of Information Technology Teaching Cases* . 2016, , 67–74.

[20] WILLCOCKS, Leslie a Mary. LACITY. *Service automation robots and the future of work. SB Publishing*. 2016.

[21] LACITY, Mary. A new approach to automating services. *MIT Sloan Management Review*. Massachusetts Institute of Technology, 2017.

[22] LAMBERTON, Chris, Damiano BRIGO a Dave HOY. *Impact of Robotics, RPA and AI on the insurance industry: challenges and opportunities*. November 29, 2017.

[23] LESLIE WILLCOCKS, Andrew. *Robotic Process Automation at Xchanging*. 2015.

[24] WILLCOCKS, Leslie, Mary LACITY a Andrew. CRAIG. *Robotic Process Automation at Telefónica O2*. 2015.

[25] POOSAPATI, Vijaya, Vedavathi KATNENI a Vijaya MANDA. *Cognitive Automation in Industry–Design Principles and Case Study*. *International Journal of Applied Engineering Research*, . 2018.

[26] NOOR, Ahmed. *Potential of cognitive computing and cognitive systems*. *Open Engineering*,. 2015.

[27] KOEDINGER, Kenneth. *Cognitive Tutors: Technology Bringing Learning Science to the Classroom*. 2006.

[28] GUDIVADA, Venkat. Cognitive Computing: Concepts, Architectures, Systems, and Applications. *Handbook of Statistics*. Elsevier, 2016, **35**.

[29] STRICKLAND, Eliza. *Layoffs at Watson Health reveal IBM's problem with AI*. IEEE Spectrum, 2018.

[30] BUTLER, Tom a Leona O'BRIEN. *Artificial Intelligence for Regulatory Compliance: Are We There Yet?*. Journal of Financial Compliance, 2019, (31), 44-59.

[31] JAMES KISNER, CFA,David. Creating Shareholder Value with AI? Not so Elementary, My Dear Watson. *Psyccritiques*. Jefferies Franchise Note, 2018.

[32] DILEK YıLMAZ BÖREKçI, Özlen. A Conceptual Study on RPAs as of Intelligent Automation. *International Conference on Intelligent and Fuzzy Systems*. 2019.

[33] ENRÍQUEZ JG, Jiménez-Ramírez. Robotic Process Automation: A Scientific and Industrial Systematic Mapping Study. *IEEE Access*. 2020.

[34] RAJ, Sumit. *Building Chatbots with Python : Using Natural Language Processing and Machine Learning Date*. Apress L. P, 2018.

[35] CATAL, Cagatay. Performance evaluation metrics for software fault prediction studies. *Acta Polytechnica Hungarica*. 2012, **9**(4), 193--206.

[36] SAMMUT, Claude. *Encyclopedia of machine learning and data mining*. Springer Publishing Company, Incorporated, 2017.

[37] TOM BOCKLISCH, Joey. *Rasa: Open Source Language Understanding andDialogue Management*. Cornell University, 2017.

[38] RASA. Components. *Rasa*. 2020. [Online] https://rasa.com/docs/rasa/nlu/choosing-a-pipeline

[39] SATHIYASEELAN BALASUNDARAM, Sirish. *A structured approach to implementing Robotic Process Automation in HR*. 2008.

[40] GARTNER. *Gartner*. 2019. [Online] https://www.gartner.com/smarterwithgartner/bots-gain-importance-in-gartner-service-technologies-bullseye/

[41] KUNGWANI, Swati. *InformationAge*. 2019. [Online] https://www.information-age.com/rpa-customer-service-123479794/

[42] AIMULTIPLE. Blog.aimultiple.com. *AIMultiple*. 2020. [Online] https://blog.aimultiple.com/rpa-case-studies/

[43] TECHNOLOGIES, Rasa. RasaNLU. *RASA*. 2020. [Online] https://rasa.com/docs/rasa/nlu/about/

[44] TECHNOLOGIES, Rasa. Evaluating Models. *Rasa*. 2020. [Online]https://rasa.com/docs/rasa/user-guide/evaluating-models/#evaluating-an-nlu-model

[45] RASA. Testing Your Assistant. *Rasa* . 2020.[Online] https://rasa.com/docs/rasa/user-guide/testing-your-assistant/#evaluating-an-nlu-model

8 Appendix

- Rasa NLU Training Data

```
{
  "rasa_nlu_data": {
    "common_examples": [
      {
        "text": "hey",
        "intent": "greet",
        "entities": []
      },
      {
        "text": "howdy",
        "intent": "greet",
        "entities": []
      },
      {
        "text": "hey there",
        "intent": "greet",
        "entities": []
      },
      {
        "text": "hello",
        "intent": "greet",
        "entities": []
      },
      {
        "text": "hi",
        "intent": "greet",
        "entities": []
      },
      {
        "text": "i want to change the password for the login",
        "intent": "password",
        "entities": []
      },
      {
        "text": "login forget",
        "intent": "password",
        "entities": []
      },
      {
        "text": "yes",
        "intent": "affirm",
        "entities": []
      },
      {
        "text": "yep",
        "intent": "affirm",
        "entities": []
      },
      {
        "text": "yeah",
        "intent": "affirm",
        "entities": []
      },
      {
        "text": "password forgotten",
        "intent": "password",
        "entities": []
```

```
    },
    {
      "text": "bye",
      "intent": "goodbye",
      "entities": []
    },
    {
      "text": "goodbye",
      "intent": "goodbye",
      "entities": []
    },
    {
      "text": "good bye",
      "intent": "goodbye",
      "entities": []
    },
    {
      "text": "stop",
      "intent": "goodbye",
      "entities": []
    },
    {
      "text": "end",
      "intent": "goodbye",
      "entities": []
    },
    {
      "text": "cannot remember my password",
      "intent": "password",
      "entities": []
    },
    {
      "text": "indeed",
      "intent": "affirm",
      "entities": []
    },
    {
      "text": "that's right",
      "intent": "affirm",
      "entities": []
    },
    {
      "text": "ok",
      "intent": "affirm",
      "entities": []
    },
    {
      "text": "great",
      "intent": "affirm",
      "entities": []
    },
    {
      "text": "my username is jake",
      "intent": "inform",
      "entities": [
        {
          "start": 15,
          "end": 21,
          "value": "jake",
          "entity": "username"
        }
      ]
    },
```

```
    {
      "text": "my login alket",
      "intent": "inform",
      "entities": [
        {
          "start": 9,
          "end": 14,
          "value": "alket",
          "entity": "username"
        }
      ]
    },
    {
      "text": "my username is admin",
      "intent": "inform",
      "entities": [
        {
          "start": 15,
          "end": 20,
          "value": "admin",
          "entity": "username"
        }
      ]
    },
    {
      "text": "michael",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 7,
          "value": "michael",
          "entity": "username"
        }
      ]
    },
    {
      "text": "lucy is my username",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 4,
          "value": "lucy",
          "entity": "username"
        }
      ]
    },
    {
      "text": "i need to change my login",
      "intent": "password",
      "entities": []
    },
    {
      "text": "i forgot my login details",
      "intent": "password",
      "entities": []
    },
    {
      "text": "my username is john",
      "intent": "inform",
      "entities": [
        {
```

```
            "start": 15,
            "end": 19,
            "value": "john",
            "entity": "username"
          }
        ]
      },
      {
        "text": "braian",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 6,
            "value": "braian",
            "entity": "username"
          }
        ]
      },
      {
        "text": "marco",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "marco",
            "entity": "username"
          }
        ]
      },
      {
        "text": "jan",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "jan",
            "entity": "username"
          }
        ]
      },
      {
        "text": "jack is my username",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "jack",
            "entity": "username"
          }
        ]
      },
      {
        "text": "admin",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "admin",
```

```
            "entity": "username"
          }
        ]
      },
      {
        "text": "admin is username",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "admin",
            "entity": "username"
          }
        ]
      },
      {
        "text": "username admin",
        "intent": "inform",
        "entities": [
          {
            "start": 9,
            "end": 14,
            "value": "admin",
            "entity": "username"
          }
        ]
      },
      {
        "text": "alket",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "alket",
            "entity": "username"
          }
        ]
      },
      {
        "text": "marco is my username",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "marco",
            "entity": "username"
          }
        ]
      },
      {
        "text": "antonio is my username",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 7,
            "value": "antonio",
            "entity": "username"
          }
        ]
```

```
    },
    {
      "text": "andrea is my username",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 6,
          "value": "andrea",
          "entity": "username"
        }
      ]
    },
    {
      "text": "my username is donald_duck",
      "intent": "inform",
      "entities": [
        {
          "start": 15,
          "end": 26,
          "value": "donald_duck",
          "entity": "username"
        }
      ]
    },
    {
      "text": "kejdi is my username",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 5,
          "value": "kejdi",
          "entity": "username"
        }
      ]
    },
    {
      "text": "milos",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 5,
          "value": "milos",
          "entity": "username"
        }
      ]
    },
    {
      "text": "my username is developer",
      "intent": "inform",
      "entities": [
        {
          "start": 15,
          "end": 24,
          "value": "developer",
          "entity": "username"
        }
      ]
    },
    {
      "text": "my username is mirko",
```

```
        "intent": "inform",
        "entities": [
          {
            "start": 15,
            "end": 20,
            "value": "mirko",
            "entity": "username"
          }
        ]
      },
      {
        "text": "username is developer1",
        "intent": "inform",
        "entities": [
          {
            "start": 12,
            "end": 22,
            "value": "developer1",
            "entity": "username"
          }
        ]
      },
      {
        "text": "john",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "john",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Gavin",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "Gavin",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Francis",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 7,
            "value": "Francis",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Igor",
        "intent": "inform",
        "entities": [
          {
```

```
            "start": 0,
            "end": 4,
            "value": "Igor",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Peter",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "Peter",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Chaney",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 6,
            "value": "Chaney",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Anthony",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 7,
            "value": "Anthony",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Malik",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "Malik",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Leonard",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 7,
            "value": "Leonard",
```

```
            "entity": "username"
          }
        ]
      },
      {
        "text": "Gil",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "Gil",
            "entity": "username"
          }
        ]
      },
      {
        "text": "William",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 7,
            "value": "William",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Dorian",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 6,
            "value": "Dorian",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Sybill",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 6,
            "value": "Sybill",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Tana",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "Tana",
            "entity": "username"
          }
        ]
```

```
    },
    {
      "text": "Kimberly",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 8,
          "value": "Kimberly",
          "entity": "username"
        }
      ]
    },
    {
      "text": "Ava",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 3,
          "value": "Ava",
          "entity": "username"
        }
      ]
    },
    {
      "text": "Jaquelyn",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 8,
          "value": "Jaquelyn",
          "entity": "username"
        }
      ]
    },
    {
      "text": "Petra",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 5,
          "value": "Petra",
          "entity": "username"
        }
      ]
    },
    {
      "text": "Lila",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 4,
          "value": "Lila",
          "entity": "username"
        }
      ]
    },
    {
      "text": "Meghan",
```

```
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 6,
            "value": "Meghan",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Vera",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "Vera",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Emma",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "Emma",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Emi",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "Emi",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Emily",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "Emily",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Yen",
        "intent": "inform",
        "entities": [
          {
```

```
            "start": 0,
            "end": 3,
            "value": "Yen",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Sonia",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 5,
            "value": "Sonia",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Kathleen",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 8,
            "value": "Kathleen",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Alea",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "Alea",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Olga",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 4,
            "value": "Olga",
            "entity": "username"
          }
        ]
      },
      {
        "text": "Chanda",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 6,
            "value": "Chanda",
```

```
            "entity": "username"
          }
        ]
      },
      {
        "text": "my email address is alketcami@gmail.com",
        "intent": "inform",
        "entities": [
          {
            "start": 20,
            "end": 39,
            "value": "alketcami@gmail.com",
            "entity": "email"
          }
        ]
      },
      {
        "text": "alketshabani@outlook.com",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 24,
            "value": "alketshabani@outlook.com",
            "entity": "email"
          }
        ]
      },
      {
        "text": "fbozgo@smartworknet.eu",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 22,
            "value": "fbozgo@smartworknet.eu",
            "entity": "email"
          }
        ]
      },
      {
        "text": "gjaupi@alteanet.it",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 18,
            "value": "gjaupi@alteanet.it",
            "entity": "email"
          }
        ]
      },
      {
        "text": "shabalket@gmail.com",
        "intent": "inform",
        "entities": [
          {
            "start": 0,
            "end": 19,
            "value": "shabalket@gmail.com",
            "entity": "email"
          }
        ]
```

```
    },
    {
      "text": "kgolemi@alteanet.it",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 19,
          "value": "kgolemi@alteanet.it",
          "entity": "email"
        }
      ]
    },
    {
      "text": "eangjeli@alteanet.it",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 20,
          "value": "eangjeli@alteanet.it",
          "entity": "email"
        }
      ]
    },
    {
      "text": "grobo@smartworknet.eu",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 21,
          "value": "grobo@smartworknet.eu",
          "entity": "email"
        }
      ]
    },
    {
      "text": "eangjeli@smartworknet.eu",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 24,
          "value": "eangjeli@smartworknet.eu",
          "entity": "email"
        }
      ]
    },
    {
      "text": "akoci@smartworknet.eu",
      "intent": "inform",
      "entities": [
        {
          "start": 0,
          "end": 21,
          "value": "akoci@smartworknet.eu",
          "entity": "email"
        }
      ]
    },
    {
      "text": "system is NPL",
```

```
        "intent": "system_name",
        "entities": [
          {
            "start": 10,
            "end": 13,
            "value": "NLP",
            "entity": "system"
          }
        ]
      },
      {
        "text": "OC2",
        "intent": "system_name",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "oc2",
            "entity": "system"
          }
        ]
      },
      {
        "text": "OMD",
        "intent": "system_name",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "OMD",
            "entity": "system"
          }
        ]
      },
      {
        "text": "OMT",
        "intent": "system_name",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "OMT",
            "entity": "system"
          }
        ]
      },
      {
        "text": "OSP",
        "intent": "system_name",
        "entities": [
          {
            "start": 0,
            "end": 3,
            "value": "OSP",
            "entity": "system"
          }
        ]
      }
    ]
  }
}
```

- **Rasa NLU Stories**

```
## Generated Story -2631642282788300614
* greet
    - utter_greet
* password
    - utter_password
* inform{"username": "alket"}
    - slot{"username": "alket"}
    - utter_system_name
* system_name{"system": "mxd"}
    - slot{"system": "mxd"}
    - utter_email
* email{"email":"alket@outlook.com"}
    - slot{"email":"alket@outlook.com"}
    - action_confirm
* affirm
    - utter_goodbye
    - action_tagui
* goodbye


## second story
* greet
    - utter_greet
* password
    - utter_password
* inform{"username": "john", "system":"NPL", "email": "johndoe@gmail.com"}
    - slot{"username": "john"}
    - slot{"system": "NPL"}
    - slot{"email":"johndoe@gmail.com"}
    - action_confirm
* affirm
    - utter_goodbye
    - action_tagui
* goodbye


## third story
* greet
    - utter_greet
* password
    - utter_password
* inform{"username": "john", "system":"NPL"}
    - slot{"username": "john"}
    - slot{"system": "NPL"}
    - utter_email
* email{"email": "johndoe@gmail.com"}
    - slot{"email":"johndoe@gmail.com"}
    - action_confirm
* affirm
    - utter_goodbye
    - action_tagui
* goodbye
```

- **Rasa NLU Custom Actions**

```
# This files contains your custom actions which can be used to run
# custom Python code.
#
# See this guide on how to implement these action:
```

```
5.  # https://rasa.com/docs/rasa/core/actions/#custom-actions/
6.
7.
8.  # This is a simple example for a custom action which utters "Hello World!"
9.
10. # from typing import Any, Text, Dict, List
11. #
12.  #import tagui
13. import rpa as r
14. from rasa_sdk import Action, Tracker
15. from rasa_sdk.executor import CollectingDispatcher
16. #
17. #
18. # class ActionHelloWorld(Action):
19. #
20. #     def name(self) -> Text:
21. #         return "action_hello_world"
22. #
23. #     def run(self, dispatcher: CollectingDispatcher,
24. #             tracker: Tracker,
25. #             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
26. #
27. #         dispatcher.utter_message(text="Hello World!")
28. #
29. #         return []
30.
31.
32. class Action_confirm(Action):
33.     def name(self):
34.         return "action_confirm"
35.
36.
37.     def run(self, dispatcher, tracker, domain):
38.         username = tracker.get_slot('username')
39.         email = tracker.get_slot('email')
40.         system = tracker.get_slot('system')
41.         response = """ we will reset the password for username {}, to the system
    {} and send an email to {}.Can you confirm?""".format(username,system, email)
42.         dispatcher.utter_message(response)
43.
44.
45. #Automation done for sap and outlook together
46.
47. class Action_tagui(Action):
48.     def name(self):
49.         return "action_tagui"
50.
51.     def run(self, dispatcher, tracker, domain):
52.         username = tracker.get_slot('username')
53.         email = tracker.get_slot('email')
54.         system = tracker.get_slot('system')
55.
56. #initiate visual automation
57.         r.init(visual_automation = True, chrome_browser = False)
58. #show desktop
59.         r.rclick ('start.PNG')
60.         r.click ('showDesktop.PNG')
61. #open SAP GUI
62.         r.dclick ('sap-icon.PNG')
63.         r.wait(5)
64. # variable from RASA
65.         r.type ('system_field.PNG', system)
66.         r.keyboard('[enter]')
```

```
        r.wait(5)
#login into the system - static values
        r.type ('user.png', 'SAP*[tab]')
        r.keyboard('[shift]a')
        r.keyboard('ppl1ance[enter]')
#enter the transaction
        r.click ('transaction.PNG')
        r.wait (1)
        r.keyboard('suo1[enter]')
        r.wait(2)
#Rasa username variable
        r.type('user-field.PNG', username)
        r.click('change-user.PNG')
        r.wait(3)
        r.click('logon-data')
#new password
        r.type('new-password.PNG', 'Password001!')
#repeat Password
        r.type('repeatPassword.PNG', 'Password001!')
# put automation for save button, in our case we do not save
# as we do a lot of tests and everytime it is necessary to reset the password

        r.wait (2)
# Logoff from the system
        r.click('system.PNG')
        r.click('close.PNG')
        r.click('yesLOGOFF.PNG')
        r.click('close2.PNG')
# Automate Outlook Application
#open the Application
        r.dclick('outlook.PNG')
        r.wait(6)
#new mail
        r.click ('newMail.PNG')
        r.wait(2)
#send mail to variable mail from RASA
        r.type('mailTo.PNG', email)
        r.type('mailSubject.PNG', 'Password Reset[tab]')
        r.keyboard('Hello[enter]' 'Your new password is: P@ssword[enter]''Best Re
    gards[enter]''Your Bot' )
        r.click ('sendMail.PNG')
#click minimize.PNG
        r.wait(7)
#close outlook
        r.click ('outlookClose.PNG')
        r.close()
```

- **Response Selection and intent classification**

```
{
  "password": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 6,
    "confused_with": {}
  },
  "system_name": {
    "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": 5,
    "confused_with": {}
  },
  "greet": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 5,
    "confused_with": {}
  },
  "affirm": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 7,
    "confused_with": {}
  },
  "inform": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 63,
    "confused_with": {}
  },
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 91
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 91
  }
}
```