

Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering



Diploma Thesis

Designing AI-powered chatbots

Ing. Farrukh Alinazarov

© 2023 CULS

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Ing. Farrukh Alinazarov

Systems Engineering and Informatics

Informatics

Thesis title

Designing AI powered chatbots

Objectives of thesis

The goal of this thesis is to design and implement a chatbot as a virtual assistant for customer support deployed in Facebook Messenger. The chatbot will be build using the elements of artificial intelligence as well as machine and deep learning methods to help the machine to understand and reply meaningfully to the customer's queries.

Methodology

The theoretical part will be based on the study of professional literature and relevant sources on building chatbots, virtual assistants, and as well as the application and outcomes of using chatbots in our lives. The types of chatbots, tools, technologies and the ability of the bots to understand and reply to user inputs will be also described as well as the process of development of chatbots. The second and practical part of the thesis will be based on open-source tools and frameworks: Rasa library, SpaCy NumPy, SciKit learn, Tensorflow and others. The code will be written in python and the above-mentioned libraries will be used. The bot will also use natural language processing and machine learning models.

The proposed extent of the thesis

60-80 pages

Keywords

Chatbot; Deep learning; Python; NLP; Artificial Intelligence; NLU

Recommended information sources

Gentsch, Peter (2018) AI in Marketing, Sales and Service – How Marketers without a Data Science Degree can use AI, Big Data and Bots. Palgrave Macmillan Cham. ISBN 978-3-319-89957-2

McTear, Michael (2020) Conversational AI – Dialogue Systems, Conversational agents and chatbots. Springer Cham. ISBN 978-3-031-02176-3

Melvin Paul Jacob, Debanjan Dutt, Dheeraj Sankar N, Om Shinde, Sayantan Bhattacharya (2021) Chatbots for customer support. Publisher: www.jetir.org. ISSN 2349-5162

Sumit, Raj (2018) Building Chatbots with Python. Using Natural Language Processing and Machine Learning. Apress Berkeley, CA. ISBN 978-1-4842-4096-0

Expected date of thesis defence

2022/23 SS – FEM

The Diploma Thesis Supervisor

doc. Ing. Vojtěch Merunka, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 7. 3. 2023

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 13. 3. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 22. 03. 2023

Declaration

I declare that I have worked on my diploma thesis titled “**Designing AI-powered chatbots**” by myself and I have used only the sources mentioned at the end of the thesis.

In Prague on 22.3.2023

Farrukh Alinazarov

Acknowledgment

I would like to express my appreciation and gratitude to my supervisor doc. Ing.Vojtěch Merunka, Ph.D, for his guidance and encouragement throughout my work on this thesis.

Additionally, I would like to express my appreciation to my parents, and my brother for their consistent support and patience during the course and thesis work.

Next, I would like to thank the general manager of Miss Sophie's Charles Bridge Yasmeira Penalosa who helped with providing training data and supported running this project as part of the internship.

Special thanks to God for guidance, for pushing me to work on this thesis hard, and for every action toward my accomplishments.

Designing AI-powered chatbots

Summary

The following diploma thesis deals with building virtual assistants for customer support in the form of chatbots from scratch. The thesis focuses on designing chatbots using the element of artificial intelligence and specifically using NLP – natural language processing. The main feature of the bot is the interaction of the computer with the user. The next function is the ability of a machine to learn from the user questions by interacting with and getting feedback. The thesis explains to the reader the definition of chatbots, their need in our life, the types of chatbots, and the benefits of their application in daily life and business.

The main goal of the thesis is the ability of the bot to recognize the user inputs by using artificial intelligence elements – NLP and NLU technologies as well as meaningfully replying to their queries. The next goal is the deployment of a chatbot in the Facebook Messenger platform. The code will be written in python and open-source frameworks and libraries such as Rasa library, SpaCy NumPy, SciKit learn Tensorflow and others.

Keywords:

Chatbot; Deep learning; Python; NLP; Artificial Intelligence; NLU

Návrh chatbotů založených na AI

Souhrn

Následující diplomová práce se zabývá budováním virtuálních asistentů pro zákaznickou podporu ve formě chatbotů od základu. Práce se zaměřuje na návrh chatbotů s využitím prvku umělé inteligence a konkrétně s využitím NLP – zpracování přirozeného jazyka. Hlavním rysem robota je interakce počítače s uživatelem. Další funkcí je schopnost stroje učit se z uživatelských otázek interakcí a získáváním zpětné vazby. Práce vysvětluje čtenáři definici chatbotů, jejich potřebu v našem životě, typy chatbotů a výhody jejich uplatnění v každodenním životě a podnikání.

Hlavním cílem práce je schopnost bota rozpoznat uživatelské vstupy pomocí prvků umělé inteligence – technologií NLP a NLU a také smysluplně odpovídat na jejich dotazy. Dalším cílem je nasazení chatbota na platformě Facebook Messenger. Kód bude napsán v pythonu a open-source frameworkech a knihovnách, jako je knihovna rasa, SpaCy NumPy, SciKit learn Tensorflow a další.

Klíčová slova:

Chatbot; Hluboké učení; Krajta; NLP; Umělá inteligence; NLU

Table of Contents

1. Introduction.....	15
2. Objectives and methodology	17
2.1.Objectives	17
2.2.Methodology.....	17
3. Literature review	18
3.1.Chatbots.....	18
3.2.The field of usage	20
3.2.1. Hospitality	20
3.2.2. Ecommerce.....	21
3.2.3. Healthcare	22
3.2.4. Financial services	22
3.3.Types of chatbots.....	23
3.3.1. Rule-based chatbots	24
3.3.2. AI-based chatbots.....	24
3.3.2.1. Retrieval -based chatbots	24
3.3.2.2.Generative-based chatbots	24
3.4.Dialogue system architecture	25
3.4.1. Speech recognition	25
3.4.2. Natural language understanding	26
3.4.3. Dialogue manager	27
3.4.4. Knowledge source	28
3.4.5. Natural language generation	28
3.4.6. Text-to-Speech Synthesis	29
4. Artificial Intelligence	30
4.1.Benefits and limitations	30
4.2.Neural networks	32

4.3. Deep neural networks (deep learning).....	36
4.4. Deep Learning models in practice	37
4.4.1. Automatic speech recognition	37
4.4.2. Voice-based assistant	37
4.4.3. Automatic machine translation	38
4.4.4. Automatic text generation	38
4.5. Natural language processing (NLP)	39
4.5.1. POS Tagging	40
4.5.2. Stemming	40
4.5.3. Lemmatization	40
4.5.4. Entity detection	41
4.5.5. Stopwords	41
4.5.6. Dependency parsing	41
4.5.7. Noun chunks	42
4.5.8. Similarity between words	42
4.5.9. Tokenization	43
5. Tools to build a chatbot	44
5.1. Python	45
5.2. Open-source libraries to develop a chatbot	45
5.2.1. SpaCy	46
5.2.2. Rasa	47
5.2.3. TensorFlow	47
5.2.4. NLTK	48
5.2.5. NumPy	48
5.2.6. SciKit learn	48
5.3. PyCharm	48
6. Practical part	50

6.1. Building customer-support chatbot	50
6.2. Setting up the environment	50
6.3. Components and terminologies used in chatbot	53
6.3.1. Rasa Open Source Flowchart Diagram	53
6.3.2. Intents	54
6.3.3. Entities	55
6.3.4. Utterances	55
6.3.5. Training the chatbot	55
6.4. Creating a New Chatbot in Rasa	56
6.4.1. Creating Domain file	58
6.4.2. Creating training data and rules	61
6.4.3. Creating pipelines and policies	63
6.4.4. Training and testing the model	67
6.4.5. Integration to Facebook Messenger	70
7. Results and Discussion	74
8. Conclusion	79
9. References	80

List of Figures

Figure 1:	Prediction of use cases for chatbots	20
Figure 2:	Application of chatbots in finance service	23
Figure 3:	Typical dialogue system architecture.....	25
Figure 4:	The branches and sections of the computational methods	33
Figure 5:	Artificial neural network architecture	34
Figure 6:	Weight of each element and input and output of the ANN system..	35
Figure 7:	The architecture of Deep learning technologies	37
Figure 8:	Simple dependency relationship between two word.....	42
Figure 9:	Python version verification	51
Figure 10:	Creating an environment in Anaconda	51
Figure 11:	Environment activation in Anaconda	52
Figure 12:	Uninstalling the old version of the pip package	52
Figure 13:	Ensuring pip package	52
Figure 14:	Reinstalling pip package	52
Figure 15:	Installing rasa open source	52
Figure 16:	Verification of rasa installation version	53
Figure 17:	Rasa command line interface	53
Figure 18:	Rasa Open-Source Architecture	54
Figure 19:	Creating a new default bot in Rasa	56
Figure 20:	Project creation completion in Rasa	56
Figure 21:	Training the basic bot in Rasa	56
Figure 22:	Testing the default bot after training	57
Figure 23:	Rasa bot project files	57
Figure 24:	Creating domain file – intents	59
Figure 25:	Creating domain file – responses	59
Figure 26:	Creating domain file – response examples	60
Figure 27:	Intents examples	61
Figure 28:	Stories examples	62
Figure 29:	Rules examples	62
Figure 30:	Rasa bot pipelines	63
Figure 31:	Rasa bot pipeline’s architecture	64
Figure 32:	Tokenizers architecture	64
Figure 33:	Featurizers architecture	64

Figure 34:	Intent Classifiers architecture	65
Figure 35:	Entity Extractors architecture	65
Figure 36:	Rasa bot policies	66
Figure 37:	Rasa bot action endpoint	67
Figure 38:	Rasa bot model explanation – intent examples	68
Figure 39:	Rasa bot model explanation – story’s examples	68
Figure 40:	Rasa bot model explanation – response examples	69
Figure 41:	Rasa bot model explanation – response text value.....	69
Figure 42:	Rasa bot training process	70
Figure 43:	Rasa bot testing process	70
Figure 44:	Access tokens	71
Figure 45:	App secret	71
Figure 46:	Credentials file in Rasa – messenger adjustment	71
Figure 47:	Ngrok interface	72
Figure 48:	Rasa running server process	72
Figure 49:	Webhooks adjustment	73
Figure 50:	Adding subscriptions	73
Figure 51:	Bot testing in Messenger part 1	74
Figure 52:	Bot testing in Messenger part 2	75
Figure 53:	Bot testing in Messenger part 3	76

List of Tables

Table 1:	Entity detection	41
Table 2:	Rasa bot file structure	58
Table 3:	Rasa bot commands	58

List of abbreviations

NLP	Natural Language Processing
NLU	Natural Language Understanding
AI	Artificial Intelligence
API	Application Programming Interface
ML	Machine Learning
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
DM	Dialogue Manager
SR	Speech Recognition
ASR	Automatic Speech Recognition
KM	Knowledge Manager
NLG	Natural Language Generation
TTS	Text-To-Speech
DNN	Deep Neural Network
IVA	Intelligent Virtual Assistants
IPA	Intelligent Personal Assistants
AR	Augmented Reality
VR	Virtual Reality
CRM	Customer Relationship Management
NN	Neural Network
DP	Deep Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
VPR	Voice-Print Recognition
SEO	Search Engine Optimization
POS	Part-Of-Speech Synthesis
NLTK	Natural Language Tool Kit
IDE	Integrated Development Environment
SDK	Software Development Kit
YML/YAML	Human-Readable Data-Serialization Language

1. Introduction

In this thesis, I'm going to design a chatbot that uses elements of artificial intelligence. The chatbot might serve as a virtual assistant for customer service. It forms a kind of foundation that can be turned into a specialized chatbot for a specific area or function that the company would require using machine learning or other techniques. The chatbot is going to use machine learning algorithms to learn from the training materials and subsequently use the learned data as a response to user inputs.

Recent developments in artificial intelligence have shown that companies should integrate chatbots into their business. Nowadays we can face chatbots almost in every sphere, retail, finance, hotels, healthcare, etc. as per the Forbes AI Stats News 86 % of clients would prefer to get answers to their questions from a chatbot rather than from a website by filling out the forms.

Virtual assistants respectively chatbots are the technologies that rapidly change the customer engagement atmosphere like never before. Imagine being forced to wait a long time in a line that might never end before speaking with a customer support agent. Analysis of human behavioral patterns suggests that technological advancements are to blame. Companies are seeking out quick solutions to accelerate operations as our patience grows thin for instance, many customers assume their questions to be answered immediately. Assuming the business gets many of these inquiries per minute, it would be completely impossible to answer each one promptly—unless, of course, there was a platform that could offer quick results, like chatbots (Melvin, Debanjan, Dheeraj, Om, and Sayantan, 2021).

One of the differences and huge benefits of chatbot solutions and human agents is its accessibility of providing advice and assistance for the services and products at any time 24/7. This advantage of chatbots leads to better customer satisfaction. And the bots indeed offer better customer service solutions than human agents many customers even didn't realize that it wasn't human agents when they were asked, and they would prefer such systems to be in their service. Chatbot systems are rapidly developing and it's in terms of the technical side as well as the popularity among users and customers. Despite their usage in daily life, chatbots are quite common as they are reachable, improve user service, can handle a huge number of customers at the same time, and are cost-effective compared to human agents as they can run non-stop while people can't work 24 hours and humans have emotions when dealing with sensitive customers. Talking about the technical part of the chatbots we know that today's chatbots can use artificial intelligence tools, so they help to better understand though making them more accurate which

leads to cost and time effective as well as customer satisfaction and individual approach to every single customer (Melvin, Debanjan, Dheeraj, Om and Sayantan, 2021).

2. Objectives and methodology

2.1. Objectives

The main goal of the following thesis is to design and implement a chatbot as a virtual assistant for customer service. The bot will use technologies such as NLP and NLU to help the machine to understand and reply meaningfully to the customer's queries. The thesis will also emphasize the deployment of the chatbot in the Facebook Messenger platform.

The secondary goal of the thesis is an introduction to artificial intelligence, machine learning, natural language processing, and natural language understanding by the machine, and as well as the process of development of chatbots will be covered.

The final goal is going to be the presentation of the developed chatbot and a discussion about its future challenges and improvements as a virtual assistant.

2.2. Methodology

The theoretical part is based on the study of professional literature and relevant sources on building chatbots, virtual assistants, and as well as the application and outcomes of using chatbots in our lives. The types of chatbots, tools, technologies, and the ability of the bots to understand and reply to user inputs are also described.

Finally, the practical part of the thesis will be based on open-source tools and frameworks: Rasa library, SpaCy NumPy, SciKit learn, Tensorflow, and others. The code will be written in python and the above-mentioned libraries will be used. The bot will also use natural language processing and machine learning models.

3. Literature review

3.1.Chatbots

Chatbot is a dialogue system of a computer program that's able to interact with humans. Different ways of interaction with chatbots are available on the market:

- Text-based
- Voice-based
- Hybrid

Another way of interaction with chatbots is by clicking on the prebuilt answer/question options button on the dialogue system. These dialogue systems receive a request so-called "input" from the users then the computer program interprets and processes what the user wants to do the bot then, replies to the user with the output to the same dialogue system. We distinguish two types of chatbot systems based on the goal:

- Task-oriented
- Non-task oriented

The first type of chatbot is widely used and we use it on daily basis in our lives those are Amazon Alexa, Siri on iPhones, and many others that are purposefully built to perform the specific tasks of the users (McTear, 2021). Below bringing an example from a piece of task-oriented dialogue system:

Example 1.

USER: Alexa, I would like to travel

ALEXA: Where are you planning to go?

USER: I'm going to Barcelona

ALEXA: What city are you leaving from?

USER: Prague

ALEXA: What date are you flying out?

USER: Next Friday

Alexa: That will be amazing. You go from Prague to Barcelona on the 25. May 2022.

The above-mentioned example shows how task-oriented dialogue systems work. In task-oriented systems the computer the bot needs a task in form of a question sent to the bot, then the bot asks additional questions which follow the intent of the user to complete the task and accomplish the goal.

Non-task-oriented chatbots /dialogue systems are designed to engage the system and humans in interaction in general talks. Below the example of non-task oriented chatbot is

illustrated (McTear, 2021).

Example 2.

USER: How are you?

MITSUKU: I'm always fine. How are you?

USER: I'm ok.

MITSUKU: Glad to hear it. What's new?

USER: It's cloudy outside today.

MITSUKU: Ah. Where would you rather be?

USER: Somewhere warm and sunny.

Example 2 illustrates non-task-oriented dialogue systems and how these systems engaged in daily lives.

“Bot, find me the best price on that CD, get flowers for my mom, keep me posted on the latest developments in Mozambique” (Leonard, 1996).

The topic isn't new. Back in 1966, the first chatbot called ELIZA was built by Joseph Weizenbaum and demonstrated the capabilities of machine and computer programs to communicate with humans using natural language. It was designed in such a way that the machine played the role of psychotherapist and was based on a structured dictionary which had a task to search for keywords in inputs. That bot was the first generation of a dialogue system between machines and humans. Regardless the bot had a psychotherapist model and it was an uncertain success, but it predefined the direction of dialogue systems for future development (Gentsch, 2018).

Particularly last ten years have seen a new quality emerge in bots improvement because of the rapid development of artificial intelligence, platforms, messaging devices, and speech recognition which lead to the realization of Andrew Leonard's wish in 1996 (Gentsch, 2018).

The remarkable advantage over the traditional ways of doing things online is that chatbots can help us to do multiple tasks and can board multiple users at once. They can assist you to book a hotel room, or booking a table in a restaurant, increasing sales and even automating payments. With their ability to serve multiple purposes, chatbots help people save a lot of time and money (Raj, 2019).

The need for chatbots is stably increasing. However, there hasn't been a lot of research that has experimentally attempted to ascertain the reasons why people use chatbots. Recently it has been reported that behind the motivation to use chatbots in daily live was revealed “productivity” as the main factor due to cost-effective and time-saving benefits (Raj, 2018).

3.2. The field of usage

Let's look at the demand for chatbots, especially from a business standpoint. Sales managers and product managers who run the business directly should not miss this huge opportunity of using chatbots from a business perspective (Raj, 2018).

More sectors are incorporating chatbots into their business operations to provide ongoing consumer engagement as a result of rising chatbot trends (Patel, 2022)

As per Gartner, "Artificial Intelligence (AI) will be a mainstream customer experience investment in the next couple of years". 47% of organizations will use chatbots for customer care and 40% will deploy virtual assistants (Patel, 2022).

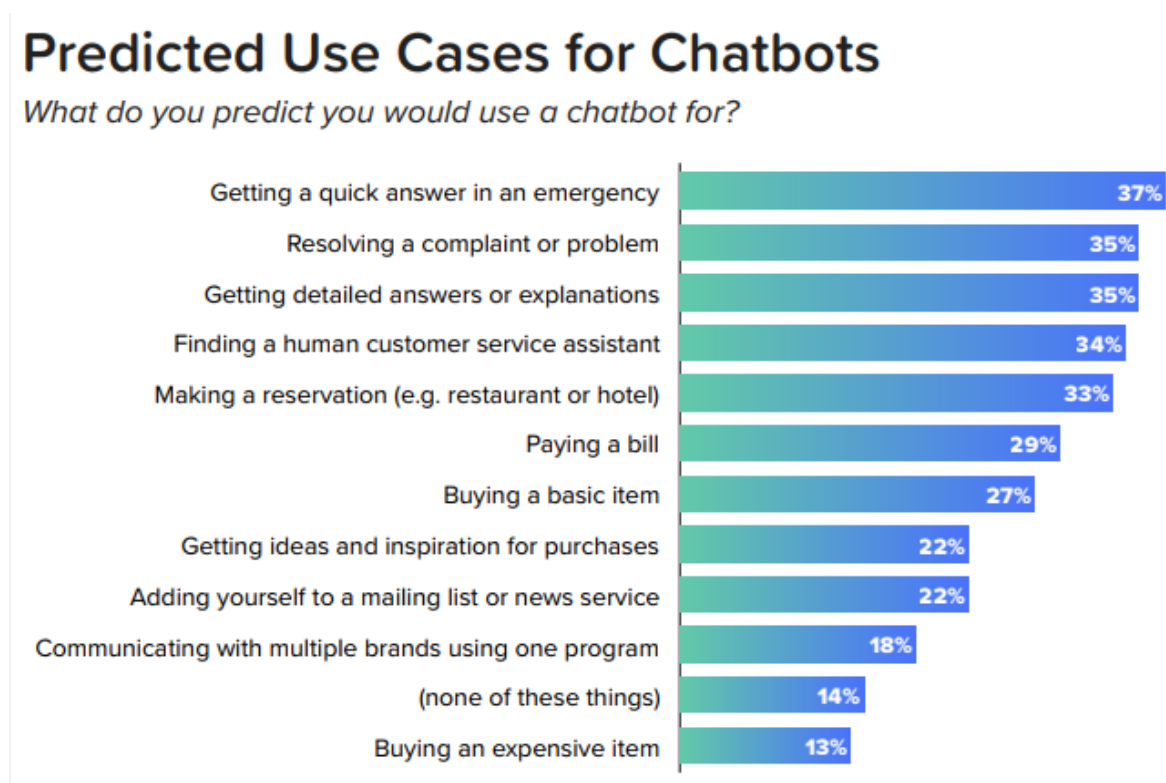


Figure 1: Prediction of use cases for chatbots. Source:(Digital Marketing Community)

As we can see from the figure the application of chatbots not only in customer engagement but in many different areas and performing multiple tasks will take a deep look at them in the upcoming paragraphs of this thesis.

3.2.1. Hospitality

Hoteliers are always looking for innovative solutions to improve the guest experience and reduce costs. Nowadays artificial intelligence offers a smart and powerful tool to fulfill goals of the hoteliers. Specifically, AI chatbots are the right choice offering guests personalized and efficient customer service for less money than the traditional methods (Shubham, 2022).

The following points are highlighting the key features from which hospitality can benefit:

- **Operation 24/7.** One of the first steps on the way to providing better customer service is offering non-stop services and AI chatbots are a perfect choice here assisting guests timely. As these tools don't have emotions like we humans and can help at any time especially in this fast-paced world for example during COVID-19 it helps perfectly non-contact with front staff or even if the front desk is closed.
- **Reduced operational costs.** The operational cost of AI chatbots is quite cheap. These systems initially can be developed using NLP with the ability to learn from each time when interacting with guests so because of that there is no need to make huge changes. In addition, they can be integrated into the hotel systems using API which makes it more effective and simple.
- **No more language barrier.** The hospitality sector is critical when comes to succeed in customer service. And knowledge of many foreign languages is key to making customers happy. If the hotel staff and guests don't speak the same language there will be misunderstandings which affect the customer standards. AI chatbot breaks down language barriers by providing 24/7 customer service in many languages.
- **Up-selling and cross-selling.** As we've talked about already about the personalized customer service that AI chatbot offers. Ai chatbot can be developed in such a way that it can offer based on user needs and budget for example premium rooms. When it comes to cross-selling the chatbot can offer additional products from hotel restaurants or bar or spa vouchers at a discounted price if the guest books a premium room or can offer discounts for future stays so these all lead to increasing revenue regardless of their support to do these tasks automatically (Shubham, 2022).

3.2.2. E-commerce

For many years, chatbots are integrated into various e-commerce operations, including order entry or modification, payment, etc. The bots with algorithms for product recommendations based on cart content or previous purchase from the same retailer are also very interesting uses for the e-commerce industry. Chatbots focus on building conversational recommendation systems to provide a smoother user experience (Cui, Huang, Wei a Tan, 2017).

The following points illustrate popular features used in E-commerce:

- Finding products
- Sending offers
- Sharing shipping information
- Customer service

As we all know how quickly e-shops are today growing in terms of innovative ways to maximize profit and dominate the market all the above-mentioned features we use in our daily life.

3.2.3. Healthcare

Healthcare is one of the critical industries for everyone. The adoption of chatbots in the following sector is beneficial for the customers as we know it all from the COVID-19 pandemic. Today ai based chatbots are getting implemented to perform the following tasks:

- ***Provide medical information.*** Large-scale healthcare data, including disease symptoms, diagnoses, indicators, and potential therapies, are used to train chatbot algorithms. Chatbots are regularly trained using public datasets, such as Wisconsin Breast Cancer Diagnosis and COVIDx for COVID-19 diagnosis (Dilmegani, 2022).
- ***Collect patient data.*** By asking straightforward questions like a patient's name, address, symptoms, current doctor, and insurance information, chatbots can gather information about the patient (Dilmegani, 2022).
- ***Handle insurance inquires.*** Patients and insurance plan participants can access insurance services and healthcare resources through chatbots. Additionally, using chatbots it is possible to automate the billing and processing of insurance claims (Dilmegani, 2022).
- ***Schedule appointments.*** Patients may plan, reschedule, and remove appointments using chatbots, which also discover doctors and dentists that are a good fit for them and their existing health issues. To deliver reminders and updates regarding medical appointments, chatbots can also be linked to users' mobile device calendars (Dilmegani, 2022).
- ***Provide mental health assistance.*** For patients with depression, PTSD, and anxiety, chatbots are trained to give cognitive behavioral therapy (CBT), and they may even teach autistic patients how to become more social and how to succeed in job interviews. Chatbots allow users to communicate with them via text, microphones, and cameras (Dilmegani, 2022).

3.2.4. Financial services

Artificial intelligence (AI) developments are gradually transforming how financial institutions function and engage with their clients. In reality, chatbots have replaced long lines and other inconveniences associated with visiting the office as the new standard for delivering financial services. Businesses are expected to save \$7.3 billion by using chatbots for financial services over the next two years (Singh, 2022).

As the figure below shows that businesses are using and planning to implement chatbots in their services which again highlights the high demand for chatbots.

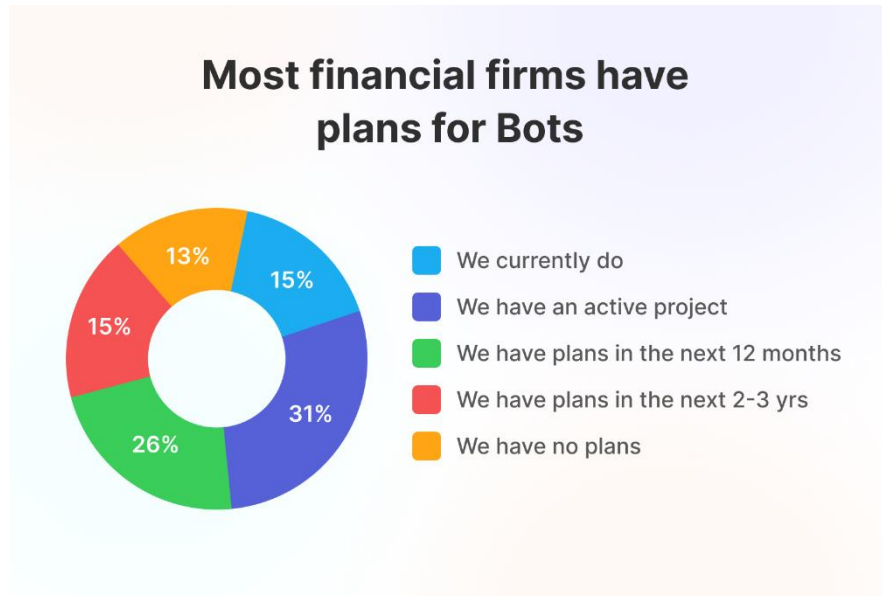


Figure 2: Application of chatbots in finance service. Source: (Revechat Company, 2022)

Chatbots are introducing a new method for financial services to increase operational effectiveness and task automation ensures faster support. The bots are offering to be a valuable asset in minimizing client annoyances and improving the interaction with financial systems. So to improve their operations, firms are now looking to employ conversational AI (Singh, 2022).

Today, there have been designed particular chatbots for financial services to automate many functions and operations including:

- Money management
- Financial advice
- Tax assistance
- Mortgage
- Fraud detection
- Loan approval
- Insurance claim

3.3.Types of chatbots

In this subchapter, we are going to talk about the different types of chatbots and their integration into the business and features. Mainly from the development perspective, we distinguish two types of chatbots

- Rule-based chatbots
- AI-based chatbots

3.3.1. Rule-based chatbots

Rule – based approach work in the way how the bot was trained before so it can reply accordingly. These rules specified can range from being very basic to extremely complicated. These bots are easy to build using a rule-based method, but they are ineffective at handling queries since their pattern does not match the rules they were trained on. Additionally, it takes a lot of time to develop rules for various instances, and it is difficult to write rules for every conceivable scenario. Simple requests are handled by the bots, but complicated queries are not. As a result, if the bot is based on some rule-based models, it will never pass the Turing test (Shridhar, 2017).

3.3.2. AI-based chatbots

AI-based chatbots use both Machine Learning (ML) and Natural Language Processing (NLP) to understand user input and intent before generating the response which makes them more efficient than rule-based bots. These bots are more trained, and they keep learning from user inputs. The more they interact with users the more accurate become their responses.

3.3.2.1.Retrieval-based chatbots

These types of the bot are designed on a set of questions along with their response. For every input by the user, the bot can find the most relevant answer from its dataset of all possible responses and then outputs. Despite that, the bot can not generate new responses by sequencing however, if it's trained well and a lot on responding to many questions with their possible answers based on the dataset with smart processing it can handle it fairly well. Simple rules for a query can be as sophisticated as complex rules utilizing a machine learning algorithm to determine the best response. Additionally, there are no mistakes with language or grammar because the solutions are known in advance and the syntax can never be incorrect (Shridhar, 2017).

3.3.2.2.Generative-based chatbots

The ability to generate answers rather than constantly responding with one from a list of answers makes generative models superior to rule-based models in this regard. As a result, they become more intelligent since they can generate answers by taking the question word for word. They are also more likely to make mistakes since they have to pay attention to spelling and grammar. These models need to be trained more precisely to improve their ability to handle these faults. Once trained, they dominate rule-based models because they can respond to intricate and illogical questions (Shridhar, 2017). If we take a look at the technical side of building generative-based chatbots they use more complex and smart technologies of deep learning such as sequence

models, end-to-end models, and neural network based technics – Recurrent neural networks (RNN) and long short term memories (LSTM) then they train dataset. This method is the hardest and thus it's smarter and more precise compared to the above-mentioned methods.

3.4. Dialogue system architecture

Even though dialogue systems have been available for a while, they have just recently entered the mainstream and become a regular part of life for billions of people. It is generally recognized that in 2011 when Apple released Siri, a personal assistant that facilitates spoken conversations with smartphone users, dialogue systems reached their maturity. Since then, dialogue systems have taken on a variety of shapes, including chatbots on platforms like Facebook Messenger, PDAs on smartphones, like Apple's Siri, Google Assistant, Microsoft's Cortana, and Samsung's Bixby, smart speakers like the Amazon Echo and Google Nest, and social robots like Pepper and Furhat (Al Moubayed, 2012).

In this chapter of the thesis, we are going to describe how these dialogue systems work and how they are designed.

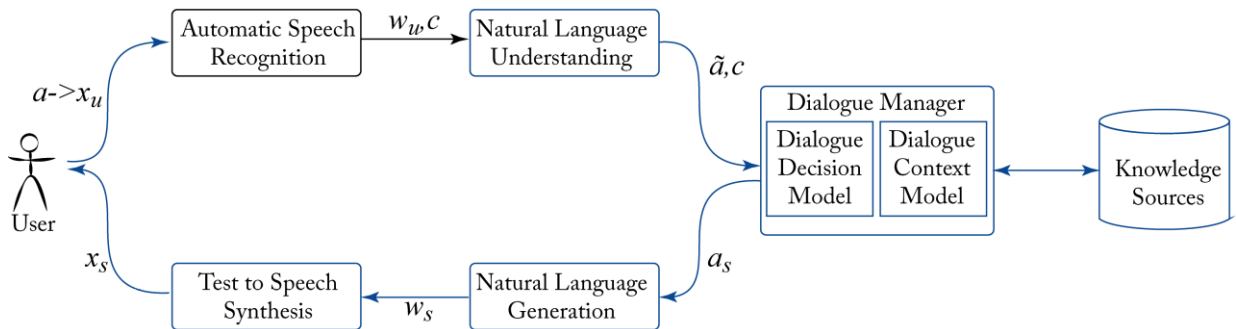


Figure 3: Typical dialogue system architecture. Source: (Conversational AI, 2021)

Figure 3 presents the typical dialogue systems architecture. It displays well in the right order how the interaction starts from user input and how the input is processed by these systems until it gets back to the user with the output respectively the reply to the user's question.

The upcoming chapters will cover each element and step of the dialogue systems separately and if needed an additional element according to the type of chatbot will be described as well.

3.4.1. Speech recognition

As figure 3 shows speech recognition or automatic speech recognition is the initial element of dialogue systems of voice and chatbots. In speech recognition, the acoustic signal x_u representing what the user said or asked is converted into a string of one or more words (the system's estimate of the user's utterance) that is produced. These conditional probabilities can be used to characterize

this process:

The equation for conditional probabilities to characterize the process of speech recognition:

$$\hat{W} = \arg \max_w P(W|X)$$

It claims that finding the greatest value of $P(W|X)$ a measure of the word string W 's probability given the acoustic input X —requires searching through all the word strings. When the Bayes rule is applied, the following results:

$$\arg \max_w P(X|W)(W)$$

- $P(W|X)$ – the probability of the acoustic signal x
- w is the *observation likelihood* that is captured by the *acoustic model*.

The *acoustic model* models the relationships between the acoustic signal and word strings. The other $P(W)$ explains the *prior probability* and it is captured by the *language model*. The *language model* simulates the likelihood of word sequences, frequently using finite state grammars but occasionally using N-gram language models. For speech recognition the acoustic and language models are trained on unlimited sample utterances thus, the developer has only to tweak the models if speech recognition is supported in that platform. An appropriate search technique can be used to compute the most probable word string W for a given acoustic signal X given these two probabilities (McTear, 2021).

The output of speech recognition, a probabilistic process, can be a set of word hypotheses that are frequently rated in an N-best list or it can be a lattice that can store many word sequence alternatives. In certain conversation systems, the NLU component receives the first-best hypothesis to further examine. The first-best hypothesis may not be accurate, hence it is beneficial to preserve numerous recognition hypotheses so that alternatives can be taken into account later on in the processing—for instance, as a consequence of re-scoring using data from other system components (McTear, 2021).

3.4.2. Natural language understanding (NLU)

The Natural Language Understanding (NLU) module's architecture aims to provide stability while allowing for the most spoken language flexibility possible (Pon-Barry, Weng, and Vargas, 2014).

This module covers various techniques for understanding natural language. NLU can have many different varieties. In syntax-driven semantic analysis, the input is first syntactically examined to discover how the words and phrases group together as elements, and the meaning is

then extracted using semantic rules affixed to the constituents. The advantage of using logic to express the meaning of the string is that it enables the use of common inference methods to give a greater degree of comprehension (McTear, 2021).

In general, the NLU module uses different tools and techniques to understand the human language/user input. It can use several steps for that purpose as sentiment detection, extraction of named entities (names, geographical names), coreference resolution (synonyms or different words referring to the same thing), etc. This module is primarily responsible for gathering all possible information that is implicitly (sentiment) or explicitly (named entities) present in the input text (Wolff, 2021).

3.4.3. Dialogue manager

The Dialogue Manager (DM) is the key element of a spoken dialogue system. It receives interpreted input from the SR and NLU components, interacts with outside knowledge sources, generates messages to be sent to the user, and controls the dialogue flow generally (McTear, 2021).

The DM maintains conversation context using the information-state-update technique, which is subsequently applied to parse incoming utterances (including fragments and revisions), resolve noun phrases, build salient answers, monitor difficulties, etc (Pon-Barry, Weng and Vargas, 2014). DM tends to prevail in goal-oriented dialogues because there is a definite goal to be achieved through the conversation (Schnelle-Walka, Radomski, Milde, Biemann a Mühlhäuser, 2016).

DM consists of two components:

- *Dialogue Context Model*
- *Dialogue Decision Model*

To assist in dialogue management, the *Dialogue Context Model* keeps track of data pertinent to the dialogue. This might contain details like what has been said thus far in the conversation and how grounded, or how much of it has been communicated between the system and the user.

Knowledge sources like the ones listed below might be included in a dialogue context model:

- *Dialogue History*
- *Task Record*
- *Domain Model*
- *Model of Conversational Competence*
- *User Preference Model*

In light of the user's input utterance and the details in the dialogue context model, the *Dialogue*

Decision Model chooses the subsequent system action. Decisions may involve requesting the user for further input, clarifying or grounding the user's earlier input, or displaying some information to the user. These judgments are pre-scripted in rule-based systems, with options depending on things like the confidence levels connected to the user's input. For instance, if the confidence levels are high enough, the system can move on to its next action and presume that it has correctly processed the input, however, if the levels are low, it may first try to confirm that it has done so or may even ask the user to repeat their statement. Such choices can be made in advance and hard-coded into the dialogue decision model (McTear, 2021).

Dialogue decisions are also could be made dynamically, for instance by considering the dialogue's present status and relying on data from a variety of domain and dialogue knowledge sources. The relative relevance of the information that has been elicited, what has to be accomplished in the job at hand, the user's demands and preferences, and the dialogue's prior history may all be taken into consideration by DM in addition to confidence ratings from the SR and NLU components (McTear, 2021).

3.4.4. Knowledge sources

Knowledge source manages updates to knowledge base sources (such as domain knowledge and device information). Domain-dependent ontologies are used to structure domain knowledge. The ontologies are maintained offline using a domain-agnostic ontology tool. In a typical interaction, the Dialogue Manager transforms the user's query into a semantic frame (i.e., a list of semantic constraints) and delivers this to the Knowledge Manager (KM) via the content optimizer (Pon-Barry, Weng, and Varges, 2014).

3.4.5. Natural Language Generation

Based on the conversation manager's output, the Natural Language Generation (NLG) component creates the text that will be displayed to the user. Figure 3 illustrates how NLG transforms the output a_s from DM into words w_s (McTear, 2021).

An approach to NLG that is more sophisticated sees generation as a pipeline model with three stages: Document Planning Microplanning, and Realization (Reiter and Dale, 2000). Document Planning requires thinking through issues like how to choose and rank options from the content that needs to be expressed, how to use discourse relations like comparison and contrast to present the information meaningfully, and how to tailor the information to the user's perceived needs and preferences. Creating sentences to communicate the information is the focus of Microplanning, which includes activities like handling referring expressions and determining

how to refer to things in a certain context. Last but not least, Realization is concerned with how to communicate the content in terms of classifying various propositions into clauses and sentences, coming up with relevant referencing phrases, and utilizing appropriate discourse signals. The most current NLG systems consider NLG as a decoding issue where the text is created word-by-word from a hidden state. These systems have been built and tested in the end-to-end neural conversation method (Dušek et al., 2020).

3.4.6. Text-to-Speech Synthesis (TTS)

After determining the output of NLG w_s , the next step is to convert it into a spoken utterance x_s . Pre-recorded prompts are frequently used in commercial systems where the output can be expected during the planning phase. Voice talents, or trained voice actors, who record the system's outputs, are also used. Although this produces an output of a high caliber, it is a costly alternative and is only appropriate for outputs that remain the same throughout time. The use of Text-to-Speech Synthesis (TTS), in which the text that will be spoken is synthesized, is an option (McTear, 2021).

Traditional TTS has two stages: Waveform Synthesis and Text Analysis. In text analysis, the text that will be spoken is converted into a representation made up of prosodic information and phonemes. Before this, the text is normalized, which entails turning acronyms, abbreviations, and other unusual terms into regular language. The internal representation Waveform Synthesis is transformed into a waveform that may be produced as spoken text that's done in the second stage. Concatenative synthesis, in which segments of recorded speech are concatenated, was up until recently the most widely used technique for waveform synthesis (McTear, 2021).

Deep Neural Networks (DNNs) have been used recently to construct end-to-end TTS systems, with encouraging results. For instance, in 2016 the DeepMind WaveNet deep neural network surpassed the top TTS systems for American English and Mandarin Chinese (Oord et al., 2016). However, there are still issues that need to be solved, such as how to accurately portray emotion and how to employ prosody elements like stress appropriately for contrastive emphasis (McTear, 2021).

4. Artificial Intelligence

Artificial intelligence (AI) has made a significant advancement in business possibilities practices. On its journey to the all-encompassing algorithmic workplace, AI is also progressively tackling administrative, decisive, and planning procedures in marketing, sales, and management (Gentsch, 2018).

Conversational AI has been defined as “the study of methods for developing software agents that can interact naturally with people in conversation” (Khatri et al., 2018). A new generation of conversational interfaces and dialogue systems have been made possible by advances in AI, notably deep learning, as well as the accessibility of enormous amounts of data and computer power (McTear, 2021).

In this chapter, we are going to provide an introduction to dialogue systems from the artificial intelligence perspective, about the benefits and limitations of the application of AI in chatbots as well as the construction of the dialogue system using deep learning methods.

4.1. Benefits and limitations

Intelligent Virtual Assistants (IVA) are AI-based virtual agents that can produce personalized responses by drawing on contexts like customer metadata, previous conversations, knowledge bases, geolocation, and other modular databases and plug-ins (Krasnokutsky, 2022). It is AI elements and algorithms that make those personal agents respectively chatbots so-called “intelligent”. Marketsandmarkets.com predicts that the Intelligent Virtual Assistant market, which grew quickly in the 2020s, would reach USD 10.5 billion by 2026.

Although it incorporates machine learning, AR/VR, data science, and next-generation analytics, AI assistant technology is similar to traditional chatbots in many aspects. While traditional chatbots are capable of answering questions using Markov chains and other similar techniques, the dynamic insights produced by intelligent virtual assistants much outweigh their static responses. The healthcare, telecommunications, travel and hospitality, retail, and BFSI industries are among the end-users of AI assistant technology. Smart speakers, cellphones, cars, trucks, home computers, home automation devices, and many other consumer goods use IVAs or IPAs. (Krasnokutsky, 2022).

The primary benefit of employing artificial intelligence to develop such solutions is that it can quickly and effectively evaluate enormous amounts of data, uncover patterns, and generate insightful recommendations. AI assistants that use voice and speech recognition make it much simpler to complete numerous daily tasks like adding events to your calendar, creating a reminder, or keeping track of your monthly costs. By 2024, more than 8 billion digital voice

assistants will be in use worldwide, approximately equal to the world's population (Krasnokutsky, 2022).

The following are some of the main advantages of creating AI-powered virtual assistants for businesses:

- **Improved customer support** – while reducing the volume of calls and service requests to human agents. You can automate the customer interaction process in business with the help of AI assistants. Your staff will be able to concentrate on more difficult work as a result of not wasting time on requests that can be handled automatically.
- **The ease of key data collection.** Without the need for a customer service staff to take meticulous notes, AI-based assistants may rapidly store away and categorize a customer's inquiries and the related metadata for analysis.
- **Personalized user experience.** AI-powered chatbots offer a high degree of personalization to the customer by adapting to each user's needs. For instance, IPAs can remember the user's preferences in addition to their name. Users are more engaged as a result, and customers are more satisfied and loyal as a result (Krasnokutsky, 2022).

Limitations and disadvantages of using AI-based virtual assistants. Behind all advantages of AI-based technologies that we've talked about above, it has also some weaknesses too, just like all technologies. The following points are describing the vulnerability of AI-based bots:

- **Data Security.** You should keep the audience data you collect secure. The data must be transferred securely from the chatbot to your CRM. Only pertinent data from your audience should be acquired, and it must be maintained securely (Barker, 2022).
- **Lack of Emotional Understanding.** Because chatbots are composed of codes, it is challenging for them to understand the user's emotions. They might not be able to tell whether the person they are conversing with is joyful, anxious, or unhappy as a result. This could make the chatbot seem emotionally indifferent, which could be bad for your brand's reputation (Barker, 2022).

Some Chatbots' Unavoidable Limitations. Because chatbots are closely related to organizations, it is essential to recognize their vulnerabilities. There are numerous restrictions, and both users and company owners have voiced their displeasure with them.

1. **Don't understand the human context.** It is one of the biggest weaknesses of chatbots. These chatbots have been programmed so that they can only learn new information. They

are unable to comprehend the context that people use, which is a significant gap that can even result in an irate consumer. The general context can be understood by AI-powered smart bots, however, 40 out of 100 situations are unrelated to the general context (Vishal, 2019).

2. **Don't do customer retention.** Every business must prioritize customer retention. It is more significant than acquiring new clients. A chatbot is much less effective at keeping consumers because it just attempts to the extent that it has been designed to (Vishal, 2019).
3. **Can't make clear decisions.** The inability of chatbots to make decisions is another drawback. They lack the necessary expertise to distinguish between excellent and bad (Vishal, 2019).

4.2. Neural networks

New systems and computational techniques, such as neural networks, are being used for machine learning, knowledge demonstration, and ultimately the application of newly acquired information to improve the output responses of complex systems (Chen et al. 2019). A data processing model called a neural network (NN) is based on how biological nerve systems, like the brain, process data. On a much smaller scale, they are concentrated on the neuronal organization of the mammalian cerebral cortex. Artificial neural networks, in the opinion of many specialists in the field, offer the best and possibly the only chance of creating a machine with intelligence (Dastres and Soori, 2021).

Figure 4 depicts the branches and sections of the computational techniques and Artificial Neural Networks.

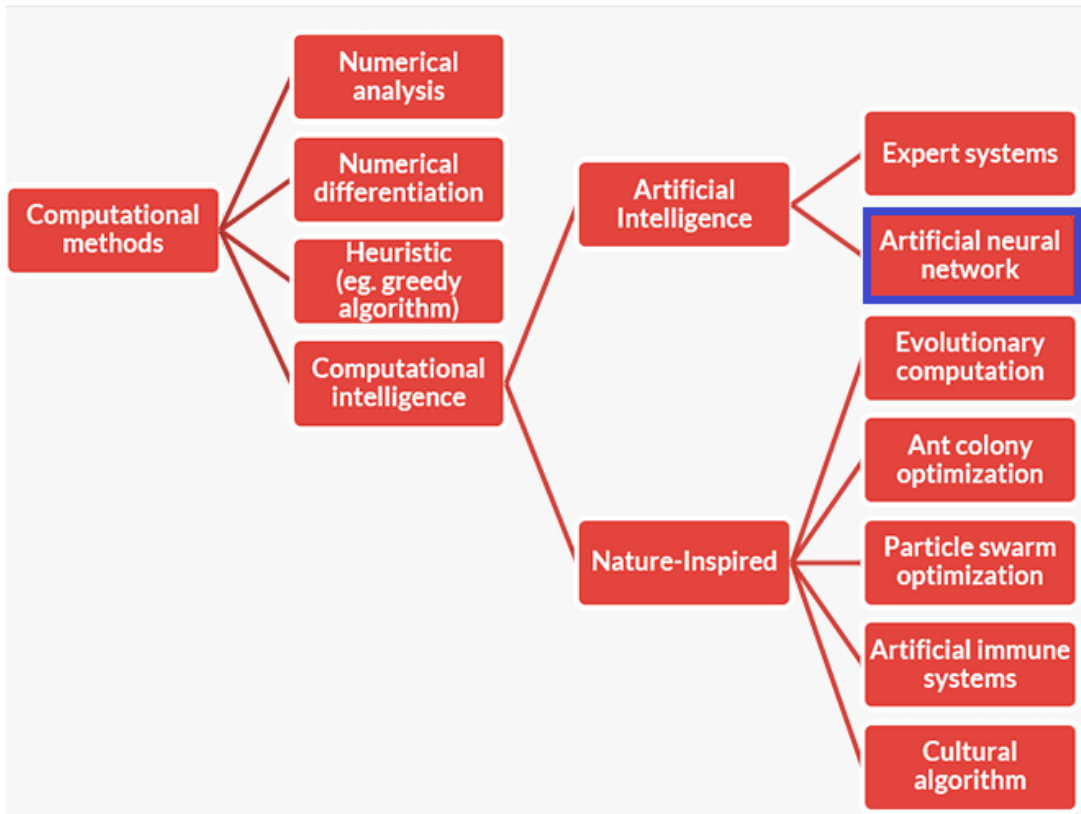


Figure 4: The branches and sections of the computational methods. Source: (Soori, 2021)

Artificial neural networks are built with neuron nodes connected in a web-like pattern, just like the human brain. The brain is made up of neuron cells, which number in the billions. Each neuron is made up of a cell body that transports information to and from the brain and processes it (Van Gerven and Bohte 2017). Such networks' central concept is (to some extent) modeled after how the biological nervous system functions, which involves processing data and information to facilitate learning and knowledge creation. The creation of new structures for the information processing system is the main component of this concept (Dastres and Soori, 2021). Figure 5 illustrates the construction of an artificial neural network.

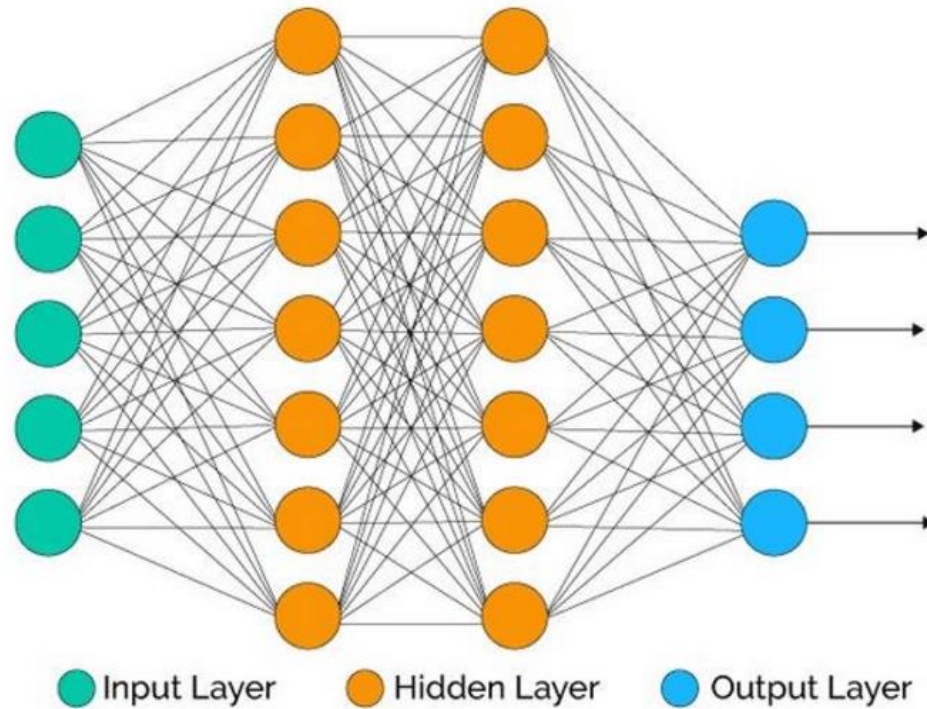


Figure 5: Artificial neural network architecture. Source: (Bre, Gimenez & Fachinotti, 2018)

The system is composed of numerous, intricately interconnected processing units called neurons that cooperate to address issues and communicate through synapses (electromagnetic connections). The neurons are arranged into layers and are highly linked. While the output layer produces the final result, the input layer receives the data. Usually, one or more secret layers are placed between the two. The exact flow of the data is impossible to predict or determine with this structure (Dastres and Soori, 2021).

In addition to having a threshold value and an activation function, each link includes a connection weight (Balakrishnan et al. 2019). Based on the input's weight sign, it is determined if each input has a positive or negative weight. The weight has an impact on a connection's signal strength (Liu et al. 2018). Neurons with a threshold beyond which a signal is only transmitted when the total signal is greater. The output is generated based on the signal from the activation value, which is the weighted sum of the summing unit. Figure 6 depicts the relationship between the weight of each element and the input and output of the ANN system (Dastres and Soori, 2021). The networks allow other cells to compensate for the loss of a damaged cell and aid in cell regeneration. These networks can learn. In essence, a system's capacity for learning is its most critical component. A learning system can respond to new problems and equations better since it is more adaptable and simpler to program. A neural network is configured to carry out specific tasks, such as finding patterns and categorizing information, throughout a learning process. Artificial neural networks, like people, learn by using examples. Injecting tactile nerve

cells, for instance, teaches the cells to avoid the heated body, and this method teaches the system to fix its error (Dastres and Soori, 2021).

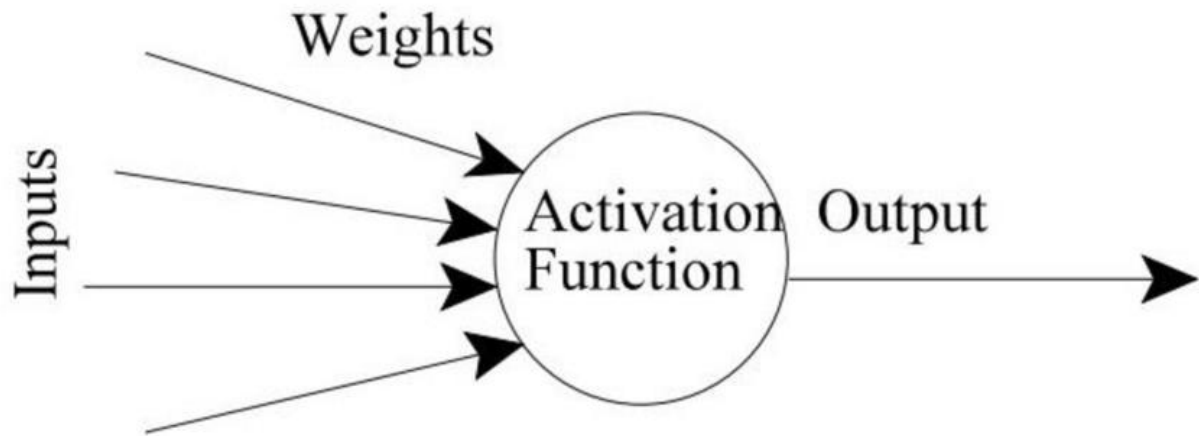


Figure 6: Weight of each element and input and output of the ANN system. source: (Soori, 2021)

How an artificial nervous system learns. The nervous system carries data in two separate ways. Examples of data are provided to the system using information units that cause layers of hidden units to be triggered at the level of extraction units at the moment of awareness (preparation) or default operation (after readiness). The feedforward system is a fundamental strategy. However, not every item operates flawlessly. Each unit receives contributions from units to one side, and the number of associations that sources of information pass through causes them to be replicated. If the sum is higher than a certain cut-off limit, the unit will "burn out" and activate the units it is connected to. Each unit holds each of the sources of information it obtains along these channels (to the right). Just like kids learn by being praised for their accomplishments, the nervous system needs some input to develop. The human mind uses information inputs to learn continuously. For instance, when we first learned how to bowl effectively. When we made an accurate throw and the ball moved to the intended location, we mentally noted the speed and trajectory of the ball to try to replicate that throw as closely as possible the following time (Chng, 2020).

Neural network usage. Artificial neural networks are being utilized more and more in the modeling and control of systems with unknown or extremely complicated internal structures. The neural network itself will learn the control function, for instance, if it is employed to control the input of an engine (Dastres and Soori, 2021). These systems learn adaptively, which means that as new inputs are introduced, the weight of the synapses adjusts to ensure that the system responds appropriately (Wu and Feng 2018).

Neural networks are implemented in many different applications today, including pattern recognition problems, which include problems with line recognition, speech recognition, picture

processing, and similar problems, as well as categorization problems, such as text or image classification (Li, Zhang, and Liu 2017). Neural networks are also used in risk analysis systems, drone control, welding quality analysis, computer quality analysis, ER testing, oil and gas exploration, truck brake detection systems, loan risk assessment, spectrum detection, and drug detection. Processes for industrial control, error control, speech recognition, hepatitis detection, remote information retrieval, mine detection under the sea, 3D object detection, handwriting, and face detection, etc. A few uses for ANNs include the calculation of known functions, approximating unknown functions, pattern recognition, and signal processing (Li, Zhang, and Liu 2017).

4.3. Deep neural networks (Deep Learning)

To represent more complex characteristics and "read" more sophisticated models for the prediction and categorization of data based on hundreds or even millions of features, artificial neural network systems need to be more complicated. Deep learning is a branch of machine learning that focuses on developing successive "layers" of ever-more-meaningful representations as it learns representations from data (Grekousis 2019). It is focused on artificial neural networks (ANNs), which are algorithms founded on the composition and operation of the human brain. Computational models with several processing layers can learn different degrees of abstraction for data representations thanks to deep learning. More than three layers of neurons are present in these neural networks (including the input and output layers). These layered representations are taught using so-called "neural network" models, where the layers are physically stacked on top of one another (Schmidhuber 2015). This may be achieved by just increasing the total number of hidden layers and/or the number of neurons per hidden layer. Increasingly complicated models may be represented by adding layers and neurons but doing so consumes more resources. Figure 7 depicts the architecture of deep learning technology (Santos et al. 2021).

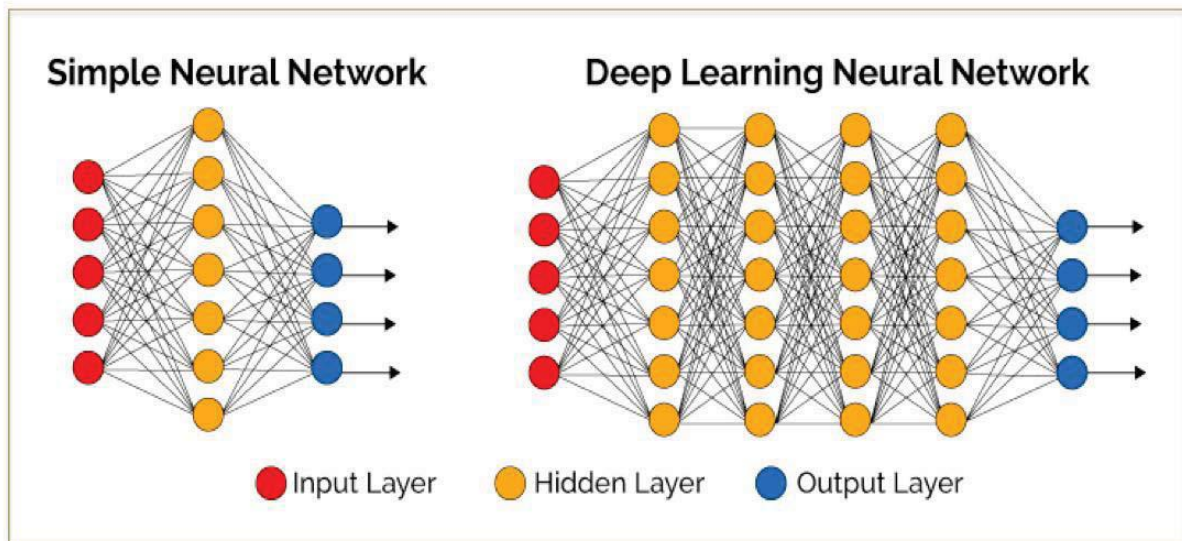


Figure 7: The architecture of Deep learning technologies. Source: (Santos et al. 2021)

Three important groups of deep learning algorithms are Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and Deep Neural Networks (DNN) (Shuohua, 2020). Each algorithm is used for different purposes.

4.4. Deep learning models in practice

The way we view innovation is changing so fast and all great innovations are thanks to the usage of deep learning models. Machine Learning and Deep Learning are not new but in the last approximately 10 years it's developing and altering the world in terms of automatization and autonomous technologies all other issues that can be solved by the computer. This chapter will introduce you to how the models of deep learning are used and the interesting thing is you will know that we are already using technologies based on those models.

4.4.1. Automatic speech recognition

The first and most convincing application of deep learning is the recognition of large-scale programmed conversations. LSTM RNNs can handle "deep learning" tasks including multi-second transitions with conversational possibilities isolated by a lot of discrete time steps, where one discrete time step is about equal to 10 milliseconds. An LTSM network can handle full data sequences and has feedback links (Purkaystha, Nahid, and Islam, 2017).

4.4.2. Voice-based assistant

Voice is most likely the most significant way to communicate with people. Human voice or speech is an information-rich signal that communicates a wide variety of information, including the speaker's emotions, speech inflection, and language content. The goal of voiceprint

recognition (VPR) is to detect, locate, and recognize a speaker based on their speech traits. Speaker recognition can be made simpler using a variety of techniques. These systems often have two phases: first, they extract the features, then they match or classify the features, with the classification element having two parts: pattern matching and decision (Hajer Y. Khdir et al 2021). The feature extraction module, which gathers the voice(s) of each speaker and uses them to build the matching speaker model, estimates a collection of speech signal features that reflect certain speaker-specific information. The voice dataset is a collection of voice models for all speakers (K. N. Van, T. P. Minh, T. N. S. B, and M. H. L. B, 2018). Well-known example is Siri from Apple, Alexa from Amazon, and Cortana from Microsoft which we use in our daily life.

4.4.3. Automatic machine translation

Machine translation is a technique that uses computerized systems to automatically translate sentences from one natural language into another, with no need for human intervention. Different methods are available to develop these kinds of systems, but a more durable method is needed to develop systems that are superior to those in use now (Singh, Shashi & Kumar, Ajai & Darbari, Hemant & Singh, Lenali & Rastogi, Anshika & Jain, Shikha, 2017). The system's objective is to produce more accurate and efficient translation systems, and well-trained network guides (Mohamed Amine Cheragui, 2012), (Li Deng And Dong Yu, 2013).

To build an improved machine translation system, many deep learning methods and libraries are needed. The system that will translate the sentence from the source language to the target language is trained using RNNs, LSTMs, etc. To maximize the accuracy of the translation system in comparison to other systems, it is a good idea to adapt the appropriate networks and deep learning algorithms (Singh, Shashi & Kumar, Ajai & Darbari, Hemant & Singh, Lenali & Rastogi, Anshika & Jain, Shikha, 2017). A good and very popular example of a machine translation model is the Google translate application that we all use.

4.4.4. Automatic text generation

Automatic text generation is the process by which computers produce texts in natural language. Text data is a common type of data, and its usage is rapidly increasing. In the area of text information extraction, it has become urgently necessary to research how to rapidly and effectively discover and use useful information from large text data. The traditional natural language-generating technology's reliance on templates has been overcome by advances in deep learning technology. To provide an end-to-end solution and minimize the level of human involvement, it can automatically learn the input-to-output mapping from the data. It allows the generation system to more broadly generalize and generate more free text under the

predetermined parameters by the requirements (Shuohua, 2020). Substantial recurrent neural networks (RNN) are used to learn the connections between things in a sequence of information chains and then create content (Daza, Calvo a Figueria-Nazuno, 2016).

Examples of using automatic text generation models are SEO, recruiting tools, and many more.

4.5.Natural language processing (NLP)

Natural language processing is the part of artificial intelligence that helps the computer to understand human spoken language. NLP integrates cognitive algorithms such as statistical, machine, and deep learning algorithms with computational linguistics, which is the rule-based modeling of spoken human language. The intelligent voice assistants and chatbots that you may use in daily life are made possible by the combination of the above-mentioned technologies (Mondal, 2021).

Humans employ a variety of grammatical mistakes, regional variations, and unique intonations in their speech every day. The machine can quickly comprehend, interpret, and react to a huge volume of text in real-time thanks to NLP technology. You've probably used NLP technology in your daily life via chatbots that provide app assistance, virtual assistants, speech-to-text note-creation apps, and voice-guided GPS apps (Mondal, 2021).

It takes humans years to overcome these difficulties and completely learn a new language. Programmers have added numerous features to the NLP technology to produce practical technology that you can use to comprehend human speech, process it, and provide a relevant answer to overcome these difficulties (Mondal, 2021).

To make work all the words said above the NLP has tom deal with the user's input respectively text and audio and process the in a way that can be analyzed and converted into data that the computer understands. For that purpose, we shine the following terms in the next chapters:

- POS Tagging
- Stemming
- Entity detection
- Stopwords
- Dependency parsing
- Noun chunks
- Similarity between words
- Tokenization

As all these techniques and methods are important to build a chatbot (Raj, 2018).

4.5.1. POS Tagging

Part-of-speech tagging helps the sentence to divide into separate text and identifies every single word or token where they belong according to the grammatical category such as noun, verb, adjective, and so on. POS tagging plays a key role if we are going to specify an entity in a sentence (Raj, 2018). Let's have a look at the example of POS tagging.

Example 1.

“How can I book a room in your hotel?”

Output:

(‘How’, ‘ADV’)

(‘can’, ‘VERB’)

(‘I’, ‘PRONOUN’)

(‘book’, ‘VERB’)

(‘room’, ‘NOUN’)

(‘in’, ‘PREP’)

(‘your’, ‘PRONOUN’)

(‘hotel’, ‘NOUN’)

That is how the POS tagging method works and assigns each word to the parts of speech.

4.5.2. Stemming

Stemming is the procedure of minimizing accented words to their word stem, the base form of that word. This algorithm tries to minimize for example the word “asking” to its root “ask”. As we can see it shows that may not be 100 % correct (Raj, 2018).

Stemming accomplishes the task in a clumsy, heuristic manner by cutting off the ends of words with the assumption that the remaining word is what we are looking for, but often involves deleting derivational affixes. (Raj, 2018).

4.5.3. Lemmatization

If we take a look at the definition of “Lemma” which is the base form of the verb for human spoken language. It seems the same method as above mentioned method “stemming” but “lemmatization” identifies the base form of the verb algorithmically and takes into account the meaning of the word. Let's take a look at an example of the word “learn” “learns” or “learned” all are written in different forms but give us the same meaning of the process of – learning.

With the use of a vocabulary and morphological study of words, lemmatization attempts to do the task more elegantly. It makes an effort to keep only inflectional endings and return a word's lemma, which is its dictionary form (Raj, 2018).

4.5.4. Entity detection

Entity detection or entity extraction or named-entity recognition does the task of extraction and classification of entities in the text into pre-defined categories. Let's have a look at the example to understand what the above-mentioned sentences mean:

“My renting apartment in Prague costs 15000 CZK”

Output:

(‘My’, ‘PERSON’)

(‘apartment’, ‘FAC’)

(‘Prague’, ‘GPE’)

(‘15000 CZK’, ‘MONEY’)

Table 1 Entity detection, source:(Raj, 2018)

Type	Description
PERSON	People, including fictional
FAC	Buildings, airports, highways, bridges, etc.
GPE	Countries, cities, states
MONEY	Monetary values, including unit

4.5.5. Stopwords

Stopwords are words that don't make a huge change in the meaning of the sentences but help us – humans to make them easy to understand but for the machine, they are making complications. The main task of the following method is to clean so-called “stopwords”. Stopwords are such as, *a, an, the, just, like, on, for, also*, and so on.

Stop words are a crucial component of text cleaning. Before we attempt to perform any processing to make sense of the text, it helps to remove nonsensical material. Imagine that you are developing a bot to lift people's spirits by analyzing their moods. To develop the best response, one must now examine the sentiment contained in the user's text input. Here, we should remove the stop words that are the data's noise before performing basic sentiment analysis (Raj, 2018).

4.5.6. Dependency parsing

Dependency Parsing is the process of examining a sentence's grammatical structure to identify related words and the nature of their relationships. Every relationship consists of a head that

modifies a dependent. Each relationship is given a classification based on how dependent the head and dependent are on one another (Jaiswal, 2021). Let's take a look at a simple example using this method.

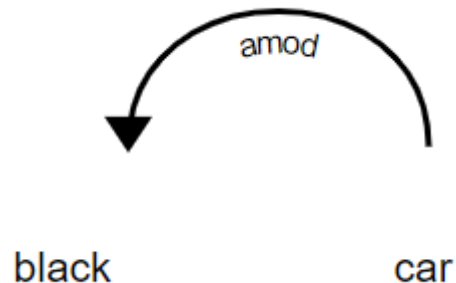


Figure 8: Simple dependency relationship between two words. Source: (Towards Data Science, 2021)

From figure 8 we can say that the *car* and *black* are in a relationship because the color *black* changes the meaning of the word “*car*”. In this case, the *car* serves as the **head**, while *black* represents the head's **dependent**. Here, the relationship is of the type "Adjectival Modifier," or **amod**. It is a noun that is modified by an adjective or an adjective phrase (Jaiswal, 2021).

4.5.7. Noun chunks

Base noun phrases, or flat phrases with a noun as their head, are what are known as noun chunks. Noun chunks can be thought of as a noun plus the phrases that describe the noun, such as "the lavish green grass" or "the largest tech fund in the world." (Spacy guides, 2022). Providing more information such as using the combination of nouns help computer program to have more data to understand.

4.5.8. Similarity between words

When using NLP, one of the most frequent use-cases is determining how similar two words are. Often determining whether two words are similar takes on significant importance. It needs frequently encounters circumstances when developing chatbots where it must not only identify terms that appear to be similar but also determine how logically linked two words are (Raj, 2018).

Word similarity, which ranges from 0 to 1, indicates how semantically similar two words are. This is accomplished by comparing word vectors in the vector space. One of the quickest NLP libraries in use today, spaCy, offers a straightforward method for this job (Uberoi, 2019).

4.5.9. Tokenization

Tokenization is the process of dividing the text into a list of tokens from a string of text. Tokens can be viewed as components, like how a word functions as a token in a phrase and how a sentence functions as a token in a paragraph (Gupta, 2020). Types of tokenization

- Text into sentences tokenization
- Sentences into words tokenization
- Sentences using regular expressions tokenization.

Check out the tokenization process.

“Welcome to the AI chatbot”

Output:

‘Welcome’,

‘to’,

‘the’,

‘AI’,

‘Chatbot’.

5. Tools to build a chatbot

A chatbot can be built into a specialized computer program that we call a “tool”. Chatbots are mainly developed in two ways:

- **Easy way.** This method is quite easy as it says. This allows every advanced computer user to develop a chatbot using third-party software/platforms with pre-built features and pre-trained models to build a chatbot with no coding knowledge.
- **Hard way.** This way is not too hard to build a chatbot but if we take time and effort we can learn step-by-step. This method needs some coding experience and depending on the type of chatbot needs to learn some techniques.

Platforms offering an easy way of building chatbots:

1. Dialogue Flow – from Google.
2. Watson – from IBM.
3. Bot Framework – from Microsoft.
4. Wit.ai.
5. Intercom.

Building a chatbot using an easy way or using third-party software is typically a good fit for users with non-technical backgrounds and knowledge of codings such as sales managers and product managers and so on. So, they can develop in a very short amount of time and have the benefit of chatbots as there are stable, reliable, and secure platforms and mainly with technical support 24/7 which is crucial for them. I think this method is suitable especially for small-size companies as they don't need and can't afford IT departments like hotels and restaurants. Some disadvantage of this method is the provider company may have an access to the company's data and which is so sensitive part of the digital platforms. Another thing is the platforms use specific algorithms which might not be an excellent solution for some specific tasks, which means the user can't have and can't make any changes in the platform even if the company – the user decides to launch its IT department in the future.

Developing a chatbot the hard way also offers really good benefits if we decide to go that way. As we described above about the benefits of using ready-made platforms for building chatbots, this method gives more diverse, more opportunities, and more choice on how to build, which algorithms are we going to use, what industry, feature, and shortly and specifically to say it's much more customizable and mainly its completely free of charge. So, in this thesis paper, we are going to build an AI-based chatbot the hard way which means we are going to use so-called “open-source” tools which we are going to describe more in detail in the upcoming

chapters. Even if the user doesn't have experience only needed thing is time, effort, and patience to learn how to build using open-source tools and it's so interesting and joy.

5.1. Python

Python is an object-oriented, interpreted, and interactive programming language. Python is the tone of the popular language among 80% of developers. It offers high-level data structures like dictionaries, modules, classes, exceptions, list and associative arrays, dynamic typing, automatic memory management, and dynamic binding. Although it has a very simple and clean syntax, it is a powerful and adaptable programming language (Sanner, 1999).

Machine learning, data science, and artificial intelligence processes are made easier by Python's large library system. Additionally, it is commonly used in data analytics because there are numerous packages designed to carry out a variety of tasks, including, machine learning, deep learning, natural language processing, web scraping, desktop application creation, frontend and backend web development, and many more. Python uses a collection of machine-learning methods to create chatbots that can respond in a variety of ways. With the help of this feature, programmers may create Python chatbots that can converse with users and give key information. In the next chapter, we are going to cover some python frameworks/libraries which are mostly used nowadays respectively related to our topic.

5.2. Open-source libraries to develop a chatbot

In computer science, the term "library" refers to a collection of precompiled, reusable files, functions, scripts, routines, and other resources that computer programmers may resort to, frequently for the creation of software. Any library that possesses an open-source license and is free to reuse, edit, and/or publish without restriction is referred to as an open-source library (Heavy AI, 2021).

Open-source platforms offer developers an easier way to build dynamic interfaces by storing readily accessible and frequently used routines and resources like documentation, data, message samples-written scripts, classes, configuration data, values, and type specifications. These precompiled modules are arranged and saved in object format so that different, unrelated programs can utilize them. A General Public License, used by open-source libraries, ensures that users have the right to freely run, examine, distribute, and change the program (Heavy AI, 2021).

As we're going to build our chatbot program using python as a programming language here are some python open-source libraries for machine learning:

- NumPy

- SciPy
- Scikit-learn
- TensorFlow
- Keras.

Here are some benefits of using open-source platforms/libraries:

- **Community.** Open-source libraries are run by a huge community of talented developers with the common goal of improving each other and solving issues too.
- **Transparency.** Full access to the whole code for everyone. This allows programmers to form expectations about the environment they will be working in.
- **Cost.** By avoiding license fees, open-source libraries and other solutions minimize the overall cost of deployment.
- **Security.** Finding and fixing security vulnerabilities is more likely when a big number of developers are active in the creation of open-source solutions.
- **Reliability.** Open-source software and libraries are rigorously evaluated by a broad and sizable set of individuals, making them more robust and dependable (Heavy AI, 2022).

In the next upcoming chapters, we are going to explore only the open-source libraries with mainly focus on their usage in building chatbots and virtual assistants.

5.2.1. SpaCy

SpaCy is an open-source, free Python library that is ideal for those who work with a lot of text. It enables you to create apps that need to handle a lot of text and is intended for usage in production. SpaCy may be used to create systems for text pre-processing before deep learning, natural language interpretation, and information extraction. From linguistic notions to machine learning skills, spaCy offers a variety of features and capacities (Syal, 2021). Some of its features are:

- Tokenization
- Part-of-speech (POS) Tagging
- Dependency Parsing
- Lemmatization
- Named Entity Recognition (NER)
- Similarity
- Text Classification
- Training

- Serialization

5.2.2. Rasa

For developing chatbots and AI-based assistants, Rasa is an open-source machine learning framework. Most of the time, you can operate in Rasa without any prior programming language knowledge. Even though there is an item called "Rasa Action Server," which requires you to write Python code, this is mostly used to start external operations like using Google API or REST API, etc (Sundaray, 2019). Rasa NLU is not simply another library with several methods for performing various tasks. It provides the potential to create practically any type of chatbot you can think of. Rasa gives you the supernatural ability to teach the computer to understand a text's meaning rather than having to write rules to do it (Raj, 2018).

Rasa has two main modules:

1. **Rasa NLU** for understanding user messages.
2. **Rasa Core** for holding conversations and deciding what to do next.

Rasa NLU - In this section, rasa aims to understand user communications to identify the message's intent and entity. Rasa NLU's many intent and entity recognition components, the majority of which have some extra dependencies, are divided into several categories (Sundaray, 2019). There are 1. Spacy and 2. Tensorflow needed to install while using rasa nlu.

Rasa Core – Here Rasa will try to assist you here by providing a contextual message flow. It can forecast discussion as a response and can activate Rasa Action Server based on User messages (Sundaray, 2019).

5.2.3. TensorFlow

TensorFlow is one of the most popular and powerful free and open-source libraries for artificial intelligence and machine learning. It provides an opportunity to use across a range of tasks however focuses specifically on training and interpretation of deep neural networks. It was designed by the Google Brain team for research and production development purposes and was written in Python, C++, and CUDA languages in 2015.

The greatest library is TensorFlow because it was designed to be user-friendly for everyone. The TensorFlow library uses a variety of APIs to build deep learning architectures like CNNs and RNNs at scale (Rungta, 2018). It offers to build end-to-end machine learning models along with pre-trained models to make the workflow easier and faster.

As for our job even if we are going to build a not complicated chatbot based on deep learning methods, TensorFlow can help us to deploy or to train our model as it is a very complex

and huge platform.

5.2.4. NLTK

The Natural Language Tool Kit, sometimes known as NLTK, is an open-source collection of modules and tools for creating Python-based systems. Along with a collection of text processing modules for tokenization, stemming, tagging, parsing, classification, and semantic reasoning, the toolkit offers simple interfaces with several corpora and lexical resources, such as WordNet (Loper, Edward & Bird, Steven, 2002).

5.2.5. NumPy

The Python package NumPy is used to manipulate arrays. Additionally, it contains matrices, Fourier transform, and functions for working in the area of linear algebra. In the year 2005, Travis Oliphant developed NumPy. Everyone can use it for free because it is an open-source project. Numerical Python is referred to as NumPy. The equivalent of arrays in Python lists, although they take a long time to execute. The goal of NumPy is to offer array objects that are up to 50 times quicker than conventional Python lists. The NumPy array object is referred to as an array, and it has several supporting methods that make using ndarray relatively simple. In data science, speed, and resources are crucial, and arrays are applied a lot (W3Schools, 2023).

5.2.6. SciKit-learn

The most reliable and powerful Python machine-learning library is Scikit-learn. Through a Python consistency interface, it offers a variety of effective tools for statistical modeling and machine learning, including classification, regression, clustering, and dimensionality reduction as well as for unsupervised learning tasks including dimension reduction (presenting data in a space of reduced dimension with little loss of meaningful information), anomaly detection (Tutorials Point, 2022).

5.3. PyCharm

It is the most widely used IDE for Python scripting. In the following areas, PyCharm provides its customers and developers with some of the best features (Tutorials Point, 2022):

- completion and inspection of the code
- Support for web programming with frameworks like Django and Flask
- advanced debugging

A developer will also find working with PyCharm to be comfortable due to the characteristics listed below:

- Code Completion
- SQLAlchemy as Debugger
- Git Visualization in Editor
- Code Coverage in Editor
- Package Management
- Local History

PyCharm offers a free version for non-commercial purposes and a professional edition that is paid version but the company offers a free subscription for one year for students after verifying the student status and has been developed by JetBrains which is originally from Prague. In this I'm going to use the professional version of PyCharm as a student I'm benefiting.

6. Practical part

6.1. Building customer-support chatbot

We are going to design a chatbot using Rasa open-source, TensorFlow, and other open-source libraries, and will be integrated into Facebook Messenger. It's going to be a task-oriented chatbot and will serve to support the customers as we mentioned at the beginning of the thesis. The bot can be implemented in the hospitality industry respectively to the hotels. We name it Virtual Assistant.

The scope of the chatbot explains to us – what it is capable of and to what degree. The following chatbot will answer specific hotel-related questions. The scope of the chatbot:

- greeting the user.
- understanding the user intent.
- replying to users meaningfully.
- answering the user's questions about what was trained for.
- helping the user to book a room.
- helping the user to find places to visit.

Except for text-based interaction, our chatbot will have the following features:

- interaction with the customers via Facebook Messenger channel
- clickable buttons.
- display images.
- clickable buttons linking to another page.

Along with features and capabilities, our chatbot will perform a task that is supporting the hotel's clients before and during their stay 24/7.

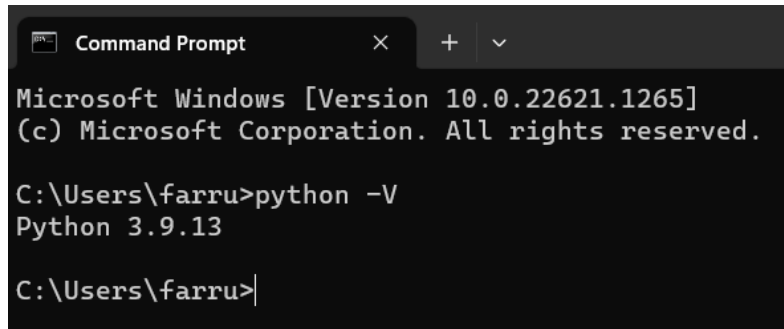
6.2. Setting up the environment

To start building the chatbot the computer needs to get installed the required programming language, dependencies, code editors, distributions, libraries, and packages. My machine runs Windows 11 operating system.

We are going to start with Python programming language installation. The following steps help to get this done:

1. Download the Python installer from the official website of Python organization the versions you need to use.
2. Running the executable python setup file – choose the path on your drive you need to install and wait until the installer finishes.

3. Verifying the python installation on the machine – to do so: press start-type *cmd* then once the command prompt is running type *python -V*.



```
Command Prompt
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Users\farru>python -V
Python 3.9.13

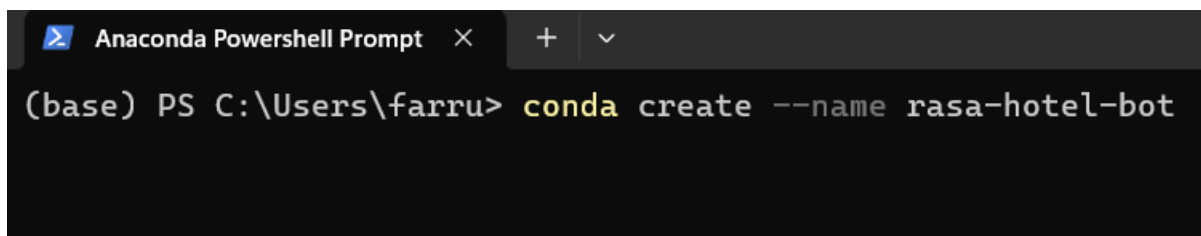
C:\Users\farru>
```

Figure 9: Python version verification

Next, we need to install a code editor I have already installed Visual Studio Code on my machine but I'm going to use IDE (integrated development environment) Pycharm from JetBrains company. The following steps need to be done to get installed PyCharm:

- Go to their JetBrains organization webpages - choose OS and choose the type of PyCharm they provide a free version named “Community” and a “Professional” paid version but they offer a free subscription for the student so I'm benefiting from using the Professional version.
- Installation can be done from the executable setup file.

Now we have already Python and PyCharm installed on the machine. The next step is installing the Anaconda distribution. But before that, it's required to install dependency for the TensorFlow which then will be installed for that we need to download and install Microsoft Visual C++ Redistributable from the Microsoft website by searching and then need to choose the right architecture according to our machine. Next would be the Anaconda installation. It's an open-source Python distribution popularly used in data science which simplifies package management and deployment. We shall use Individual Edition or Anaconda distribution. Can get it from the Anaconda company website. Once it's installed in our machine, need to run Anaconda PowerShell Prompt and then need to create an environment. We name our environment “rasa-hotel-bot”.

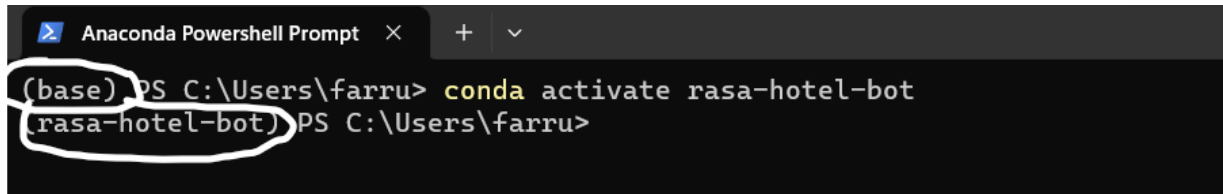


```
Anaconda PowerShell Prompt
(base) PS C:\Users\farru> conda create --name rasa-hotel-bot
```

Figure 10: Creating an environment in Anaconda

Anaconda asks to confirm the creation of the environment we need to press ‘y’ to confirm. Our

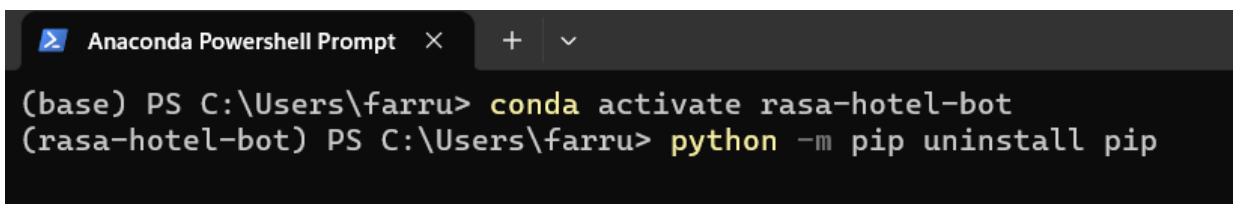
environment is ready to use, the following command helps to activate and deactivate our environment.



```
Anaconda Powershell Prompt x + v
(base) PS C:\Users\farru> conda activate rasa-hotel-bot
(rasa-hotel-bot) PS C:\Users\farru>
```

Figure 11: Environment activation in Anaconda

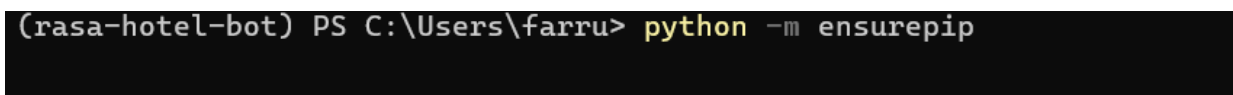
As the Anaconda terminal shows we have activated our environment. The next step is to make sure that is our pip up to date. Pip is the package management system in Python that deals with the installation of software packages. For that, we need to uninstall the pip first.



```
Anaconda Powershell Prompt x + v
(base) PS C:\Users\farru> conda activate rasa-hotel-bot
(rasa-hotel-bot) PS C:\Users\farru> python -m pip uninstall pip
```

Figure 12: Uninstalling the old version of the pip package.

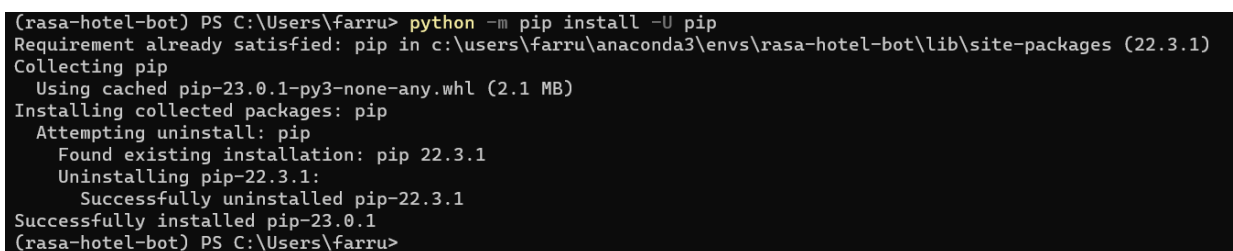
Now we need to install back the pip.



```
(rasa-hotel-bot) PS C:\Users\farru> python -m ensurepip
```

Figure 13: Ensuring pip package.

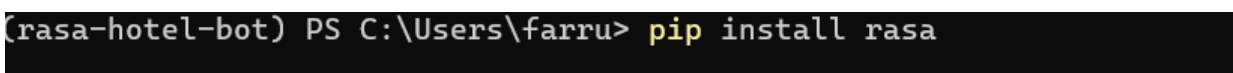
Next, it's required to install U pip



```
(rasa-hotel-bot) PS C:\Users\farru> python -m pip install -U pip
Requirement already satisfied: pip in c:\users\farru\anaconda3\envs\rasa-hotel-bot\lib\site-packages (22.3.1)
Collecting pip
  Using cached pip-23.0.1-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
  Successfully installed pip-23.0.1
(rasa-hotel-bot) PS C:\Users\farru>
```

Figure 14: Reinstalling pip package.

As our last window shows that we have the latest version of pip – 23.0.1. We are going to install rasa in our environment.



```
(rasa-hotel-bot) PS C:\Users\farru> pip install rasa
```

Figure 15: Installing rasa open source

Now we have installed the latest rasa version on our machine.

```
(rasa-hotel-bot) PS C:\Users\farru> rasa --version
Rasa Version      :      3.4.4
Minimum Compatible Version: 3.0.0
Rasa SDK Version  :      3.4.0
Python Version    :      3.10.9
Operating System  :      Windows-10-10.0.22621-SP0
Python Path       :      C:\Users\farru\anaconda3\envs\rasa-hotel-bot\python.exe
(rasa-hotel-bot) PS C:\Users\farru>
```

Figure 16: Verification of rasa installation version

It has all the necessary libraries such as TensorFlow and other packages that have been installed during the rasa installation process. Algorithms respectively NLP and machine learning pipelines will be installed later if required. Now, our workspace is ready to start building our chatbot from scratch. If we type `rasa -h` it will show all the commands used in rasa. We are working with rasa open source.

```
(rasa-hotel-bot) PS C:\Users\farru> rasa -h
usage: rasa [-h] [--version] {init,run,shell,train,interactive,telemetry,test,visualize,data,export,x,evaluate} ...

Rasa command line interface. Rasa allows you to build your own conversational assistants 🗯️. The 'rasa' command allows you to easily run most common
commands like creating a new bot, training or evaluating models.

positional arguments:
  {init,run,shell,train,interactive,telemetry,test,visualize,data,export,x,evaluate}
    Rasa commands
  init                Creates a new project, with example training data, actions, and config files.
  run                 Starts a Rasa server with your trained model.
  shell              Loads your trained model and lets you talk to your assistant on the command line.
  train              Trains a Rasa model using your NLU data and stories.
  interactive         Starts an interactive learning session to create new training data for a Rasa model by chatting.
  telemetry           Configuration of Rasa Open Source telemetry reporting.
  test               Tests Rasa models using your test NLU data and stories.
  visualize          Visualize stories.
  data               Utils for the Rasa training files.
  export             Export conversations using an event broker.
  x                  Run a Rasa server in a mode that enables connecting to Rasa Enterprise as the config endpoint.
  evaluate           Tools for evaluating models.

options:
  -h, --help          show this help message and exit
  --version           Print installed Rasa version
(rasa-hotel-bot) PS C:\Users\farru>
```

Figure 17: Rasa command line interface

This figure displays the commands available in rasa.

6.3. Components and terminologies used in chatbot

The importance of every single component is their relationship and subsequently interacting with each other makes the whole system work as one complex and working chatbot. We are going to some introduction about the chatbot's components in rasa open source.

6.3.1. Rasa Open Source Flowchart Diagram

Rasa's architecture is designed to be able to handle growth and expansion. The following diagram presents a summary of the Rasa architecture. The two main elements are dialogue management and Natural Language Understanding (NLU) component (Rasa Documentation, 2023).

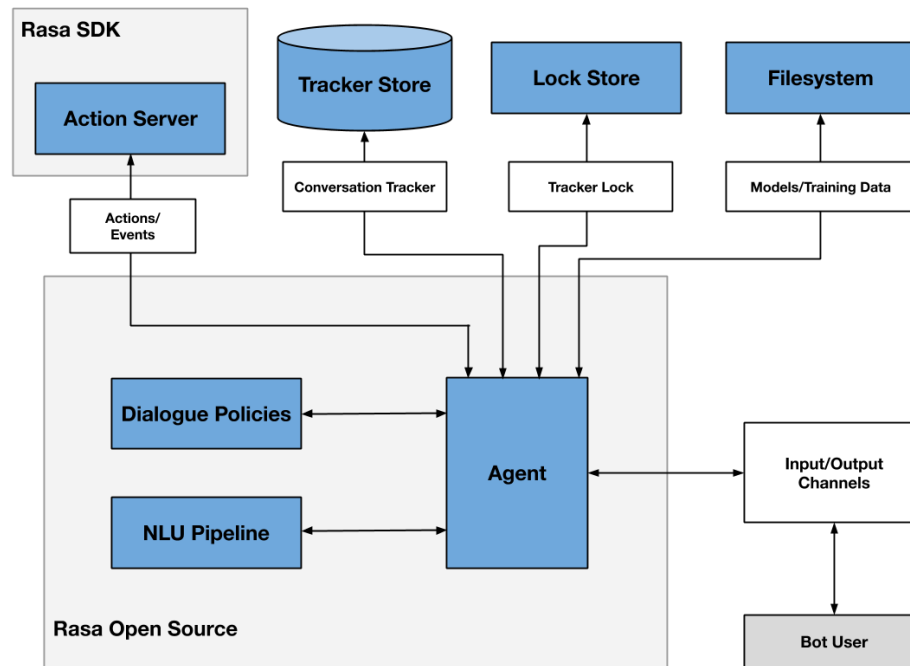


Figure 18: Rasa Open Source Architecture. Source:(Rasa documentation, 2023)

The NLU Pipeline is responsible for processing user input by utilizing an NLU model, which has been generated by the trained pipeline, to handle tasks such as intent classification, entity extraction, and response retrieval. The Dialogue Policies, as shown in the diagram, represent the component responsible for determining the appropriate next action in a conversation based on the current context. This component is known as the dialogue management component (Rasa Documentation, 2023).

6.3.2. Intents

In a chatbot, intent refers to the purpose or goal that a user wants to achieve when they enter a question during a conversation. For instance, if a user tells a chatbot to "book a concert ticket," as humans, we can understand that the user is requesting the chatbot to book a concert ticket. In the context of chatbots, such requests are referred to as "intents" (Raj, 2018). For example, the "book_concert" intent could be created for this specific request. Similarly, if a user says "I want to order pizza" or "Can you help me order pizza," these requests could be classified under an "order_pizza" intent. The number of intents that can be defined depends on the complexity and scope of the chatbot. Each "intent" can be expressed, as we humans say one "intent" in many different ways. In chatbots, each "intent" has different examples which go under a specific "intent". For example, intent "order_pizza" can be asked as "I want to order pizza", "Can I have two pizzas", "I would like to eat pizza" and many ways all the examples will be categorized and understood as intent "order_pizza". All the different intents along with possible examples should

be provided so the bot will be able to understand the user's intent even if it will be expressed in different ways. Those examples are called training data. As much as can have the bot will have the training data as more accurate and precise will the bot's response to the user (Rasa tutorials, 2021).

6.3.3. Entities

Entities provide a convenient way to extract critical information from an ongoing conversation, such as a phone number or email address, among other things. They can be used to help your chatbot identify and capture essential data from users in real-time, streamlining the conversation and enabling your bot to provide more accurate and helpful responses (Zabrzanski, 2022).

As an example, when a user says, "Book a concert ticket," the intent could be to book a ticket, and the associated entity is "concert." However, the entity could be substituted with a different keyword such as "flight" or "movie," depending on the user's specific request and the context of the chatbot's function. The use of entities allows the chatbot to capture more precise details about the user's intention and provide more accurate and relevant responses (Raj, 2018).

6.3.4. Utterances

Utterances refer to the various ways in which users may express the same question or intent during a conversation with a chatbot. These could include different phrasings, synonyms, or variations in the language that all convey the same underlying meaning. By identifying and grouping similar utterances, chatbots can improve their ability to accurately recognize and respond to user requests, making the conversation more natural and intuitive for the user (Raj, 2018).

The chatbot's utterances refer to the user's input, which can be in the form of spoken or written text. An example of an utterance is when a user types a sentence such as "What is the current time in San Francisco, California?" where the entire sentence is considered as the user's input to the chatbot (Telus International, 2021).

6.3.5. Training the chatbot

Once we have defined the intents and entities in the training data, we can use "*rasa train*" command to train the chatbot. The training process involves using machine learning algorithms to learn patterns in the training data and to predict the intents and entities of new user messages.

Training chatbots is critical for their ability to provide accurate, reliable, and personalized responses to users. By investing time and effort in training, we can create bots that are more effective, efficient, and engaging for users.


```

? Do you want to speak to the trained assistant on the command line? ☺ Yes
C:\Users\faru\anaconda3\envs\rasa-hotel-bot\lib\site-packages\sanic_cors\extension.py:39: DeprecationWarning: distutils Version classes are deprecated. Use
packaging.version instead.
  SANIC_VERSION = LooseVersion(sanic_version)
2023-03-05 16:46:33 INFO     root - Connecting to channel 'cmdline' which was specified by the '--connector' argument. Any other channels will be ignored.
To connect to all given channels, omit the '--connector' argument.
2023-03-05 16:46:33 INFO     root - Starting Rasa server on http://0.0.0.0:5005
2023-03-05 16:46:35 INFO     rasa_core.processor - Loading model models\20230305-163650-formal-inference.tar.gz...
2023-03-05 16:46:49 WARNING  rasa.shared.utils.common - The Unexpected Intent Policy is currently experimental and might change or be removed in the future
⚠ Please share your feedback on it in the forum (https://forum.rasa.com) to help us make this feature ready for production.
2023-03-05 16:46:57 INFO     root - Rasa server is up and running.
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> Hello
Hey! How are you?
Your input -> I am sads
Here is something to cheer you up:
Image: https://i.imgur.com/nGF1K8f.jpg
Did that help you?
Your input -> yes
Great, carry on!

```

Figure 22: Testing the default bot after training

The figure illustrates a basic conversation between the user and the bot in the PowerShell Prompt.

Next, I'm going to show the folder and file structure how this bot is built, and how it works. For that, we shall need to run Visual Studio Code or PyCharm and then open the provided folder where our project was created.

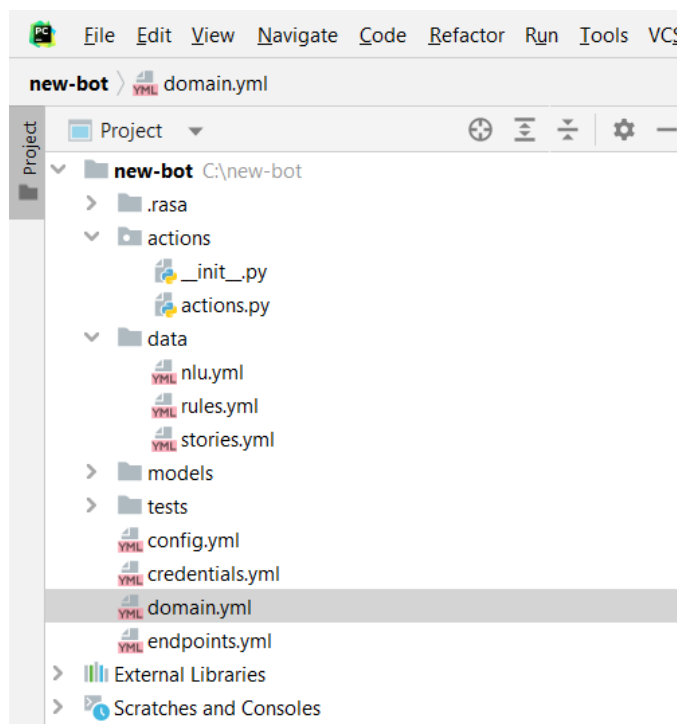


Figure 23: Rasa bot project files

The figure shows us the Rasa project that we have just created, and we can have look at its structure. How it looks like on all chatbots that were created using Rasa open source.

ere's a quick overview of some of the files that are typically created when setting up a new Rasa assistant:

Table 2 Rasa bot file structure. Source:(Rasa learning center)

config.yml	The file specifies the configuration settings for the machine learning models that are used to train the assistant. This file contains the settings for the NLU (natural language understanding) and Core components of the assistant, such as the algorithm to be used, the number of epochs, and the batch size.
domain.yml	This file defines the intents, entities, actions, and responses that the assistant will be able to handle.
data/nlu. yml	This file contains examples of user input and their associated intents and entities. It is used to train the Rasa NLU component.
data/stories. yml	This file contains example conversations between the user and the assistant. It is used to train the Rasa Core component.
data/rules. yml	Contains a set of pre-defined rules that can be used to define dialogue policies for the Rasa assistant. These rules are used to provide guidance to the assistant on how to respond to user inputs and manage the conversation flow.
endpoints.yml	This file specifies the endpoints for the assistant, such as the API endpoint for the NLU model or the webhook for external services.

Rasa commands. It is important to be familiar with a few key commands from the command line when working with Rasa. Here are some examples:

Table 3 Rasa bot commands. Source:(Rasa learning center)

Commands	Description
Rasa init	This command is used to initialize a new Rasa project. It creates a basic project structure with all of the necessary files and folders.
Rasa train	This command is used to train the machine learning models for the Rasa assistant. It reads the data files and configuration settings specified in the project directory and trains the models accordingly.
Rasa shell	This command is used to start an interactive chat session with the Rasa assistant. It allows you to test the assistant's responses and engage in a simulated conversation.
Rasa run	This command is used to start a server that can be used to interact with the Rasa assistant via an API or web interface.
Rasa test	This command is used to evaluate the performance of the Rasa assistant by running a series of tests on the trained models.
Rasa interactive	Launches an interactive shell to converse with the Rasa assistant and provide feedback on its responses. Allows developers to fine-tune the performance of the assistant and improve its accuracy by correcting intent or entity recognition

6.4.1. Creating Domain file

The domain file is the most important in Rasa's assistant. The domain.yml file serves as the configuration file for all the knowledge that the chatbot possesses. The domain file is typically a YAML file and has the following elements:

- **Intents:** A list of all the intents that the assistant can recognize, along with any example phrases associated with each intent. Below are some intents from our chatbot that will be seen when presenting a conversation between me and the bot.

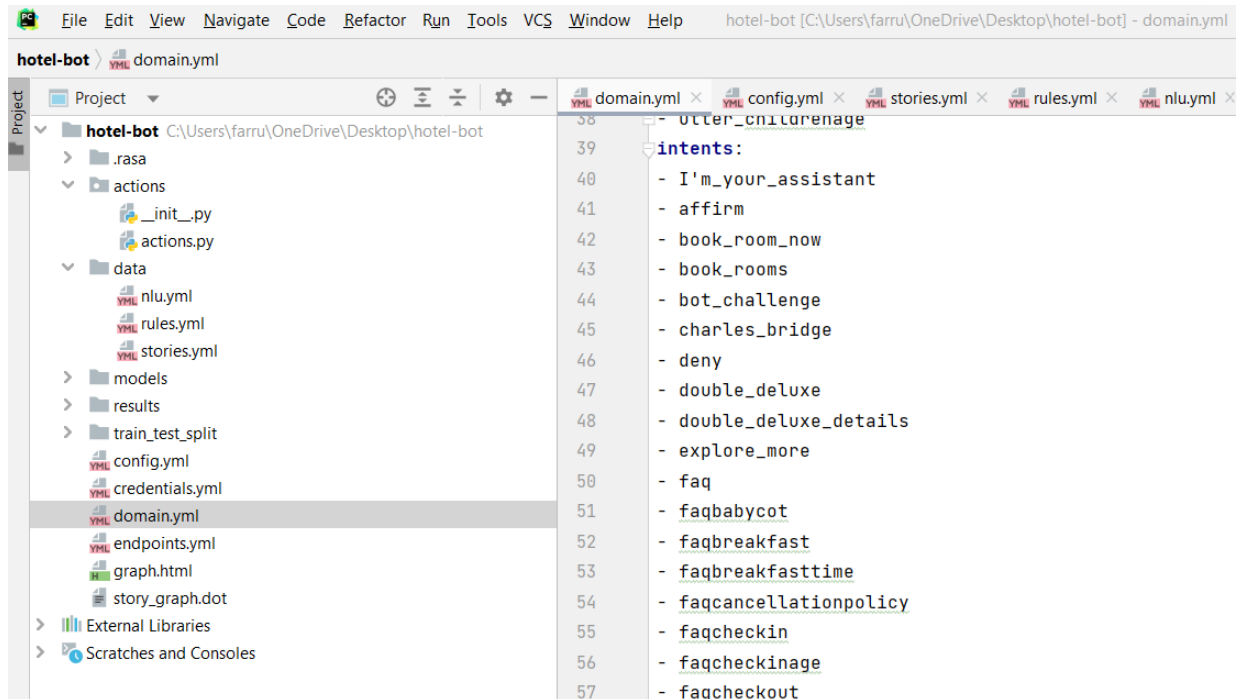


Figure 24: Creating domain file - intents

The training data for intent is stored in the nlu.yml file. Eventually, we shall need to specify all the intents that the chatbot should recognize in the domain.yml file.

- **Entities:** Entities refer to specific information that is extracted from the incoming text.
- **Slots:** Slots are variables that the chatbot remembers during the conversation. They hold information that can be used to provide more personalized and relevant responses to the user.
- **Responses:** Responses refer to a collection of all the potential replies that the chatbot can offer to the user, as well as templates that can be utilized to produce dynamic responses.

This is an example excerpt from a domain.yml file that illustrates some responses that our chatbot can generate:

```

83 responses:
84 utter_greet:
85 - text: Hi , Welcome to Miss Sophie's Charles Bridge! 😊
86   buttons:
87     - title: Visit Places
88       payload: /visit_places
89     - title: Book Rooms
90       payload: /book_rooms
91     - title: FAQ
92       payload: /faq
93 utter_babycot:
94 - text: Yes.Baby cots are available in our hotel.
95 utter_breakfasttime:
96 - text: Breakfast is served from 7:30 to 10:30, every day.

```

Figure 25: Creating domain file - responses

As per the figure, we used some higher level of response format along with buttons giving the user more flexibility and options so he can just easily click, and the bot will automatically reply.

In Rasa chatbot development, there is a naming convention for responses, whereby each response starts with “**utter**” and is followed by a descriptive term that indicates its purpose or content. Then follows the text of the response in form of “- **text:**” then we can type our response which will be displayed to the user.

If want to include the buttons in our bot it contains:

- Title,
- Payload.

In the title, we can type the simple title. In the payload after slash we should paste the intent so if we conclude the payload makes the button functional if we give the right intent from the intent list and intent has to be its answer in the response list in the form of “utter” later will be shown.

We can add images to our responses and web URL as well.

```
190 utter_prague_castle:
191   - text: ''
192     elements:
193       - title: Prague Castle
194         image_url: https://cdn-vsh.prague.eu/object/31/cd-pano-752-sal2011nik-big-m.jpg
195         subtitle: Prague Castle has been an important symbol of the Czech state for more
196         buttons:
197           - title: Read More
198             payload: /prague_castle
199           - title: Location
200             type: web_url
201             url: https://goo.gl/maps/S77ibByJgzkGo6AZ7
202           - title: Explore more places
203             payload: /explore_more
```

Figure 26: Creating domain file – response examples

The figure illustrates a response called “utter_prague_castle” this response is more advanced and comes packed in the element which will be displayed to the user all that it contains in one element. This response is the perfect example of using URLs and images in our response.

- **Actions:** The list of custom actions indicates all the actions that the chatbot can undertake during the conversation. These actions can range from generating responses to executing tasks based on user inputs.
- **Forms:** Forms refer to a collection of all the forms that the chatbot can use to gather information from the user. They are predefined sequences of questions and responses that the chatbot uses to obtain specific information needed to complete a user's request or task.

As we can see above not all elements of the domain are always used some of has an explanation

character that will be used in the advanced chatbot or that bot itself uses to understand for example we not using forms and custom actions or slots as we mentioned this is going to be simple customer support chatbot and it's not necessary to use all elements of the platform (Rasa Documentation, 2023).

6.4.2. Creating training data and rules

To develop a chatbot with Rasa, training data is required which is comprised of text data used to train models and other features. This can include user-generated text and conversational patterns, such as customer support logs or conversations with the chatbot, provided that data collection and reuse complies with your privacy policy (Rasa Documentation, 2023).

While pre-existing logs can serve as a helpful starting point, actual user interactions with the chatbot provide the most valuable training data.

In our case, I'm enough familiar with user-generated training data in hospitality as I've been working at the hotel and handled customer-related issues in the hospitality industry. I've decided to design a customer support chatbot and that lead to creating the training data for our chatbot.

In Rasa, the training data are saved in a folder called "data". The folder contains three YAML files

- nlu.yml,
- stories.yml,
- rules.yml.

The examples that represent your chatbot's intents are kept in the nlu.yml file. Below is a piece of example from our bot:

```
|- intent: faqcheckin
  examples: |
    - What is the check-in time of a hotel?
    - What are the standard check-in?
    - At what time can I check-in?
    - what is the checkin time?
    - what is the check-in time?
    - Check-in time?
    - check in time?
|- intent: more_rooms
  examples: |
    - room for 4 people
    - room up to 4 people
    - junior suite
    - Junior suite
```

Figure 27: Intents examples

As the figure shows the structure of file nlu.yml. Earlier we talked about the intents each intent has many examples of how it could be asked by the user. The user might ask for the specific thing in many ways and they are saved in nlu.yml file. In other words, we call them training data. According to the bot that we are going to build and how large we are going to be our bot we can

design more training data in the same principle as the figure shows.

Stories: stories.yml defines the flow of the conversation between the chatbot and the user. It is a part of the training data used by the dialogue management model to predict the next action to take in the conversation.

- **Story:** A story is a sequence of steps that the chatbot takes to guide the user through a conversation. Each story consists of a set of story steps, which are defined using the steps keyword.
- **Story step:** A story step is a single interaction between the chatbot and the user, such as a message or an action. Each story step is defined using the utter_ prefix for messages or the action_ prefix for custom actions.
- **Intent:** An intent is a specific goal or action that the user wants to achieve through the conversation. Each story step can be associated with a specific user intent, which is defined using the intent keyword.

Here is the example from our bot:

```
- story: faq checkin
  steps:
  - intent: greet
  - action: utter_greet
  - intent: faq
  - action: utter_faq_prompt
  - intent: faqcheckin
  - action: utter_checkintime
- story: book room now
  steps:
  - intent: greet
  - action: utter_greet
  - intent: book_rooms
  - action: utter_rooms
```

Figure 28: Stories examples

Rules: Rules are a form of training data utilized for training the dialogue management model of a chatbot. They allow for concise descriptions of conversational segments that must consistently follow a predetermined path. While a story is an illustrative example to learn from, a rule serves as a pattern that the chatbot must adhere to. This is the main distinction between the two.

Below is a sample rule that could be included as rules.yml file in the chatbots path (Rasa Learning Center, 2023).

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

Figure 29: Rules examples

This rule says: “Whenever the bot receives bot_challenge intent, the response should be always

utter_iamabot response.”

6.4.3. Creating pipelines and policies

In Rasa, machine learning is employed to predict the user's intent and the best course of action. The configuration of these machine learning pipelines needs to be set up in the config.yml file. Here are our pipelines in config.yml file looks like:

```
---
recipe: default.v1
language: en

pipeline:
  - name: "SpacyNLP"
    model: "en_core_web_trf"
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
  - 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
```

Figure 30: Rasa bot pipelines

As the figure displays, we have used multiple machine learning algorithms have been used in our chatbot to make more precise and accurate predictions, however, is also possible to use fewer algorithms and or we can make several experiments by using every single and most suitable algorithm according to our type of bot so we can compare the algorithms between each other. In the upcoming paragraphs, I'm explaining the above-illustrated pipelines, how they work, and their roles and functionalities.

Pipelines: In a Rasa pipeline, several types of components can be used to configure the machine learning models. The main components include: (Warmerdam, 2021).

- Tokenizers
- Featurizers

- Intent Classifiers
- Entity Extractors

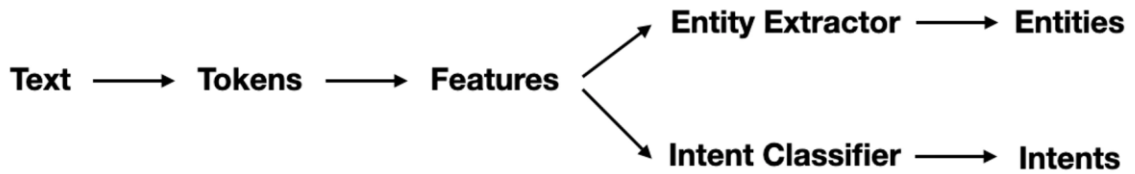


Figure 31: *Rasa bot pipeline’s architecture. source:(Warmerdam, 2021)*

Let’s comment on each pipeline’s components briefly.

Tokenizers: The initial stage of the Rasa pipeline involves breaking down an utterance into smaller text units known as tokens. This tokenization step is typically the first component in the pipeline, preceding the featurization of the text for use in machine learning models (Warmerdam, 2021). Below how the tokenizers work is illustrated.



Figure 32: *Tokenizers architecture. Source:(Warmerdam, 2021)*

Featurizers: After the text is split into tokens, the next step in a Rasa pipeline is to add numeric machine-learning features to the tokens using features. This allows the machine-learning model to work with the text (Warmerdam, 2021). The diagram below illustrates how the word "Hi" could be encoded as a numeric feature vector. Below how the featurizers work is illustrated.



Figure 33: *Featurizers architecture. Source:(Warmerdam, 2021)*

Rasa has two types of features - sparse features and dense features. Sparse features are typically generated by a CountVectorizer, and a LexicalSyntacticFeaturizer can generate window-based features that are useful for entity recognition. Dense features, on the other hand, consist of many pre-trained embeddings from language models. These embeddings are commonly obtained from spacy via SpaCyFeaturizers or from hugging face via LanguageModelFeaturizers. If we want to

use dense features, it's important to also include an appropriate tokenizer in our pipeline (Warmerdam, 2021).

Intent Classifiers: After tokenizing the input sentence, we can generate machine-learning features for each token and the sentence as a whole. These features can then be passed to an intent classification model, and we suggest using Rasa's DIET model which can manage both intent classification and entity extraction. The DIET model is also capable of learning from both token-level and sentence-level features (Warmerdam, 2021). Let's have a look at the example below using intent classifiers.

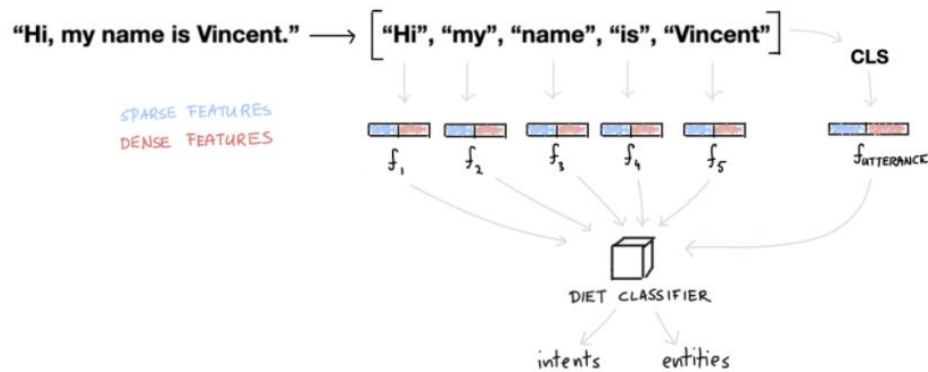


Figure 34: Intent Classifiers architecture. Source:(Warmerdam, 2021)

Entity Extractors: While DIET can learn to detect entities, it may not be the best approach for all entity types. For instance, entities that have a clear pattern, such as phone numbers, don't require an algorithm to recognize them. In such cases, using a RegexEntityExtractor can be sufficient. Additionally, it's recommended to include a DucklingEntityExtractor or a SpacyEntityExtractor in the pipeline if it is suitable for a specific use case (Warmerdam, 2021). Let's have look at the entity extractor's functional structure.

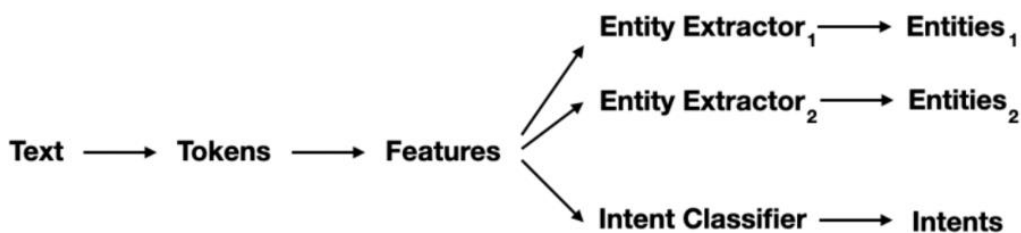


Figure 35: Entity Extractors architecture. Source:(Warmerdam, 2021)

Policies: Policies in Rasa are used to determine the next action that the conversational AI system should take based on the current conversation state. A policy takes into account the current conversation history, as well as any contextual information available, and selects the most appropriate action to take next. Rasa provides a variety of pre-built policies, including rule-based policies and machine learning-based policies such as the TED policy and the Memoization

policy. Rasa also allows for custom policies to be created and used in conversational AI systems (Rasa Documentation, 2023).

Let's have look at the policies that we have used in our chatbot.

```
policies:  
  - name: MemoizationPolicy  
  - name: RulePolicy  
  - name: UnexpectTEDIntentPolicy  
    max_history: 5  
    epochs: 100  
  - name: TEDPolicy  
    max_history: 5  
    epochs: 100  
    constrain_similarities: true
```

Figure 36: Rasa bot policies

The **MemoizationPolicy** is responsible for checking whether the current conversation matches any of the previously seen conversations in your training data, known as stories. If a match is found, it will predict the next action based on the actions that were taken in the matching story. This policy is useful for handling common conversation paths that have been seen before and does not require the use of machine learning to make predictions (Rasa Documentation, 2023).

The **RulePolicy** is designed to handle conversations that match predefined rule patterns. It allows you to create rules that specify certain actions to take based on specific conditions or user inputs. If user input matches a rule, the policy will predict the action associated with that rule. The RulePolicy can be useful for handling simple, structured conversations where you know ahead of time what the expected user inputs and bot responses will be (Rasa Documentation, 2023).

The **TEDPolicy** is a machine learning-based policy that uses a Transformer Embedding Dialogue (TED) model to predict the next best action based on the conversation history and the current state of the conversation. The TED model uses self-attention mechanisms to understand the relationships between different parts of the input, making it well-suited for natural language processing tasks like dialogue management (Rasa Documentation, 2023).

UnexpectTEDIntentPolicy is a policy that can help to analyze conversations and respond to unexpected user inputs. It is recommended to use it together with other policies as it is only capable of triggering the special `action_unlikely_intent` action. The model architecture of UnexpectTEDIntentPolicy is similar to TEDPolicy, but its task is different. Instead of identifying

the best action to take next, UnexpectEDIntentPolicy learns the set of intents that are most likely to be expressed by the user in the given conversation context based on training stories. During inference, it checks if the predicted intent by NLU is the most probable given the conversation context. If the predicted intent is likely to occur, it does not trigger any action. If the predicted intent is unlikely given the conversation context, it triggers an `action_unlikely_intent` with a confidence score of 1.00. (Rasa Documentation, 2023).

The parameter "**epochs**" in Rasa sets the number of times the training data is seen by the algorithm, with the default value being 1. One epoch refers to one forward pass and one backward pass of all the training examples. Increasing the number of epochs may help the model learn better, but it may not always improve performance. On the other hand, reducing the number of epochs may result in faster training of the model (Rasa Documentation, 2023).

The parameter "**max_history**" determines the amount of conversation history that the model uses to make the next action prediction. By default, this parameter is set to None, which means that the entire conversation history from the start of the session is considered. However, if you want the model to consider only a certain number of previous dialogue turns, you can set `max_history` to a specific value. It's important to choose a suitable `max_history` value so that the model has enough previous dialogue turns to make an accurate prediction (Rasa Documentation, 2023).

6.4.4. Training and testing the model

Before launching the training of our model, we have to make sure to adjust the `endpoints.yml` file. In Rasa, `endpoints.yml` is a configuration file that contains the endpoints that the Rasa bot uses to communicate with external services such as NLU (Natural Language Understanding) and action servers. This file is typically located in the `./rasa` directory of the project. For now, we need to uncomment `action_endpoint`.

```
12  
13 action_endpoint:  
14 url: "http://localhost:5005/webhook"
```

Figure 37: Rasa bot action endpoint

This endpoint is used to connect to an action server that executes custom actions defined in the bot's domain, in this case, it's running in our machine so that's why the server is "http://localhost:5005/webhook" the port is on default 5005. We also can specify the URL of the action server and the name of the action endpoint in this section.

In the previous sub-chapters, we have been talking about each piece of our chatbot like

Intents, training data, responses, and pipelines and policies (ML algorithms). For all those components to interact with each other and make a working chatbot I'm going to present one example below we shall then show the training process. I'm going to bring a very simple example and its greeting by the user and response by the bot then the user will ask one question after that response by the bot follows which our conversation will finish.

User: "Hello"

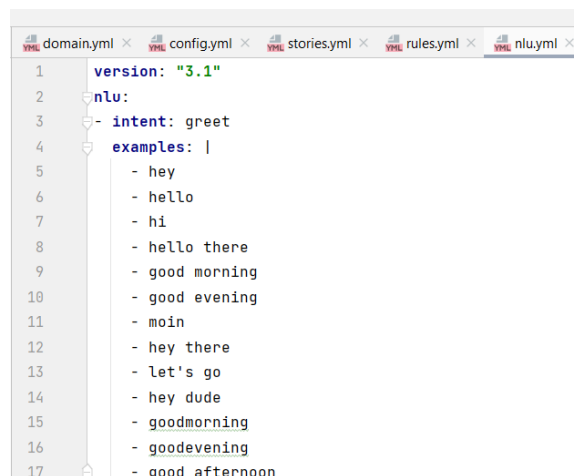
Bot: "Hi, Welcome to Miss Sophie's Charles Bridge!"

Clickable buttons: "Visit Places, Book Rooms, FAQ"

User: FAQ

Bot: "Ok, you can ask me anything regarding the Hotel"

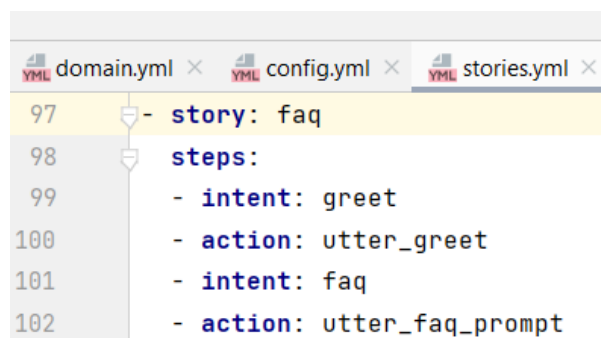
Explanation: When the bot receives "Hello" from the user it understands it as greeting intent which here:



```
domain.yml x config.yml x stories.yml x rules.yml x nlu.yml x
1  version: "3.1"
2  nlu:
3  - intent: greet
4  examples: |
5    - hey
6    - hello
7    - hi
8    - hello there
9    - good morning
10   - good evening
11   - moin
12   - hey there
13   - let's go
14   - hey dude
15   - goodmorning
16   - goodevening
17   - good afternoon
```

Figure 38: Rasa bot model explanation – intent examples

As the figure shows the greet intent is located in nlu.yml file along with examples no matter whether the user sends hi, hello or good morning the bot will accept it as greet intent. The next step of the bot is to react respectively to reply to the user so the bot looks to the stories as how it was taught what to reply when it receives greet intent. Have look at the next step of the bot



```
domain.yml x config.yml x stories.yml x
97  - story: faq
98  steps:
99    - intent: greet
100   - action: utter_greet
101   - intent: faq
102   - action: utter_faq_prompt
```

Figure 39: Rasa bot model explanation – story's examples

As the figure illustrates a story from stories.yml file first comes to greet intent we showed the value of the greet in the previous step and figure. Second is action it's the reaction of the bot to the user as we see it redirects to the response named "utter_greet". Have look at the values of the "utter_greet":

```

80 - more_rooms
81 - prague_castle
82 - visit_places
83 responses:
84 utter_greet:
85 - text: Hi , Welcome to Miss Sophie's Charles Bridge! 😊
86 buttons:
87 - title: Visit Places
88   payload: /visit_places
89 - title: Book Rooms
90   payload: /book_rooms
91 - title: FAQ
92   payload: /faq

```

Figure 40: Rasa bot model explanation – response examples

As we can see the value of the "utter_greet": is' domain.yml file and it's the same as the one we mentioned above so now we know how it works. Then the user clicks on the "FAQ" button. The FAQ is intent as figure 39 shows the next step of the bot is calling the action "utter_faq_prompt". Let's have look at the value of the "utter_faq_prompt" action:

```

113 utter_faq_prompt:
114 - text: OK.you can ask me anything regarding the Hotel

```

Figure 41: Rasa bot model explanation – response text value

As the figure displays the value from the "utter_faq_prompt" and its correct response to the user who asked "FAQ". That's how it works the bot picks correct answers thanks to the machine learning algorithms and rule-based algorithms which we have explained above in the pipelines and policies chapter.

Now we are ready to train our model and below I'm presenting training in Rasa. I'm using Anaconda PowerShell Prompt and it should be in the same folder where our project is saved. We can indeed use the terminal in Pycharm too but I like the training view in the Anaconda PowerShell Prompt. As the figure shows after successful training the trained files are saved in a folder called "models" in our main project folder. We need to run the following command: *rasa train*

page for the app. Have look at our case below.


Access Tokens

Create new Page

Generate a Page access token to start using the platform APIs. You will be able to generate an access token for a Page if:

1. You are one of the Page admins, and
2. The app has been granted the Page's permission to manage and access Page conversations in Messenger.

Note: If your app is in dev mode, you can still generate a token but will only be able to access people who manage the app or Page.

Pages ↑	Tokens
 Virtual Assistant 106087749088208	— Generate token

[Add or remove Pages](#)

Figure 44: Access tokens. Source:(Facebook App dashboard)

4. We have a page created as we can see and next, we need to click on **Generate token**. We have to keep the access token somewhere in notepad later we are going to need it.
5. Locate the **App Secret** in the app dashboard under **Settings** → **Basic**. We have to keep the secret code as well for future use.

Dashboard

- Settings
- Basic
- Advanced

App Roles

App ID	1630694490719567	App secret Show
Display name	Virtual Assistant	Namespace	

Figure 45: App secret. Source:(Facebook App dashboard)

6. We use the collected *secret* and *page-access-token* in your **credentials.yml**, and we add a field called **verify** containing a string of our choice. I put “hotel-bot” as verify. Next, we paste the *secret* code and *page-access-token* that we received from our app earlier.

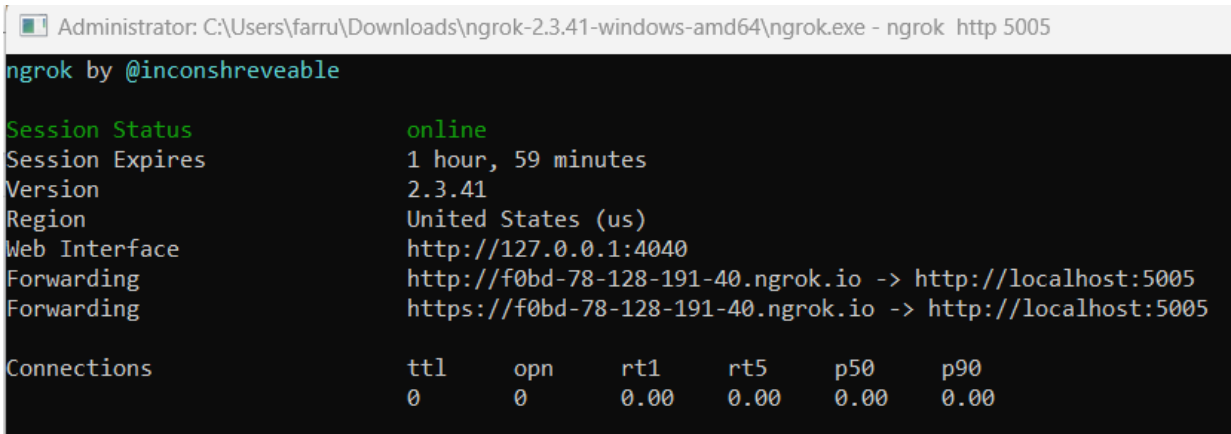
```
facebook:  
  verify: "hotel-bot"  
  secret: "  
  page-access-token: "  
  debug: true
```

Figure 46: Credentials file in Rasa – messenger adjustment

7. Now we have to set up Webhook it's in our **dashboard** – **under products** – **Webooks** – **settings** we scroll down to the **Webooks** window there we click on Edit callback URL as it says we need a URL from our server which runs our bot since we are running in our machine we need to access our localhost server to the internet.
8. To access our localhost server to the internet we use Ngrok software that helps us. We go to

their website and download the latest version if it doesn't work which happened in our case, I tried to use ngrok 2.3.41. that is a stable version so far. Next, we shall need the so-called auth-token we go to their website and sign up then we can get the authToken from the dashboard.

9. Then, we extract the archive file from ngrok, next we run it as administrator. We copy our authToken from our dashboard and paste it along with the command to the ngrok. Next, we type `ngrok http 5005` – this rasa action server as we have talked about many times earlier. Then we get the following output:



```
Administrator: C:\Users\farru\Downloads\ngrok-2.3.41-windows-amd64\ngrok.exe - ngrok http 5005
ngrok by @inconshreveable

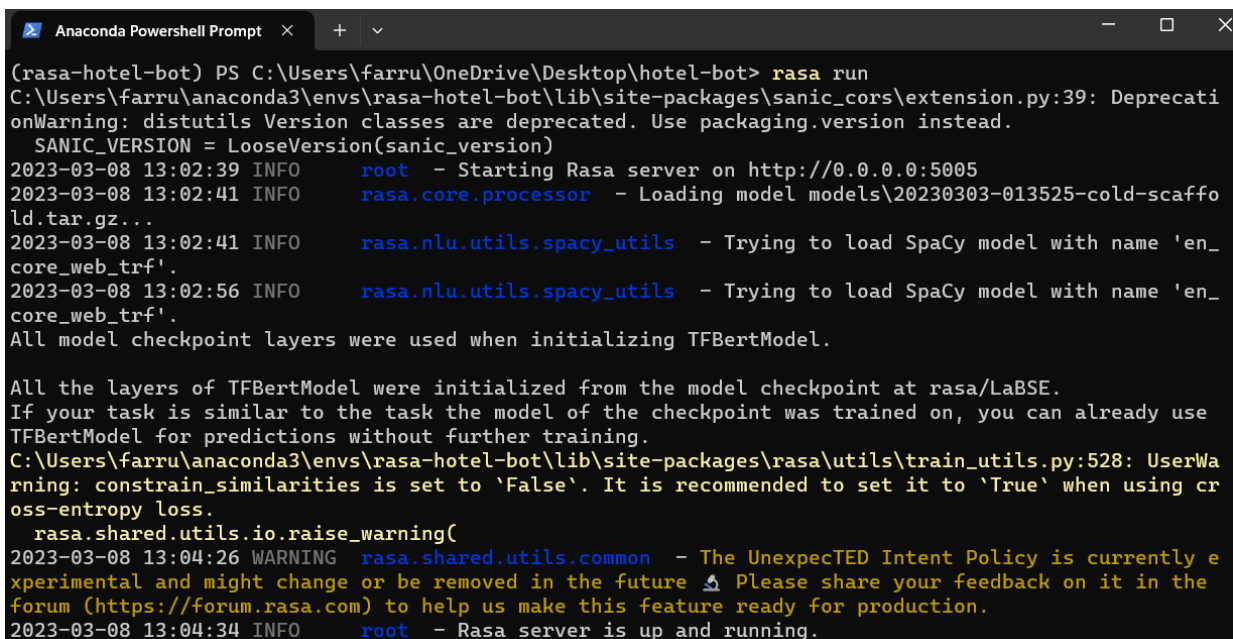
Session Status      online
Session Expires    1 hour, 59 minutes
Version            2.3.41
Region             United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding          http://f0bd-78-128-191-40.ngrok.io -> http://localhost:5005
Forwarding          https://f0bd-78-128-191-40.ngrok.io -> http://localhost:5005

Connections
  ttl    opn    rt1    rt5    p50    p90
   0     0    0.00  0.00  0.00  0.00
```

Figure 47: Ngrok interface

As the figure shows the ngrok app connected our localhost server to the internet. This is the URL <https://f0bd-78-128-191-40.ngrok.io> which we are going to paste in our Webhooks URL callback.

10. Now we should run our rasa server by the `rasa run` command.



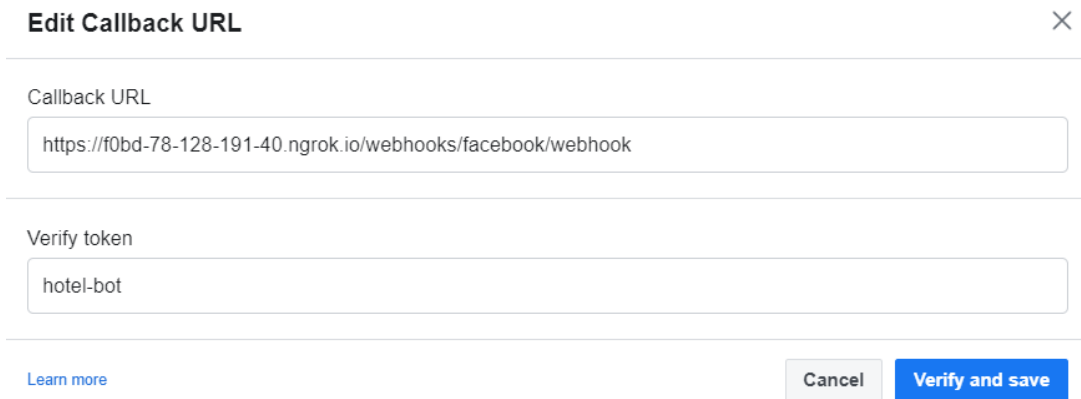
```
Anaconda Powershell Prompt
(rasa-hotel-bot) PS C:\Users\farru\OneDrive\Desktop\hotel-bot> rasa run
C:\Users\farru\anaconda3\envs\rasa-hotel-bot\lib\site-packages\sanic_cors\extension.py:39: Deprecati
onWarning: distutils Version classes are deprecated. Use packaging.version instead.
  SANIC_VERSION = LooseVersion(sanic_version)
2023-03-08 13:02:39 INFO      root - Starting Rasa server on http://0.0.0.0:5005
2023-03-08 13:02:41 INFO      rasa.core.processor - Loading model models\20230303-013525-cold-scaffo
ld.tar.gz...
2023-03-08 13:02:41 INFO      rasa.nlu.utils.spacy_utils - Trying to load SpaCy model with name 'en_
core_web_trf'.
2023-03-08 13:02:56 INFO      rasa.nlu.utils.spacy_utils - Trying to load SpaCy model with name 'en_
core_web_trf'.
All model checkpoint layers were used when initializing TFBertModel.

All the layers of TFBertModel were initialized from the model checkpoint at rasa/LaBSE.
If your task is similar to the task the model of the checkpoint was trained on, you can already use
TFBertModel for predictions without further training.
C:\Users\farru\anaconda3\envs\rasa-hotel-bot\lib\site-packages\rasa\utils\train_utils.py:528: UserWa
rning: constrain_similarities is set to 'False'. It is recommended to set it to 'True' when using cr
oss-entropy loss.
  rasa.shared.utils.io.raise_warning(
2023-03-08 13:04:26 WARNING    rasa.shared.utils.common - The UnexpectED Intent Policy is currently e
xperimental and might change or be removed in the future 🐞 Please share your feedback on it in the
forum (https://forum.rasa.com) to help us make this feature ready for production.
2023-03-08 13:04:34 INFO      root - Rasa server is up and running.
```

Figure 48: Rasa running server process

As the figure illustrates our – Rasa server is up and running.

11. Now we need to set up Webhooks.



Edit Callback URL ✕

Callback URL

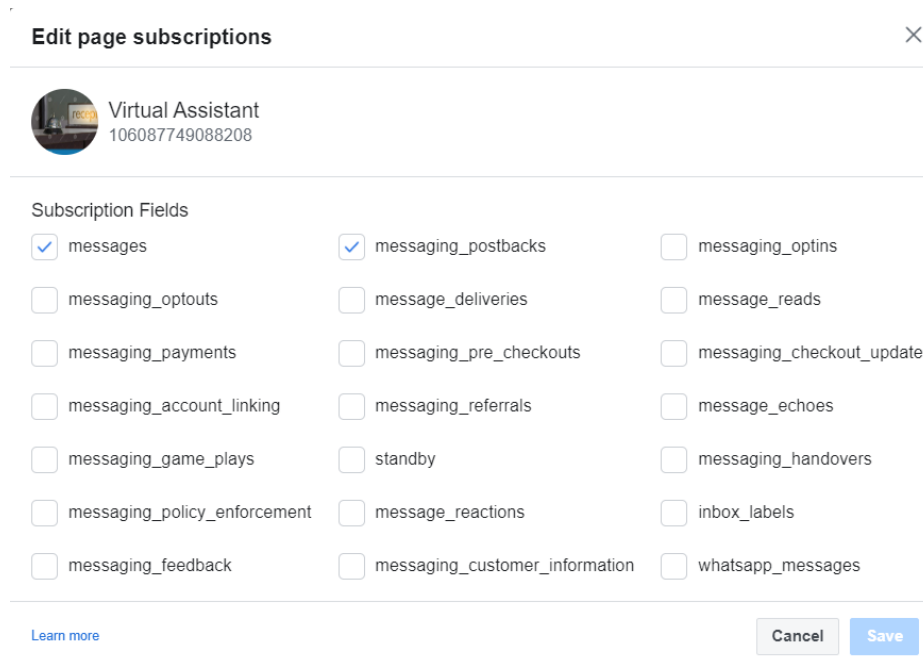
Verify token

[Learn more](#) Cancel Verify and save


Figure 49: Webhooks adjustment. Source:(Facebook webhooks)

As the figure shows here, we paste the above-mentioned URL and need paste this path `/webhooks/facebook/webhook` and in **Callback URL** then paste “`hotel-bot`” in **verify token** we have to paste as we showed in our previous figure from our **credentials.yml** file then click on **Verify and save**.

12. Next step, we have to click on Edit in the Webhooks window to set subscriptions.



Edit page subscriptions ✕

 Virtual Assistant
106087749088208

Subscription Fields

<input checked="" type="checkbox"/> messages	<input checked="" type="checkbox"/> messaging_postbacks	<input type="checkbox"/> messaging_optins
<input type="checkbox"/> messaging_optouts	<input type="checkbox"/> message_deliveries	<input type="checkbox"/> message_reads
<input type="checkbox"/> messaging_payments	<input type="checkbox"/> messaging_pre_checkouts	<input type="checkbox"/> messaging_checkout_updates
<input type="checkbox"/> messaging_account_linking	<input type="checkbox"/> messaging_referrals	<input type="checkbox"/> message_echoes
<input type="checkbox"/> messaging_game_plays	<input type="checkbox"/> standby	<input type="checkbox"/> messaging_handovers
<input type="checkbox"/> messaging_policy_enforcement	<input type="checkbox"/> message_reactions	<input type="checkbox"/> inbox_labels
<input type="checkbox"/> messaging_feedback	<input type="checkbox"/> messaging_customer_information	<input type="checkbox"/> whatsapp_messages

[Learn more](#) Cancel Save

Figure 50: Adding subscriptions. Source:(Facebook App dashboard)

As the figure shows we have marked **message** and **messaging_postbacks** subscriptions then we **save**.

The current chapter finishes with chatbot integration on Facebook Messenger. The upcoming chapter will introduce the reader to the results by presenting the running chatbot on Facebook Messenger.

7. Results and Discussion

In this chapter we are going to demonstrate to you what we have been talking about and building in the practical part of this thesis, respectively the result of the development of our chatbot and as well as the deployment of the bot on Facebook Messenger along with screenshots from the conversation between the bot and the user. Next, we are going to comment on the applied AI algorithms and pipelines and how they are affecting the conversation flow and model.

Now we set everything and finally we can test our chatbot by going to our Facebook Messenger and on the search bar typing the **Virtual Assistant** name of our Facebook page.

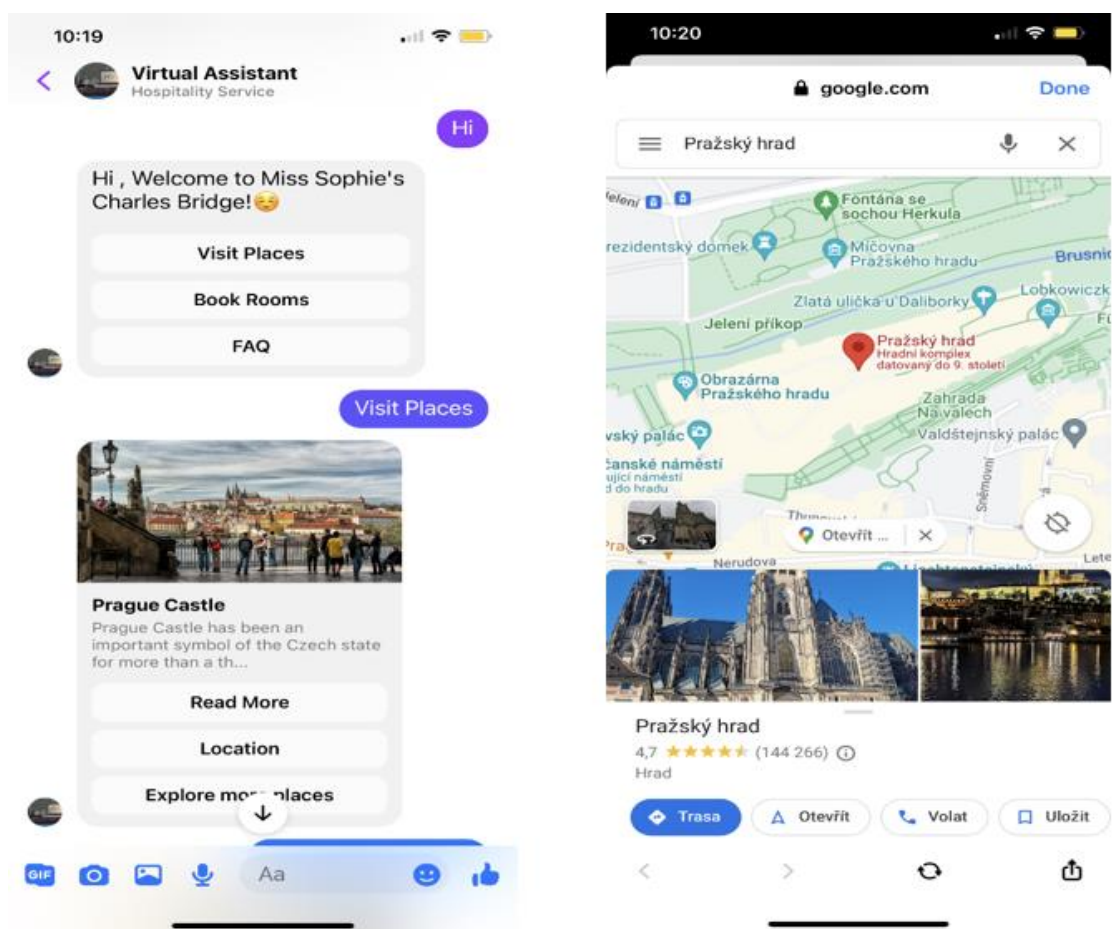


Figure 51: Bot testing in Messenger part 1. Source:(Facebook Messenger)

The figures best demonstrate our desired chatbot has been successfully built and integrated into Facebook Messenger. The screenshot between the user and the bot follows from left to right. The initial conversation starts with the user greeting and follows the bot response along with the service offer giving the user comfortable choice - nice clickable buttons so the user doesn't need even to type in this case. However, we mentioned that it's going to be a simple task-oriented chatbot. When our customers click the button of their wished service our intelligent bot replies with a well-designed element along with a picture and the necessary linkable buttons. As our

customer clicks on the “Location” button our smart bot redirects to Google Maps showing the customer the location of the chosen object by this action our bot performs a specific task and tries to help the customer. Good to indicate at this point that our intelligent bot doesn’t get away from the customer by redirecting the customer to the browser when linking to another external service but it opens in the platform so once the customer is satisfied he can easily click on the “Done” button and he is back in our chatbot in he can continue using the bot. So does the customer who’s wishing to get to know other options offered by the bot and the conversation flow moves forward by offering the next object/ response.

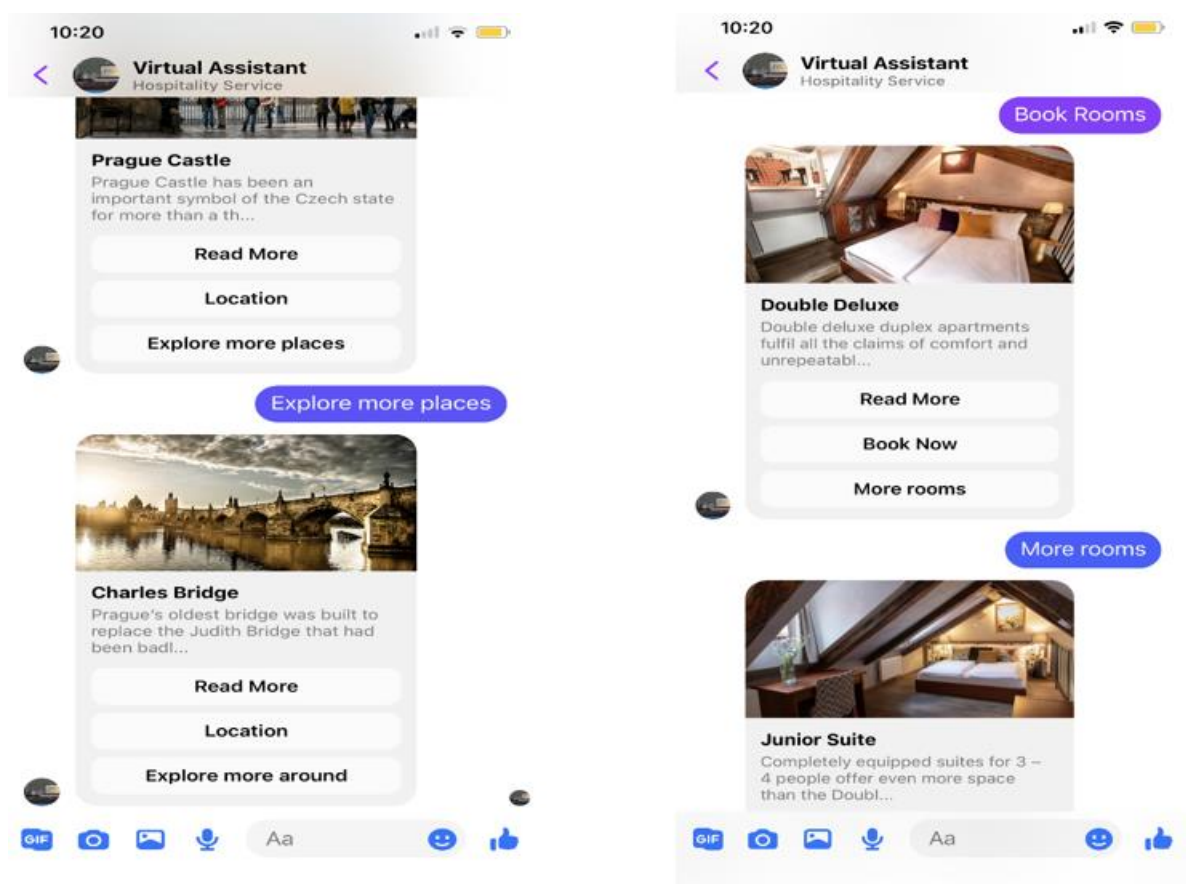


Figure 52: Bot testing in Messenger part 2. Source:(Facebook Messenger)

The upcoming screenshots are following the initial conversation with the bot. As we were talking about another service related to that button, the bot offers another nice place to visit according to the customer’s request to the bot. The next action our customer as the figures illustrate is regarding the rooms our customer clicks on the related button from the initial reply which is “Book Rooms” here the ability of our intelligent bot to back to the conversation history and meaningfully reply thanks to the algorithms of Artificial Intelligence leads to correct predictions and keeps in memory history of the conversations. The response of our bot is also an element based along with necessary linkable buttons to internal response or external services. Then,

interaction follows with the customer’s intent to show more rooms. Our bot replies with the same element-based response with images and uses buttons to help the customer with booking or by providing more details about rooms in that case. If our customer clicks on “Book Now” the bot will help to book a room in fact by redirecting to the internal booking page of the hotel. In this thesis, I’m providing screenshots from some conversations between the bot and the user only and it's not required to provide here all possible conversation cases. Let’s check our bot for normal text-based conversation in the upcoming figures below.

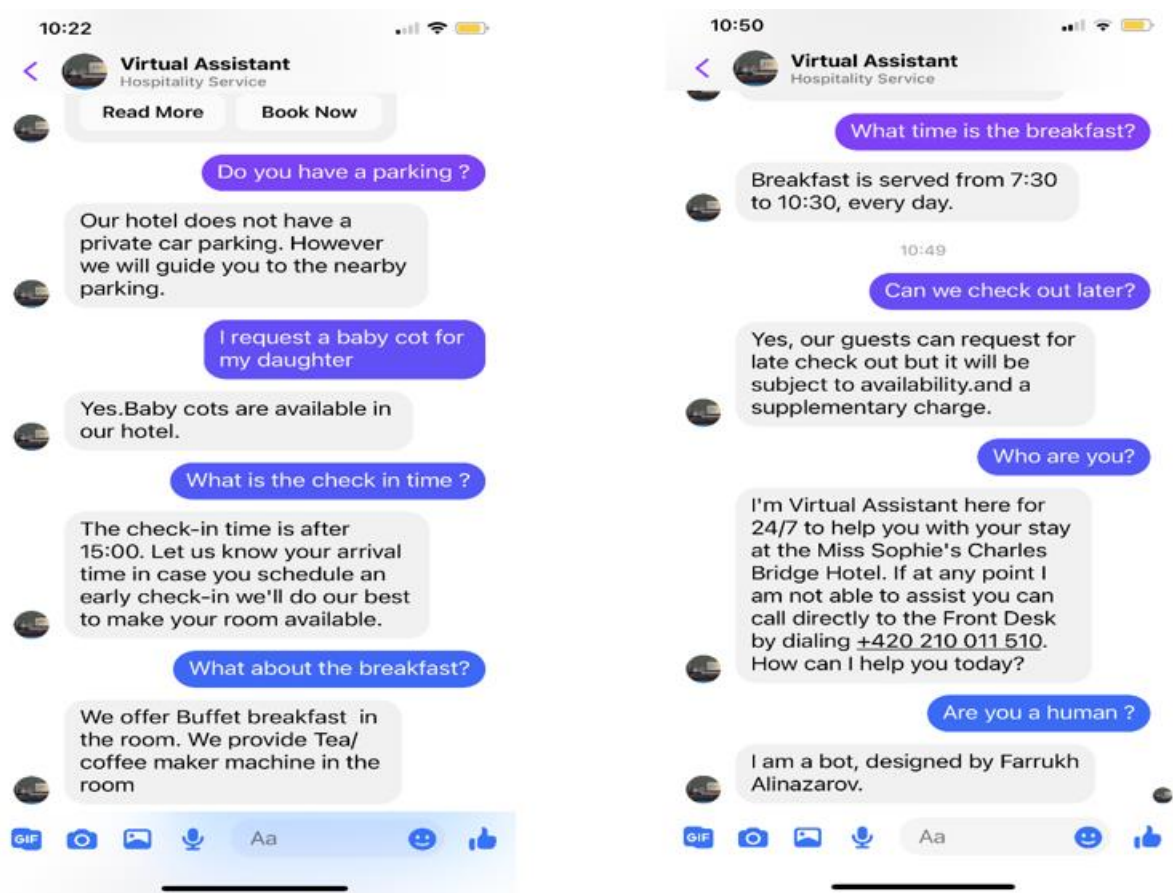


Figure 53: Bot testing in Messenger part 3. Source:(Facebook Messenger)

The above-presenting figures are following the previous conversation. This time our users have asked questions with the purpose of getting a reply in text form. Our user has used different forms of questions along with the following intents:

- Parking
- Baby cot
- Check-in time.
- Breakfast
- The time of breakfast

- Late check out.
- Bot Challenge

As our bot shows the answers are correct and meaningful thanks to the right choice of pipelines and policies. These types of exchange of information between the user and the bot fulfill the fundamental goal of this thesis and they're indicating to the reader that has been used elements of AI and a good example of the success of API usage to connect the bot to external channels like Facebook Messenger which was another goal while defining the thesis objectives at the beginning of the work.

To build this chatbot I spent more than 3 months. It consists of 1150 lines of code. For the chatbot to work properly and accurately respond to the user's query meaningfully, I have trained 62 times and a total of 62 models were needed for me to reach the goal of the thesis.

In the past ten years, there has been tremendous progress in the conversational AI sector. This expansion was amplified by the pandemic, which saw a digital transformation take center stage and become crucial for any company to survive the catastrophe. That trend will be moving forward. The trend will continue improving and **making purchases** on social platforms will be popular and easy to do especially **on Instagram, Messenger, WhatsApp, and Telegram** as these messaging platforms are widely used. AI chatbots will make a step by making the process of purchasing easy and convenient as they can provide service 24/7 and a new feature that the developers are and should work on accepting payment on the messaging platforms with help of AI-powered chatbots. Another big change is going to be the **personalization** of the services. AI chatbots would contribute on it to delivering personalized experiences. They are now helping to improve customer satisfaction already. The next step is going transforming intelligent chatbots into sales consultants at the expert level to provide the best consultation or guidance on making payments and finding the right-fit products. The next is expanding the **AI voice bots** instead of call-center agents. AI-powered automated voice assistants can help at a high level to support clients to find orders by looking in the databases or informing the customers about the status of the order or issue status thanks to APIs and voice-to-text and text-to-voice and other ML features. And revolution going the most reshaping AI technology - is building **chatbots with emotional intelligence**. Developing AI chatbots with the ability to of understanding human emotions. That includes a new study and experiments on learning and recognizing human moods, mimics, and voice tones from the videos and audios to teach the machine. So then, the AI chatbots can be customizable, where they can understand the human mood, tonality, and sentiments based on these parameters they going to provide their response to make them more accurate and more personalized. Another future trend is going to be the usage of conversational AI in Metaverse.

And of course, well-known to everyone is Chat GPT is the revolution number one in this industry and was developed by OpenAI. It's the most powerful AI-based chatbot at this time so far, the generative-based chatbot can do a lot of tasks and it is reshaping the generative AI and technology world in general.

8. Conclusion

In my diploma thesis, I demonstrated how to build a chatbot for customer support in the hospitality industry. The chatbot was developed using Rasa open-source framework and the code was written in Python. The SpaCy, TensorFlow, NumPy, and many other libraries (which come with Rasa open-source package) have been applied to help the machine to understand human language as machine and deep learning methods. As our results show, using the above-mentioned algorithms of AI the bot can reply to the user's query accurately. Nowadays companies engage customers from social media thus the bot was integrated into Facebook Messenger. The bot can be easily integrated into other social and travel media channels using API.

In the theoretical part of the thesis where we have clarified the definition of a chatbot as well as its application in different business industries. The benefits and outcomes of using chatbots presented in this thesis indicate the growing demand for their application in our lives. According to our study, we distinguish two types of chatbots: Rule-based and AI-based. We investigated how the bots used to be rule-based with limitations and with the exploration of AI and various machine learning algorithms, they became powerful by understanding natural language and performing multiple tasks. Thanks to the deep explanation of the chatbot's architecture of every component of the chatbot as well as their relationship with each other – helped me a lot with further elaboration in the practical part during the designing process of a chatbot. Thanks to open-source tools today, everyone can learn and build chatbots with the features and algorithms they want. It's free and more secure rather than using ready-made paid platforms.

In the Results and Discussion chapter, we have presented the fully functional AI-powered chatbot for customer support in the hospitality industry which was deployed in Facebook Messenger along with the screenshots of testing the bot by having a conversation with it.

We can continue to improve the following chatbot by adding more features and by more training data as well as training them. The chatbot has been designed for hotels and hostels so the hoteliers can benefit from this bot. The key features which can be beneficial are:

- non-stop customer supports
- more direct bookings.
- trip advisor for staying customers.
- helps to offer personalized services.

9. References

- AL MOUBAYED, S., BESKOW, J., SKANTZE, G., GRANSTRÖM, B. (2012). *Furhat: A Back-Projected Human-Like Robot Head for Multiparty Human-Machine Interaction*. In: Esposito, A., Esposito, A.M., Vinciarelli, A., Hoffmann, R., Müller, V.C. (eds) *Cognitive Behavioural Systems*. Lecture Notes in Computer Science, vol 7403. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-34584-5_9
- BALAKRISHNAN, Harikrishnan NELLIPPALLIL, Aditi KATHPALIA, Snehanu SAHA, and Nithin NAGARAJ. (2019). *ChaosNet: A chaos based artificial neural network architecture for classification*. Review of. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29 (11):113125.
- BARKER, Shane (2022) *AI Chatbot Implementation Challenges and Benefits*. Available at <https://shanebarker.com/blog/challenges-and-benefits-of-ai-chatbots/>
- CHEN, Mingzhe, Ursula Challita, Walid Saad, Changchuan Yin, and Mérouane Debbah (2019). *Artificial neural networks-based machine learning for wireless networks: A tutorial*. Review of. *IEEE Communications Surveys & Tutorials* 21 (4):3039-71.
- CHERAGUI, Mohamed Amine (2012) *Theoretical Overview of Machine Translation*. African University, Adrar, Algeria, Icwit.
- CHNG (2020) *How does a neural network learn*, cit. 25-08-2020
- DASTRES, Roza & SOORI, Mohsen. (2021). *Artificial Neural Network Systems*. *International Journal of Imaging and Robotics*. 21. 13-25.
- DAZA, ANGEL & CALVO, HIRAM & FIGUEROA-NAZUNO, J. (2016). *Automatic Text Generation by Learning from Literary Structures*. 9-19. 10.18653/v1/W16-0202.
- DILMEGANI, Cem (2023). *Chatbots in Healthcare: Top 6 Use Cases & Examples in 2023*. AI Multiple February 2, 2023. Available at <https://research.aimultiple.com/chatbot-healthcare/>
- DUŠEK, Ondřej, Jekaterina NOVIKOVA, Verena RIESER (2020). *Evaluating the state-of-the-art of End-to-End Natural Language Generation: The E2E NLG challenge*, *Computer Speech & Language*, Volume 59, 2020, Pages 123-156, ISSN 0885-2308, <https://doi.org/10.1016/j.csl.2019.06.009>.
- GENTSCH, Peter (2018) *AI in Marketing, Sales and Service - How Marketers without a Data Science Degree can use AI, Big Data and Bots*. Palgrave Macmillan Cham. ISBN 978-3-319-89957-2
- GREKOUSIS, George. 2019. *Artificial neural networks and deep learning in urban geography: A systematic review and meta-analysis*. Review of. *Computers, Environment and Urban Systems* 74:244-56.

GUPTA, Mehul (2020) *Tokenization algorithms in Natural Language Processing (NLP)*. Medium. March 31, 2020 Available at <https://medium.com/data-science-in-your-pocket/tokenization-algorithms-in-natural-language-processing-nlp-1fceb8454af>

HEAVYAI, (2021). *Open Source Library*. Available at <https://www.heavy.ai/technical-glossary/open-source-library#>

JAISWAL, Shivane (2021). *Natural Language Processing — Dependency Parsing*. Towards Data Science, August 1, 2021. Available at <https://towardsdatascience.com/natural-language-processing-dependency-parsing-cf094bbbe3f7>

KHATRI, Chandra Anu Venkatesh, Behnam Hedayatnia, Raefer Gabriel, Ashwin Ram, and Rohit Prasad (2018). *Alexa prize—state of the art in conversational AI*. *AI Magazine*, 39(3):40–55, 2018b. DOI: 10.1609/aimag.v39i3.2810 38, 112, 148, 170

KHDIER, Hajer Y., JASIM, Wesam M and SALAH A. (2021) *Aliesawi Deep Learning Algorithms based Voiceprint Recognition System in Noisy Environment*.

KRASNOKUTSKY, Evgeniy (2022) *AI Virtual Assistant Technology Guide 2022*. Medium, March 21 2022. Available at <https://mobidev-biz.medium.com/ai-virtual-assistant-technology-guide-2022-c375d7539aa0/>

LEI CUI, Shaohan HUANG, Furu WEI, Chuanqi TAN, Chaoqun DUAN, and Ming ZHOU. (2017). *SuperAgent: A Customer Service Chatbot for E-commerce Websites*. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102, Vancouver, Canada. Association for Computational Linguistics.

LEONARD, Andrew (1996) *Bots The Origin of New Species*. Available at <https://archive.nytimes.com/www.nytimes.com/books/first/l/leonard-bots.html>

LI Deng And DONG Yu (2013) *Deep Learning: Methods and Applications*. Microsoft Research One Microsoft Way Redmond, Wa 98052; USA, Vol. 7, Nos. 3–4 (2013) 197–387.

LI, HAO, ZHIEN Zhang, and Zhijian LIU (2017). *Application of artificial neural networks for catalysis* a review. *Review of. Catalysts* 7 (10):306.

LIU, Jinjin, Yongchun CHEN, Li LAN, Boli LIN, Weijian CHEN, Meihao WANG, Rui LI, Yunjun YANG, Bing ZHAO, and Zilong HU. (2018). *Prediction of rupture risk in anterior communicating artery aneurysms with a feed-forward artificial neural network*. *Review of. European radiology* 28 (8):3268-75.

LOPER, EDWARD & BIRD, STEVEN. (2002). *NLTK: the Natural Language Toolkit*. CoRR. cs.CL/0205028. 10.3115/1118108.1118117.

MCTEAR, Michael (2020) *Conversational AI - Dialogue Systems, Conversational agents and*

chatbots. Springer Cham. ISBN 978-3-031-02176-3

MELVIN Paul Jacob, DEBANJAN Dutt, DHEERAJ Sankar N, OM Shinde, SAYANTAN Bhattacharya (2021) *Chatbots for customer support*. Publisher: www.jetir.org. ISSN 2349-5162
MONDAL, Arnab (2021) *Complete Guide to build your AI Chatbot with NLP in Python*. Analytics Vidhya, October 25, 2021. Available at <https://www.analyticsvidhya.com/blog/2021/10/complete-guide-to-build-your-ai-chatbot-with-nlp-in-python/>

NAHID, Md Mahadi Hasan & PURKAYSTHA, Bishwajit & ISLAM, Md Saiful. (2017). *Bengali speech recognition: A double layered LSTM-RNN approach*. 10.1109/ICCITECHN.2017.8281848.

PATEL, Snigdha (2022) *Top 12 Chatbots Trends and Statistics to Follow in 2023*. ReveChat December 7, 2022. Available at <https://www.revechat.com/blog/chatbots-trends-stats/>

PON-BARRY, HEATHER & WENG, FULIANG & VARGES, SEBASTIAN. (2006). *Evaluation of Content Presentation Strategies for an In-car Spoken Dialogue System*. 10.21437/Interspeech.2006-530.

PRESS, Gil (2019) *AI Stats News: 86% Of Consumers Prefer Humans To Chatbots*. Forbes October 2, 2019. Available at <https://www.forbes.com/sites/gilpress/2019/10/02/ai-stats-news-86-of-consumers-prefer-to-interact-with-a-human-agent-rather-than-a-chatbot/>

RAJ, Sumit (2018) *Building Chatbots with Python. Using Natural Language Processing and Machine Learning*. Apress Berkeley, CA. ISBN 978-1-4842-4096-0

RASA, Documentation (2023). *Introduction to Rasa Open Source & Rasa Pro*. Available at <https://rasa.com/docs/rasa/>

RASA, Learning Center (2023). *Conversational AI with Rasa*. Available at <https://learning.rasa.com/>

RUNGTA, Krishana (2018) *TensorFlow in 1 Day: Make your own Neural Network*. Independently published. ISBN-978-1720092254

SANNER, MF. (1999) *Python: a programming language for software integration and development*. J Mol Graph Model;17(1):57-61. PMID: 10660911.

SANTOS, IRIA, LUZ CASTRO, Nereida RODRIGUEZ-FERNANDEZ, Alvaro Torrente-PATINO, and Adrian CARBALLAL. 2021. *Artificial Neural Networks and Deep Learning in the Visual Arts: A review*. Review of. Neural Computing and Applications:1-37.

SCHNELLE-WALKA, D., RADOMSKI, S., MILDE, B., BIEMANN, C., & MÜHLHÄUSER, M. (2016). *Nlu vs. dialog management: To whom am i speaking?*. In Joint Workshop on Smart Connected and Wearable Things (SCWT'2016), co-located with IUI. <https://doi.org/10.1145/2898888.2898900>

org/10.13140/RG (Vol. 2, No. 1928.4247).

SHRIDHAR, Kumar (2017) *Rule based bots vs AI bots*. Medium, May 22, 2017. Available at <https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa/>

SHUBHAM (2023) *AI Chatbot for Hotels – Reduce Costs and Improve Guest Service*. Botshot.AI March 13, 2023. Available at <https://botshot.ai/resources/blog/chatbot-for-hospitality-industry/>

SHUOHUA, Zhou (2020). *Research on the Application of Deep Learning in Text Generation*. Journal of Physics: Conference Series. 1693. 012060. 10.1088/1742-6596/1693/1/012060.

SINGH, Praveen (2022). *Chatbots for Financial Services: Benefits, Examples, and Trends*. ReveChat, December 5, 2022. Available at <https://www.revechat.com/blog/chatbots-for-financial-services/>

SINGH, SHASHI & KUMAR, AJAI & DARBARI, HEMANT & SINGH, LENALI & RASTOGI, ANSHIKA & JAIN, SHIKHA. (2017). *Machine translation using deep learning: An overview*. 162-167. 10.1109/COMPTELIX.2017.8003957.

SPACY guides (2023). *Linguistic Features*. Available at <https://spacy.io/usage/linguistic-features>

SUNDARAY, Bikash (2019) *Create Chatbot using Rasa Part-1*. Towards Data Science, August 27, 2019. Available at <https://towardsdatascience.com/create-chatbot-using-rasa-part-1-67f68e89ddad>

SYAL, Anuj (2021). *Exploring spaCy: Your one-stop library to build advanced NLP products*. Towards Data Science, February 2, 2021. Available at <https://towardsdatascience.com/exploring-spacy-your-one-stop-library-to-build-advanced-nlp-products-d242d8d753af>

TELUS International, (2021). *A beginner-friendly glossary of chatbot terms*. Available at <https://www.telusinternational.com/insights/digital-experience/article/beginner-chatbot-terms-glossary>

TUTORIALSPPOINT, (2023). *PyCharm Tutorial*. Available at <https://www.tutorialspoint.com/pycharm/index.htm>

TUTORIALSPPOINT, (2023). *Scikit-Learn Tutorial*. Available at https://www.tutorialspoint.com/scikit_learn/index.htm/

UBEROI, Anannya (2019) *Python | Word Similarity using spaCy*. Geeks for Geeks, July 19, 2019. Available at <https://www.geeksforgeeks.org/python-word-similarity-using-spacy/>

VAN GERVEN, MARCEL, and Sander BOHTE. (2017). *Artificial neural networks as models of neural information processing*. Review of. Frontiers in Computational Neuroscience 11:114.

VAN, K. N. MINH, T. P. , T. N. S. B, and M. H. L. B (2018) *Text-dependent Speaker Recognition System Based on Speaking Frequency Characteristics* no. January, 2018.

VISHAL (2019) *What are the Limitations of Chatbots and Their Future Scope*. SmatBot, May 21, 2019. Available at <https://www.smatbot.com/blog/what-are-the-limitations-of-chatbots>

WARMERDAM, Vincent (2021) *Intents & Entities: Understanding the Rasa NLU Pipeline*. Rasa, March 29, 2021. Available at <https://rasa.com/blog/intents-entities-understanding-the-rasa-nlu-pipeline/>

WOLFF, Rachel (2021). *What is Natural Language Understanding(NLU)?*. Available at <https://monkeylearn.com/blog/natural-language-understanding/>

WU, Yu-Chen, and Jun-wen FENG (2018). *Development and application of artificial neural network*. Review of. Wireless Personal Communications 102 (2):1645-56.

W3SCHOOLS (2023). *NumPy Tutorial*. Available at https://www.w3schools.com/python/numpy/numpy_intro.asp

ZABRZENSKI, Dariusz (2022). *Add system entities to your Story*. Chatbot, April 4, 2022. Available at <https://www.chatbot.com/help/system-entities/how-to-use-system-entities/>