



الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research
جامعة غرداية



جامعة غرداية
University of Ghardaia
كلية العلوم و التكنولوجيا
Faculty of Science and Technology
حاضنة الاعمال لجامعة غرداية
The business incubator for the University of Ghardaia
مخبر الرياضيات و العلوم التطبيقية
Mathematics and Applied Sciences Laboratory



A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master

Within Ministerial Resolution 1275 , Graduation Certificate - Startup Institution

Domain: Mathematics and Computer Science

Field: Computer Science

Specialty: Intelligent Systems for Knowledge Extraction

Topic

Developing an Arabic Chatbot for Tourism in Algeria

Presented by:

Smahi REZGUI & Mustapha MOULAY ABDALLAH

Publicly defended on July 6, 2023

Mr. Slimane OULAD-NAOUI	MCB	Univ.Ghardaia	President
Mr. Abderrahmane ADJILA	MAA	Univ.Ghardaia	Examiner
Mr. Charaf Abdelkarim MOSBAH	MCB	Univ.Ghardaia	Examiner
Mr. Abdellah ANICHEL	MCA	Univ.Ghardaia	Examiner
Mr. Slimane BELLAOUAR	MCA	Univ.Ghardaia	Supervisor
Mrs. Oumelkheir MALKI	PhD Student	Univ.Ghardaia	Co-Supervisor

Academic Year: 2022/2023

Dédicace

...Louange à Allah, qui est le bénéficiaire de sa grâce et a accompli ce que j'ai menacé

*À la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie
et mon bonheur ; maman *Messouda *que j'adore ; je ne vois pas ma vie sans toi
maman** je t'aime ***

*À l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et
de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, à toi mon père
Mouhamemed.*

*...À mes sœurs, Hanan, Anfal, Halima et Wissam, et à mes neveux, le trio, Rajaa,
Hana et Walaa.*

...A tous mes amis de la Master 2 et ceux qui ont étudié avec moi

...A mon frère et compagnon de route, mon binome Mustapha.

*...A mon ami Safwane, Ali, Firas, Naceredine et tous les membres des familles
REZGUI et Bousiha et à tous ceux qui me connaissent*

REZGUI Smahi

Acknowledgment

I dedicate this work to

my source of inspiration and strength -my beloved parents-, who have always been supporting me. Every step of this journey has been infused with your love, guidance, and unwavering faith in my potential. In moments of doubt, you lifted me up with your unwavering words of encouragement, reminding me of my capabilities. Your presence in my life has instilled in me the resilience and determination to overcome every obstacle in the pursuit of my dreams.

To my siblings, who have been my closest allies. And to my nephew and nieces.

*To my uncles, aunts, all my cousins and all **MOULAY ABDALLAH** family members.*

*To my brother & right-hand man **REZGUI Smahi**, who always been supportive and helpful in the academic journey.*

To Firas, Saber, Yahia, Nacereddin, Zakaria And all my colleagues in the class of 2nd year master.

To my friends and to everyone who participated in this study, thank you for your time and contribution.

Mustapha MOULAY ABDALLAH

Remerciements

*Before acknowledging anyone, we would like to thank **Allah**, the Almighty and Merciful, who has given us the strength and patience to complete this modest work.*

*We would like to thank our supervisor **Mr. BELLAOUR Slimane** for his advice, support, and guidance throughout the work. His insightful suggestions and continuous support have significantly contributed to the success of this endeavor.*

*We also express our sincere gratitude to **Mrs. MALKI Oumelkheir** for her advice and encouragement throughout this project.*

We would like to express our heartfelt gratitude to all the members of the jury for agreeing to evaluate our work.

We would also like to extend our sincere thanks to all the teachers in the Department of mathematics and Computer Science at the University of Ghardaia who have contributed to our education.

we express our heartfelt appreciation to our beloved parents who have always supported us and have always been the driving force behind our success. Their belief in our abilities has motivated us to strive for excellence.

We would like to extend our warmest thanks to all those who have contributed, directly or indirectly, to our development and the completion of this project. May their efforts be recognized and appreciated.

ملخص

تواجه صناعة السياحة في الجزائر العديد من التحديات التي تقيد نموها وتقدمها. واحد من التحديات الرئيسية هو نقص المعلومات المتاحة حول خدمات السياحة في بلدنا. لمعالجة هذه المسألة، تركز هذه المذكرة على تطوير روبوت دردشة عربي مصمم خصيصًا للسياحة في الجزائر لولاية غرداية. الهدف هو إنشاء روبوت دردشة يمكنه توفير معلومات دقيقة ومفيدة للسياح، مما يعزز تجربتهم العامة. تستكشف رسالتنا العلاقة بين التكنولوجيا والسياحة، مع التركيز بشكل خاص على فوائد وتحديات روبوتات الدردشة. باستخدام منصة Rasa ونموذج GPT-3، يتم تطوير روبوت الدردشة من خلال عملية تشمل جمع البيانات وتدريب النموذج والاختبار. يتم نشر روبوت الدردشة النهائي على منصة التراسل Telegram. تظهر نتائج الدراسة نجاح روبوت الدردشة في الإجابة على أسئلة المستخدمين وتقديم توصيات شخصية. يثبت روبوت الدردشة أنه أداة قيمة لصناعة السياحة في الجزائر، تقدم خدمات فعالة ومخصصة للسياح. ومع ذلك، يُنصح بإجراء مزيد من الأبحاث لتحسين فهمه للغة الطبيعية وإدارة الحوار لزيادة الفعالية. بشكل عام، تساهم هذه المذكرة في تقديم خدمات السياحة في الجزائر من خلال معالجة فجوة المعلومات من خلال تطوير روبوت دردشة عربي. من خلال استغلال قوة التكنولوجيا، يثري روبوت الدردشة إمكانية الوصول إلى خدمات السياحة ومصادقيتها، مما يحسن في نهاية المطاف تجربة السائح العامة.

الكلمات المفتاحية : صناعة السياحة في الجزائر، شات بوت عربي، غرداية، منصة Rasa، نموذج GPT-3، جمع البيانات، فهم اللغة الطبيعية، إدارة الحوار، Telegram.

Abstract

The tourism industry in Algeria is facing numerous challenges that restrict its growth and progress. One major challenge is the lack of accessible information about tourism services in our country. To address this issue, this thesis focuses on the development of an Arabic chatbot specifically designed for tourism in Algeria for the state of Ghardaia. The goal is to create a chatbot that can provide accurate and helpful information to tourists, enhancing their overall experience. Our master thesis explores the relationship between technology and tourism, with a particular emphasis on the benefits and challenges of chatbots. Using the Rasa platform and GPT-3 model, the chatbot is developed through a process that involves data collection, model training, and testing. The final chatbot is deployed on the Telegram messaging platform. The results of the study demonstrate the success of the chatbot in answering users' questions and providing personalized recommendations. The chatbot proves to be a valuable tool for the tourism industry in Algeria, offering efficient and tailored services to tourists. However, further research is recommended to improve its natural language understanding and dialogue management for enhanced effectiveness. Overall, this thesis contributes to the advancement of tourism in Algeria by addressing the information gap through the development of an Arabic chatbot. By harnessing the power of technology, the chatbot enriches the accessibility and reliability of tourism services, ultimately improving the overall tourist experience.

Keywords: Algeria tourism industry, Arabic chatbot, Ghardaia, Rasa platform, GPT-3 model, data collection, natural language understanding, dialogue management, Telegram.

Résumé

L'industrie du tourisme en Algérie fait face à de nombreux défis qui limitent sa croissance et son progrès. Un défi majeur est le manque d'informations accessibles sur les services touristiques dans notre pays. Pour résoudre ce problème, cette thèse se concentre sur le développement d'un chatbot arabe spécialement conçu pour le tourisme en Algérie pour la wilaya de Ghardaia. L'objectif est de créer un chatbot capable de fournir des informations précises et utiles aux touristes, améliorant ainsi leur expérience globale. Notre thèse de master explore la relation entre la technologie et le tourisme, en mettant particulièrement l'accent sur les avantages et les défis des chatbots. En utilisant la plateforme Rasa et le modèle GPT-3, le chatbot est développé grâce à un processus qui implique la collecte de données, la formation de modèles et les tests. Le chatbot final est déployé sur la plateforme de messagerie Telegram. Les résultats de l'étude démontrent le succès du chatbot dans la réponse aux questions des utilisateurs et la fourniture de recommandations personnalisées. Le chatbot s'avère être un outil précieux pour l'industrie du tourisme en Algérie, offrant des services efficaces et adaptés aux touristes. Cependant, des recherches supplémentaires sont recommandées pour améliorer sa compréhension du langage naturel et sa gestion de dialogue pour une efficacité accrue. Dans l'ensemble, cette thèse contribue à l'avancement du tourisme en Algérie en comblant le fossé d'information grâce au développement d'un chatbot arabe. En exploitant la puissance de la technologie, le chatbot enrichit l'accessibilité et la fiabilité des services touristiques, améliorant ainsi l'expérience touristique globale.

Mots clés : Industrie du tourisme en Algérie, chatbot arabe, Ghardaia, plateforme Rasa, modèle GPT-3, collection de données, compréhension du langage naturel, gestion du dialogue, telegram.

Contents

Introduction	1
1 Technology and Tourism	2
1.1 Tourism	3
1.1.1 Introduction	3
1.1.2 Tourism in Algeria	4
1.1.3 Tourism Services	5
1.2 Technolgies in Tourism	5
1.2.1 Introduction	5
1.2.2 Technologies Used in Tourism	6
1.3 Chatbot Technologies	8
1.3.1 Chatbot Definition	8
1.3.2 Chatbot Categories	8
1.3.3 Type of Chatbot	9
1.3.4 Benefits of Chatbots	10
1.3.5 Challenges of Chatbots	10
1.4 Chatbot Development	11
2 Rasa Platform	15
2.1 Introduction to Rasa	16
2.1.1 What is Rasa?	16
2.1.2 Main Features of Rasa	16

Contents

2.2	Components of Rasa	17
2.2.1	Natural Language Understanding (NLU)	17
2.2.2	Dialogue Management (Rasa Core)	21
2.3	Rasa General Architecture	25
2.4	Building Chatbots with Rasa	25
2.4.1	Rasa Installation	26
2.4.2	Creating Rasa Project, Intents, Entities, and Actions	26
2.4.3	Rules and Stories	32
2.4.4	Training and Evaluating Chatbots	34
2.5	Deploying a Rasa Assistant	37
2.5.1	Telegram Application	37
2.5.2	Deploy Rasa Assistant in Telegram	37
3	State of the art	39
3.1	Introduction	40
3.2	Rule Based Chatbots	40
3.3	Generative Chatbots	41
3.3.1	Generative Text Chatbots	41
3.3.2	Generative Speech Chatbots	43
3.4	Hybrid Chatbots	44
3.5	Arabic Chatbots	46
4	Developing and building chatbot	52
4.1	Introduction	53
4.2	Chatbot Goal	53
4.3	Development Tools	53
4.4	Data Collection	53
4.4.1	Data Collection Method	54
4.4.2	Data Structure	54
4.5	Implementation of the Chatbot	55

Contents

4.5.1	Create the Project	55
4.5.2	Insert the Training Data	55
4.5.3	Model Training	58
4.5.4	Test and Evaluation	58
4.5.5	Result and Analysis	59
4.6	Deploy and Run the Chatbot in Telegram	60
4.7	Example of Chatbot Conversations	61
	Conclusion	76
	References	77

List of Figures

1.1	Chatbot classification	9
2.1	Tokenisation	18
2.2	Featurizers	19
2.3	Diet classifier	20
2.4	High-level illustration of DIET [Rasa, 2022]	21
2.5	Example of a non-linear conversation	24
2.6	Architecture of Chatbots that is developed using Rasa [Rasa, 2023]	25
2.7	Project structure in Rasa	26
2.8	Define Customer action	31
2.9	Customer Actions - Run method	32
2.10	Using GPT-3 Model	33
3.1	Steps of training ChatGPT [OpenAI, 2022]	42
3.2	Modular subsystem of Ibn-sina	47
3.3	The Framework of the Enhanced ArabChat	48
3.4	A sample conversation between a user (U) and BOTTA (B).	49
3.5	OloBot High-level Architecture	50
3.6	Flight booking DS architecture	51
4.1	An example of storing unstructured data	55
4.2	An example of storing structured data	55
4.3	Project structure in Rasa	56

4.4	Intent Training Data 1	57
4.5	Intent Training Data 2	58
4.6	An example of storing unstructured data	59
4.7	Response	60
4.8	Responses - Buttons	61
4.9	Responses - Images	61
4.10	Example response - Pictures of a club and a hotel -	62
4.11	Training stories	63
4.12	Rules	63
4.13	Hotel contact data	64
4.14	Customer Action - Hotel Contact response	65
4.15	Model Configuration	66
4.16	Model training process	67
4.17	Testing NLU data	67
4.18	Rasa Core Testing data	68
4.19	Confusion matrix for Intents	69
4.20	Confusion matrix for Actions	70
4.21	Creating a bot in Telegram with BOtFather	71
4.22	Set bot tokens in Rasa	71
4.23	Rasa server Exécution	72
4.24	Conversation between a user and a chatbot - Information about Ghardaia Province -.	73
4.25	A conversation about transportation and the city of Zelfana	73
4.26	Conversation about the equestrian club	74
4.27	Conversation about a hotel	75

List of abbreviations and acronyms

ML	<i>Machine Learning</i>
NLP	<i>Natural Language Processing</i>
NLU	<i>Natural Language Understanding</i>
AIML	<i>Artificial Intelligence Markup Language</i>

Introduction

Rapid technological advancements have significantly transformed various industries, and tourism is one of these industries. In fact, tourism plays a crucial role in Algeria's economy, and the integration of technology in this sector has become increasingly important. The emergence of chatbots as a powerful tool for customer service and engagement has gained the attention of various industries, including tourism. However, there remains a gap in the availability of Arabic chatbots tailored to the Algerian tourism sector. In this study we aim to address this gap by developing a chatbot that can cater to Arabic-speaking tourists visiting Ghardaia.

The traditional solutions, such as tourist information centers, brochures, and guidebooks, for providing information and assistance to tourists often involve human operators or static websites with limited interactivity. These approaches have several limitations, such as high operational costs, limited availability, and lack of personalization. However, with the advancement of technology, other approaches such as mobile applications, chatbots, and virtual assistants have emerged. These solutions offer more personalized and interactive experiences for tourists while also reducing operational costs and increasing availability.

The chatbot developed in this study is designed to provide timely and accurate information about Ghardaia's tourist attractions, accommodations, transportation options, and other relevant aspects. The implementation process involves data collection, the development and construction of chatbots using the Rasa platform with the aid of GPT-3 model in some aspect. Also the model training, testing, evaluation, and deployment of the chatbot on the Telegram messaging platform. The resulting chatbot demonstrates promising results in addressing users' questions and enhancing their overall travel experience.

The rest of this manuscript is organized into four main chapters. Chapter 1 explores the role of technology in tourism, chatbot technologies, their benefits and challenges. Chapter 2 provides a detailed overview of the Rasa platform, its components, and how to build chatbots using this platform. Chapter 3 presents a state-of-the-art review of existing chatbots, including rule-based, generative, hybrid, and Arabic chatbots. Finally, Chapter 4 describes our implementation phase in detail, from data collection and Arabic chatbot development to training, testing, and evaluation before its deployment on Telegram.

1

Technology and Tourism

1.1 Tourism

1.1.1 Introduction

The tourism industry recorded significant expansion in the last few decades, becoming a major socioeconomic phenomenon in the 21st century, it not only uses quantities of financial resources and employs a large number of people but also impacts society and the economy to a considerable extent, provoking interest in identifying incidences and evaluating its results [Arionesei et al., 2014].

Tourism is the field that expresses the state of a person's travel from one country to another to visit archaeological, natural sites, or contemporary sites. Tourism is one of the areas that has received interest over the years from tourists or even major organizations in countries. It is a major contributor to the economy of many countries; and the reason for creating new jobs, improving public facilities, increasing community pride, saving cultural heritage, and offering new investment opportunities. Tourism is psychologically rewarding as many people practice it to get out of the pressures of the environment in which they live in terms of work and society.

Tourists' selection of their destinations is influenced heavily by the natural and contemporary attractions, they are looking to get maximum experience through enjoying and exploring these attractions; quality of attractions has a big impact on tourists' perceptions, it takes tourists' satisfaction to higher levels.

Despite the importance of tourist attractions, the development and growth of the local tourism lie greatly in the development of infrastructure and the quality of the consumed products and services, such as accommodation and transportation, which affect tourists' satisfaction, tourism services' prices, level of services at accommodations, internal transport quality. Security and safety are the main factors that affect satisfaction.

The country's growth and competitiveness in the field of tourism depend on the development of its infrastructure, especially the facilities needed by tourists, for example, an increase in the quality of hotels can lead to an increase in the number of days spent by tourists, and thus their spending. Conversely, if the facilities are of poor quality, it expedites the tourist to change his destination; Thus spending less while giving potential tourists a negative assessment of the country.

In addition to paying attention to the development of tourism facilities and infrastructure, countries must ensure that this does not affect the country and tourist sites culturally, environmentally, or socially, that development should improve the quality of life and the local standard of living people while preserving the local culture. Residents should also take care of the facilities to ensure the destination's sustainability and thus secure additional income by maintaining the facilities and giving the tourist a good image of the place.

1.1.2 Tourism in Algeria

Algeria is among the countries that possess vast and diverse tourism potential. Tourists visit Algeria to explore its historical, archaeological, natural, geographical, and cultural areas. Algeria is located between Morocco and Tunisia in Northern Africa, bordering the Mediterranean Sea, Western Sahara, Mauritania, Mali, Niger, and Libya. It is the largest country in Africa covering an area of 2,381,741 km^2 . Algeria's tourism resources include its remarkable Grand Sud (Sahara Desert), cultural heritage, oases, architecture, and Mediterranean coastline [Jafari et al., 2000]. The diversity of tourism sources in Algeria has made it an interesting destination for different types of tourists, the country abounds with a coastline extending over 1,200 km overlooking the Mediterranean Sea, which attracts the attention of tourists interested in coastal and beach tourism. There are other types of tourism such as mountain tourism where visitors can explore the Atlas Mountains, cultural tourism where visitors can visit the ancient Roman ruins of Timgad, and urban tourism where Algiers represents the main urban city in Algeria, the capital has a lively atmosphere, vibrant arts scene and french architecture. Desert tourism also takes the greatest interest from tourists in the diversity that exists in the Algerian desert, which makes it includes several types such as archaeological tourism, agricultural tourism, spa tourism, and oases tourism.

The Algerian economy was based mainly on hydrocarbon, but after the fall of its price in the past few years, tourism gained more interest from various Algerian and international tourism agencies, startups, and the government as an effective way to raise the development of the country's economy. On the part of the state, laws and modern regulations were put in place to help develop tourism, projects also were put in place to build new tourist areas. On the other hand, we see encouragement from individuals to receive tourists by promoting the country on social media, as there were campaigns for domestic tourism as well.

Tourism in Algeria is facing several challenges, for example, it suffers from a deficit in terms of hotel facilities, food quality, qualification of the industry, and a lack of huge publicity campaigns funded by the state abroad to introduce the country contrary to what many countries pursue. All this is due to the limited financial resources of the sector and the low level of tourist awareness. To address the challenges facing the tourism sector, the government should allocate more efforts and financial resources towards building tourism infrastructure and promoting domestic and international tourism through marketing activities and media channels. It is also important to establish competitive pricing for services to increase demand. Additionally, technology must be used in the sector as it is applied in all tourist countries globally, this can contribute to taking tourism to a higher level.

1.1.3 Tourism Services

The concept of tourism services means all the services that are provided to the tourist in particular since his arrival at his destination and everything that the tourist needs to make his stay more enjoyable. It is important to work on meeting all their needs and desires until their departure and return to their home or original place of residence, regardless of the length or type of stay [malki and chenini, 2022].

The services provided by the tourism industry include :

1. **Accommodation:** It includes hotels, resorts, and motels that provide accommodation as the core service for their customers, along with other complementary services. Institutions in this field compete fiercely to provide high-quality services while seeking innovation and meeting the requirements of sustainable development through smart rooms and relying on clean, green technology. .
2. **transportation:** It includes all the means and accompanying services for transporting travelers from one place to another, whether within the tourist destination or from their place of residence. It represents an important element in assessing the attractiveness of an area and its preference by travelers over another area.
3. **Food and beverage services:** A major part of the tourism industry includes restaurants, cafes, mobile beverage carts, ice cream and hot and cold beverage stands, and others. Their services include not only the type of food but also the quality of service, interior design, equipment, attractive and modern decorations, and the provision of additional services such as internet access.
4. **Entertainment:** It refers to places that people go to for entertainment, such as parks, children's play areas, zoos, sports halls, theaters, and cinemas.
5. **Guided tours:** An essential part of the tourism industry. They provide visitors with a unique experience by taking them on a journey through the history, culture, or natural beauty of a destination.

1.2 Technolgies in Tourism

1.2.1 Introduction

Technology has grown exponentially in the past few decades, its being used in every aspect of our life; phones, computers, the internet, or even in other fields such as renewable energy and

robotics. As the field grows, technology allows us to perform tasks that would be impossible to do manually, like communicating with people all around the world, the fast access to information, or automating repetitive or dangerous tasks. Technology is all that includes hardware and software such as computers, applications, or services that are used in a specific domain.

Technology is one of the most important fields, it has shown a big potential to solve the real world's big problems. As its boundaries keep being pushed, technology shows endless possibilities of growth in major domains like computing, medicine, and especially in the tourism field.

Tourism has undergone a major transformation after the emergence of technology, which allowed tourists to access information about tourist cities and institutions. It helped countries and tourism institutions in marketing themselves and enabling customers and the public to have access and communicate with them, which may enhance their knowledge and interest in visiting the country. Technology started being used in the tourism field in the 90s as a result of the wide access to the computer; terms such as tourism information technology (IT) and E-tourism started to appear.

Not long ago was the norm for tourists to make travel plans based on suggestions made by travel agents or the media, now, tourists use technology to search for information and explore more possibilities and choices for their trips, studies about technology in tourism began to shift. The use of technology in tourism has moved through different stages. Since the 2000s, as mobile phones with touch screens became increasingly popular, technology has become a strong global transformational force in tourism, with consumer-generated content and social media has gained central importance for the economic success of destinations and businesses. Any tourism entity can now be judged and rated; recommended or disapproved [Gössling, 2017].

Artificial intelligence and machine learning as a subset of AI, the most advanced technologies nowadays are used in different ways in the tourism field, they have helped tourism to develop. Major tourism applications are developed with machine learning approaches. For example, machine learning is used to convert essential data in tourism into information like statistics, multimedia, and maps. Machine learning is also used in the different trip stages (before, during, and after); for example hotel booking confirmation, schedule recommendation for tourist's day, and analyzing reviews after the trip.

1.2.2 Technologies Used in Tourism

Many types of existing technologies have been incorporated into the tourism sector, including the following:

1. **Mobile Technologies:** Smartphones have become an integral part of our daily lives, and this has made a great focus on creating technologies in the field of tourism that fall within the category of mobile applications.

2. **World Wide Web:** the World Wide Web began to show its commercial application potential in 1994. After the wide penetration of internet technologies and the World Wide Web, tourists gained direct access to information about tourism products and organizations inexpensively and interactively [Yuan et al., 2019]. many technologies were born with the coming of the world wide web we mention, for example, some of the existing technologies:
 - *flight platforms:* Due to several applications and web platforms provided by airlines and travel companies, clients can book plane tickets which saved time and energy for travelers as they can also research their possible destination, check ticket prices and schedule flights with different companies.
 - *Reserving Hotel:* The same thing for hotels, where it is possible only through the phone to see all the hotels in a specific area and see the services provided by each hotel in addition to the possibility of displaying pictures of this hotel without the trouble of moving to this hotel
 - *Codes QR:* Also, among the technologies used in this sector, its use spread greatly during the Corona pandemic, as the user only scans the code, and then menus and their prices are displayed, and so on, to avoid any direct contact

There are also many technologies in the tourism sector

3. **Contactless payment:** One of the technologies used in the tourism sector is contactless payment. By providing this feature, the individual pays for goods and services without the need for physical contact. This method is convenient and safe for tourists because it enables them to pay on their travels without the need to exchange currency or deal in cash. Instead of swiping a magnetic stripe or inserting a credit card into a machine, the payment process takes place only by bringing the phone or payment card closer to the card reader. ¹
4. **Internet of things:** The Internet of Things (IoT) is a system of interconnected devices with unique identifiers that can transfer data over a network without human interaction. And in the hospitality industry, IoT can eliminate frequent touchpoints and improve the guest experience. One resort, the Aria in Las Vegas, has implemented IoT technology in hotel rooms. Using a tablet in the room, guests can control curtains, temperature, and lighting. Guests can also coordinate their wake-up call to include an alarm, opening the curtains, and turning on the lights all at the same time.²
5. **Robots:** Robots are used widely in this field. Robots are spread in major airports to carry or check luggage, confirm tickets, or direct the traveler through the airport. They are also used

¹<https://www.techtargget.com/whatis/feature/6-technology-trends-in-the-travel-industry>

²<https://www.techtargget.com/whatis/feature/6-technology-trends-in-the-travel-industry>

in restaurants and hotels as well, to serve the customer. Major big cities use robots due to their availability and serve customers without discrimination, which guarantees good and fast service, and all of this has positive repercussions on the tourism sector.

6. **Virtual tours:** With the development of virtual world technologies and 3D virtual tours appeared, many people use them to find out the advantages of their potential destinations or to visit areas that are difficult for them to go to for some reason, as this feature gives very close pictures of the real world as if you are visiting it at that moment
7. **Chatbot Technologies:** Also, one of the common and widely used techniques that are the focus of our topic, we will discuss in the next section

1.3 Chatbot Technologies

1.3.1 Chatbot Definition

Chatbot is a program designed to simulate human conversations with others in text or spoken format using NLP. terms such as smart bots, digital assistants, conversational agents, and interactive agents are all used to describe chatbots.

The primary purpose of a chatbot is to provide suitable responses and perform potentially specific actions based on the context of the conversation and the user's intention. It could be a stand-alone application or by adding it to a website or other applications such as Telegram, WhatsApp, and Facebook.

As AI technology continues to advance, chatbots are gaining more attention and adoption due to the improvements they achieved. More advanced chatbots are being developed and implemented in various fields such as education, entertainment, customer service, and more. The use of chatbots is expected to keep increasing in the future as research and development in this field are ongoing to improve their performance and capabilities.

1.3.2 Chatbot Categories

There are many categories of chatbots, and each category is different from the others. They are all shown in Figure1.1:

- **Interaction mode:** Chatbots can be categorized into text and/or speech conversational chatbots based on how the chatbot interacts with its input and output [AlHumoud et al., 2018].

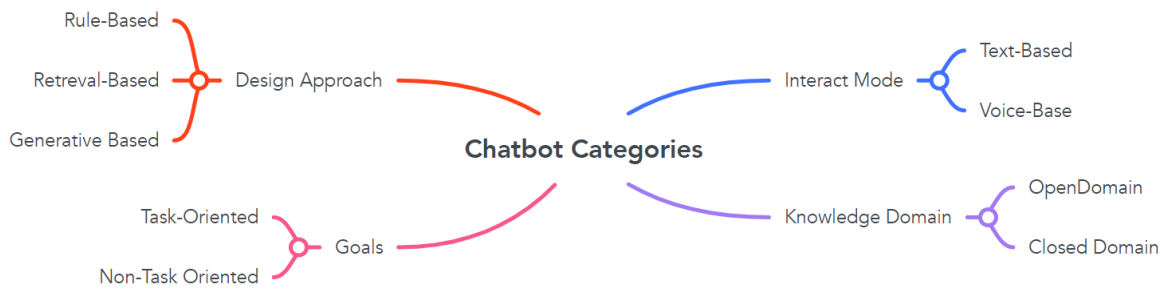


Figure 1.1: Chatbot classification

- Knowledge domain:** In this criterion, we mean here the knowledge and information that the chatbot possesses about a specific domain or topic. According to Closed domain chatbots, it is trained in a specific domain only, meaning that they will fail to answer any question outside the domain in which it was trained, while open-domain chatbots can Respond and interact on different topics.
- The Goal:** In this standard, we find two different divisions as well, task-oriented dedicated to executing and completing some tasks for users, for example, making hotel reservations, booking a flight, or requesting a specific product. as for non-task-oriented, its main goal is to provide only information about what the user asks so they behave as informative chatbots.
- The response generation approach:** According to this criterion, chatbots are classified into two categories. First, rule-based chatbots are programmed to reply to specific questions. They essentially generate responses based on predefined rules, so the user is restricted to a limited range of questions. This chatbot is ineligible for dealing with the user's spelling and grammatical mistakes in the input as well [Adamopoulou and Moussiades, 2020]. The retrieval-based model and the generative-based model are two types of AI-based chatbots. In the retrieval model, the response is generated by finding the best query matches in the pre-constructed conversational repository [Song et al., 2018].

1.3.3 Type of Chatbot

There are different types of chatbots, including the following:

- AI Chatbots:** Artificial intelligence chatbots understand language beyond pre-programmed commands. They are trained by loading large amounts of existing data of the clients, applying algorithms, and utilizing the outcomes to deliver a response. The more the AI chatbot communicates with the user, the more information it gathers. It updates its model with the help

of growing data to deliver accurate answers. This sort of AI likewise upholds conversational AI instead of unnatural, pre-programmed interactions [Trisha and Sahana, 2022].

- (b) **ML Chatbot:** As it is widely known, machine learning chatbots depend greatly on information to learn. Through analyzing patterns, prior experiences, and dialogues, these chatbots can expand their conversational skills. The principle is rather straightforward: the larger the amount of data we offer, the more capable these bots will be in managing a broader variety of conversations in the future. What is intriguing is that they create their replies based on previous data. It is remarkable to observe how chatbots that use machine learning can provide individuals with an interactive experience that is akin to interacting with a human.
- (c) **Generative Chatbot:** An open-domain chatbot program that curates one-of-a-kind language blends through autonomous generation is known as a generative chatbot. To build generative chatbots, seq2seq models that were designed for automatic translation can be utilized. Named the encoder-decoder model, this design employs both long-range and short-range LSTMs to fashion text derived from the data set. Additionally, machine translation applications can also benefit from the seq2seq model. By predicting the input word, the algorithm can then anticipate the subsequent words based on their likelihood of occurrence.
- (d) **Rule Based Chatbot:** The type of chatbots that are only able to perform tasks that have been explicitly programmed into them. They follow a specific set of rules to provide a response to the users input. They are relatively simple and has limited capabilities, but they can be useful in certain scenarios like handling repetitive tasks.

1.3.4 Benefits of Chatbots

One of the important advantages of the chatbot is its availability throughout the hours and days of the week, which is useful and may not be able to provide the average person. Also, one of the advantages of the chatbot is reducing costs and improving work efficiency, especially in routine tasks. Chatbots can learn from the user's inputs and adapt to his needs. it can also support more than one language which make it more public and can be used by people with different language. Chatterbots has shown big capabilities in so many fields, it could act like it was a human being which makes them powerful.

1.3.5 Challenges of Chatbots

- Natural language processing (NLP) is a key challenge for chatbots, as it can be difficult to accurately interpret and respond to human language. User experience is important for chatbots, and it can be challenging to design a chatbot that is intuitive and easy to use.

- Training data is essential for AI-based chatbots, and it can be difficult to obtain enough high-quality data to create an effective chatbot.
- Chatbots can sometimes provide incorrect or irrelevant responses, which can lead to frustration for users and damage the reputation of the business.
- Another thing that needs to be considered is the style of the chatbot. The user doesn't like to deal with the answering machine (which the chatbot is). They want a little bit more affecting interaction. That means chatbots need to have some attitude. It can go as far as selecting the gender of a bot. But the most common is selecting several manners of conversing – more formal, informal or flowery, or excessively minimalist.

1.4 Chatbot Development

I - The most common design techniques used to build chatbots are:

- Parsing:** It concerns taking the user request in text form as an input, analyzing and processing it by using some NLP functions to extract meaningful information that can be easily manipulated. Its main purpose is to determine the intent of the user [Gupta et al., 2020].
- Pattern matching:** To search for patterns effectively, you must determine a specific set of guidelines or rules that outline what constitutes the pattern you are seeking. Once you have established these prerequisites, you can then put these mandates into action and compare them to the data set or input to see if there is any correlation. This could involve scrutinizing particular values, analyzing the data's span or configuration, or applying different parameters as needed. From programming to data analysis and even to machine learning, pattern matching serves as a versatile tool for data manipulation and analysis. As a result, it has become a pivotal part of various applications, making it a powerful addition to any software.
- Artificial Intelligence Mark-up Language (AIML):** A coding language dedicated to creating chatbots and automated assistants, as it relies on creating rules for how the chatbot responds to different user inputs, and these rules are expressed as pairs in the form of a question and an answer, one of its most important advantages is that it can deal with different languages and the possibility of using variables as well. It includes conditional logic and others, its general form is as follows:

```
<category>
  <pattern>User Input</pattern>
  <template>Bot Response</template>
</category>
```

- (d) **Chat-Script:** Intelligent chatbots and conversational agents can be effortlessly created with ChatScript, which utilizes an open-source natural language processing engine. CS, short for ChatScript Script, is the language behind ChatScript, a rule-based system. Developers can utilize this language, which has syntax resembling C++, to craft intricate chatbots with advanced behavior and reasoning functionalities. Among the many features provided by ChatScript are machine learning algorithms, pattern matching, and context-aware responses. ChatScript can manage vast amounts of knowledge and data, thanks to its fundamental trait. By using a hierarchical form of knowledge organization and storage called Knowledge Representation (KR), ChatScript provides developers with a convenient and accessible way to manage and retrieve information via their chatbot.
- (e) **Natural Language Processing (NLP):** Investigating how computers comprehend and manipulate natural language (whether spoken or written), Natural Language Processing (NLP) forms a vital aspect of AI. Through this technology, chatbots are empowered. Moreover, NLP devises a means of transforming the user's text or speech into structured data, which in turn allows pertinent replies to be chosen for the user. LSTM, name entity recognition, vector recognition, text similarity, dialogue planning, intent classification, and lexicon are the NLP techniques presently utilized. Text similarity perceives the degree of similarity between two of texts.
- (f) **Natural Language Understanding (NLU):** Is a subfield of Natural Language Processing (NLP) that involves transforming human language into a machine-readable format. It is considered an AI-hard problem³. NLU is the technology behind chatbots, which is a computer program that converses with a human in natural language via text or voice. Chatbots follow a script and can only answer questions in that script. These intelligent personal assistants can be a useful addition to customer service⁴.
- (g) **Artificial Neural Network Models:** Regarding the era of responses, this type can be utilized for both retrieval-based and generative chatbots. The primary distinction between neural community-based chatbots and rule-based ones lies within the presence of a studying set of rules. ANN-primarily based chatbots may be categorized into supervised and unsupervised, relying on the machine mastering algorithms hired. Deep studying, a subset of machine mastering, can perform in unsupervised mode while working with unstructured or unlabeled data. Within conversational modeling, deep mastering neural networks, specifically recurrent neural networks (RNN), Sequence to Sequence (Seq2Seq), and Long Short-Term Memory (LSTMs) have won traction.

³https://en.wikipedia.org/wiki/Natural-language_understanding

⁴<https://www.techtarget.com/searchenterpriseai/definition/natural-language-understanding-NLU>

- **Recurrent Neural Network (RNN):** A recurrent neural network (RNN) is a type of artificial neural network that uses sequential data or time series data. These deep learning algorithms are commonly used for different problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feedforward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. ⁵
- **Long Short-Term Memory (LSTM):** is an artificial neural network used in the fields of artificial intelligence and deep learning. It is a powerful type of Recurrent Neural Network (RNN) that has been used in a wide range of applications such as language modeling, machine translation, and text summarization⁶. LSTM can process data sequentially and keep its hidden state through time. Unlike standard feedforward neural networks, LSTM has feedback connections⁷
- **Sequence to Sequence (seq2seq):** models is a special class of Recurrent Neural Network architectures that is used to solve complex Language problems like Machine Translation, Question Answering, creating Chatbots, Text Summarization, etc⁸. Sequence to sequence (Seq2Seq) modeling is a powerful machine learning technique that has revolutionized the way we do natural language processing (NLP)⁹

II - It is worth mentioning other methods that have appeared and are available to help build and develop chatbots, through what are called platforms.

The platforms providing automated customer service and performing tasks efficiently and effectively should be on every business’s priority list. A chatbot development platform makes it easier and more accessible to achieve that. With AI technologies at their disposal, chatbots can now offer personalized and natural interactions with users, thus enhancing the overall user experience.

Here are some of the most popular ones:

- **Dialogflow:** Designing conversational user interfaces into various kinds of systems, such as mobile apps, web apps, and bots, among others, is made easy and intuitive with Dialogflow’s platform, which provides natural language understanding. Unique and exciting interactions between customers and product producers are made possible with Dialogflow. Using text or

⁵<https://www.ibm.com/topics/recurrent-neural-networks>

⁶<https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/>

⁷https://en.wikipedia.org/wiki/Long_short-term_memory

⁸<https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/>

⁹<https://vitalflux.com/sequence-models-data-types-examples/>

audio samples gathered from mobile or voice recordings, Dialogflow can analyze different kinds of consumer information. Using synthetic speech or text messaging, there is a variety of ways that this technology can communicate with customers.

- **IBM Watson:** By leveraging conversational AI, IBM Watson Assistant utilizes models centered on deep learning, natural language processing (NLP), and machine learning to interpret inquiries and locate optimal responses while following through with the actions intended by the user. In addition, intent classification and entity recognition are employed to contextualize customer interactions and facilitate their transfer to human agents when necessary.
- **Amazon Lex:** Is a cloud-based service provided by Amazon Web Services (AWS) that enables developers to build conversational interfaces for their applications using natural language processing (NLP) and machine learning (ML) techniques. Developers can use Amazon Lex to create chatbots, virtual assistants, and other conversational interfaces that can be used for a variety of tasks such as B. Customer service, product recommendations, and scheduling
- **Microsoft Bot Framework:** It appeared in early 2016 and, like all previous frameworks, it provides techniques and resources for building a chatbot that performs tasks and answers user questions naturally.¹⁰
- **Rasa Platform:** Rasa is an open-source project based on natural language understanding (NLU) and dialogue management, and interaction, we chose Rasa to develop a chatbot for our topic, because it provides many features and tools that make building chatbots more efficient, Therefore, we will learn more about its advantages and characteristics in the next section

¹⁰Chatbot implementation with Microsoft Bot Framework, P8

2

Rasa Platform

2.1 Introduction to Rasa

This section introduces the Rasa framework and its key features, making it a robust and flexible development environment for applications in the natural language processing and machine learning domains.

2.1.1 What is Rasa?

Rasa is a machine learning framework that is open-source and used for creating conversational AI chatbots and assistants. Its main purpose is to allow developers to build chatbots that can comprehend natural language input and provide intelligent responses to users,[Bocklisch et al., 2017]. By using machine learning algorithms, Rasa can interpret user input and generate appropriate responses. It can also be integrated with other systems to provide a seamless conversational experience. Rasa is highly customizable and flexible, which enables developers to tailor their chatbots to specific domains and use cases. It provides a wide range of features and tools for training, testing, and deploying chatbots, as well as for monitoring performance and improving accuracy over time.

2.1.2 Main Features of Rasa

Rasa provides powerful tools for identifying and understanding natural languages (NLU). It contains built-in tools to extract intent and entity information from messages entered by the user, based on machine learning algorithms to recognize patterns in the text. Rasa also provides a system for control and dialogue management. It relies on machine learning to determine the appropriate steps to take in each conversation.

Rasa can be integrated with different communication channels such as Slack, Facebook Messenger, and Telegram . . . etc. It is highly customizable and flexible, allowing developers to tailor the system to their specific use cases and requirements. Perhaps among the most important features, especially for developers, is that Rasa is an open-source framework, which means that anyone can know everything about the structure and development of Rasa and how it works, in addition to everyone being able to use it and contribute to its development.

Finally, Rasa has a large and active community of developers and users, which means that information and support regarding problems that may occur are widely available.

2.2 Components of Rasa

The Rasa open-source framework offers two main components, namely, Natural Language Understanding (NLU) and Dialogue Management (Rasa Core):

2.2.1 Natural Language Understanding (NLU)

NLU is responsible for understanding and extracting information from user messages. It is a machine learning-based method that uses various algorithms to extract intent and entity information from natural language text. The following are its basic elements:

1. *Intent Detection*: One of the main tasks of Rasa NLU is to define the goal and purpose of the message that the user submits. The intention is the goal or purpose of the user in sending the message. For example, if the user sends the message “I am looking for a hotel in Ghardaïa state”, the message intends to search for a hotel. This is known through the machine learning algorithms used by Rasa NLU and this will be detailed in this section.
2. *Entity Recognition*: Rasa NLU also identifies the entities from the user’s message, and the entities are specific pieces of information relevant to the message’s purpose. For example, when the user enters "look for a great hotel at a cheap price," here the entities are "hotel" and "price." Entities are extracted depending on different built in algorithms such as ‘*RegexEntityExtractor*’, *CRFEntityExtractor* or *DIETClassifier*
3. *Preprocessing*: Rasa NLU preprocesses user messages to remove noise and irrelevant information before recognizing intents and entities. This includes removing stop words, punctuation marks, and other irrelevant text.
4. *Customizable*: Rasa NLU is highly customizable, allowing developers to train and tune algorithms according to their specific use cases and requirements. This means that it can be customized to handle domain-specific entities and intents.
5. *Machine Learning*: Rasa NLU uses machine learning algorithms like SVM, CRF, and LSTM to recognize intents and entities. This means it can learn from user interactions and improve over time to provide better answers and increase user engagement.
6. *Multilingual support*: Rasa NLU supports multiple languages including English, Spanish, German, Arabic, etc.

Each of the aforementioned elements can be included under the name NLU Pipeline.

NLU Pipeline: In the context of NLU, a pipeline refers to the series of steps involved in processing user messages to extract intent and entity information. NLU pipelines consist of a set of preconfigured steps or components, that work together to extract the meaning from natural language text. These components are responsible for tasks like tokenization, entity recognition, intent classification, etc.

Each component in the NLU pipeline is designed to perform a specific task, and the order of the components in the pipeline determines how user messages are processed. For example, a pipeline could start with a component that preprocesses the user’s messages by removing stop words and punctuation. This could be followed by components that perform entity recognition, and then components that perform intent classification.

There are different types of components that we find in the pipeline and they are as follows:

- **Tokenizers:** The first step is to divide the text into smaller text parts that we know as tokens Figure (2.1). The division process takes place based on some rules that can be created, such as division based on a “space” or based on “punctuation marks”.

To do this process, usually we use the *WhiteSpaceTokenizer* header for the English language, but for the rest of the languages, the *Spacy* library is a great option.



Figure 2.1: Tokenisation

For the Arabic language, it is possible to use the *ArabicStemmer* and *RegexTokenizer* provided by the *nltk* library.

- **Featurizers:** In Rasa, features are input to machine learning models for natural language understanding and dialog management. Features can be divided into two categories: sparse features and dense features.
 - **Sparse features** are binary features that indicate the presence or absence of a particular word or phrase in the input text. Sparse features are represented as a sparse matrix, each row represents a training sample, and each column represents a feature¹. For example, in a binary bag-of-words representation, each column corresponds to a unique word in the vocabulary, and each element in the matrix has the value 1 if the word exists in the input text, and 0 otherwise.

¹<https://rasa.com/blog/train-on-larger-datasets-using-less-memory-with-sparse-features/>

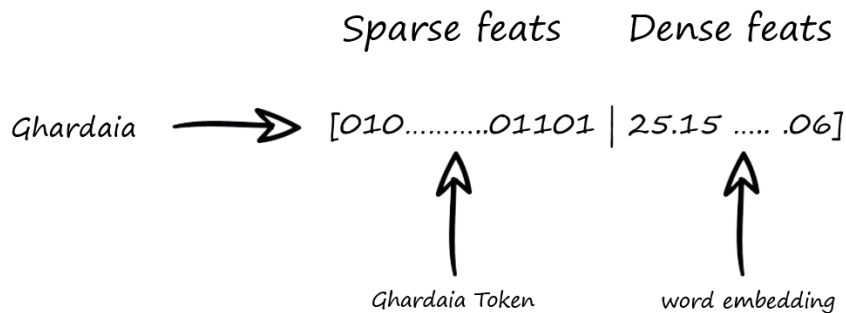


Figure 2.2: Featurizers

- *example*: let's say we have the following two customer reviews:

Review 1: "This product is amazing, I love it!"

Review 2: "I'm very disappointed with this product, it doesn't work"

We can represent each review as a bag-of-words vector using a vocabulary of four words: "product", "amazing", "disappointed", and "work". The resulting sparse feature vectors would look like this:

Review 1: [1, 1, 0, 0] # "product" and "amazing" are present

Review 2: [1, 0, 1, 1] # "product", "disappointed", and "work" are present

Note that most of the elements in these vectors are zero, indicating that most of the words in the vocabulary are not present in the corresponding review. This sparsity is common in text classification tasks and can be handled efficiently using sparse matrix representations and algorithms designed to work with sparse data.

- **Dense features** are continuous numerical features that represent a finer-grained representation of the input text. Dense features are typically generated using neural network models such as word embeddings or sequence encoders to convert input text into a fixed-length vector representation. For example, pre-trained word embedding models can be used to generate dense features that capture the semantics of input text.

- *example*: Suppose we have a dataset of house prices, where each house is represented by several numerical features, such as its size, number of bedrooms, and number of bathrooms. If we have 10,000 houses in our dataset, then each house is represented as a 3-dimensional vector, with each element representing one of the numerical features, we have the following three houses:

House 1: size = 1200 sqft, bedrooms = 3, bathrooms = 2

House 2: size = 1500 sqft, bedrooms = 4, bathrooms = 2

House 3: size = 2000 sqft, bedrooms = 5, bathrooms = 3

We can represent each house as a dense feature vector with three elements, where each element represents one of the numerical features. The resulting dense feature vectors would look like this:

House 1: [1200, 3, 2]

House 2: [1500, 4, 2]

House 3: [2000, 5, 3]

Note that all of the elements in these vectors are non-zero, indicating that each house has a high number of non-zero features. This density is common in numerical prediction tasks and can be handled efficiently using matrix computations and algorithms designed to work with dense data.

Previously, we saw that features are created for words individually, and features are also created for the entire sentence, this is referred to as the CLS token.

- Intent Classifiers:** For intent classification, by default Rasa uses the *DIET* classifier, as it has the added benefit of being capable of entity extraction. The model is equipped to learn from token and sentence features for comprehensive understanding. Following the generation of features for every token and the sentence as a whole, the text can then be forwarded to the DIET classifier (Figure 2.3).

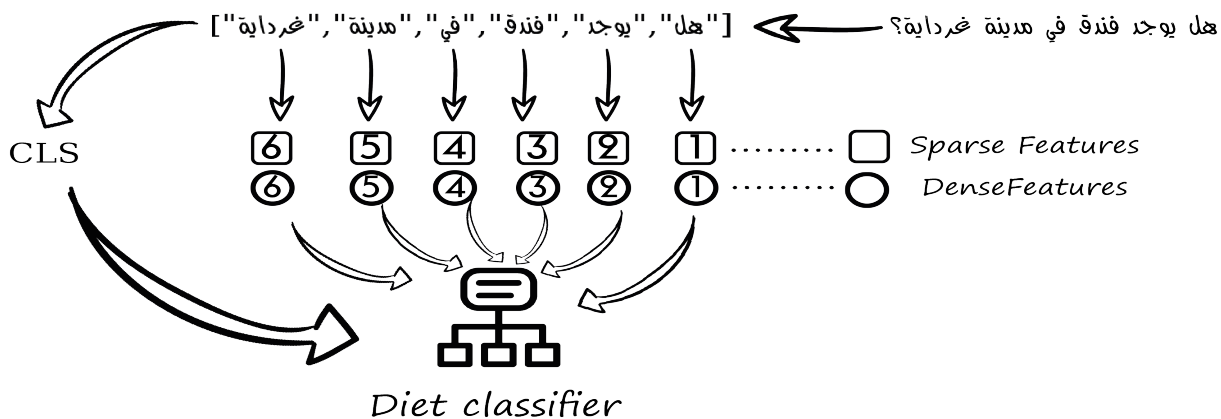


Figure 2.3: Diet classifier

Diet Classifier: In the past, the algorithms hosted by Rasa used to do one of two tasks, either detecting entities or classifying intent, but the matter is different with the DIET classifier, as it can do both.

DIET is a multi-task transformer architecture that handles both intent classification and entity recognition together. It provides the ability to plug and play various pre-trained embeddings

like BERT, GloVe, ConveRT, and so on (Figure 2.4)².

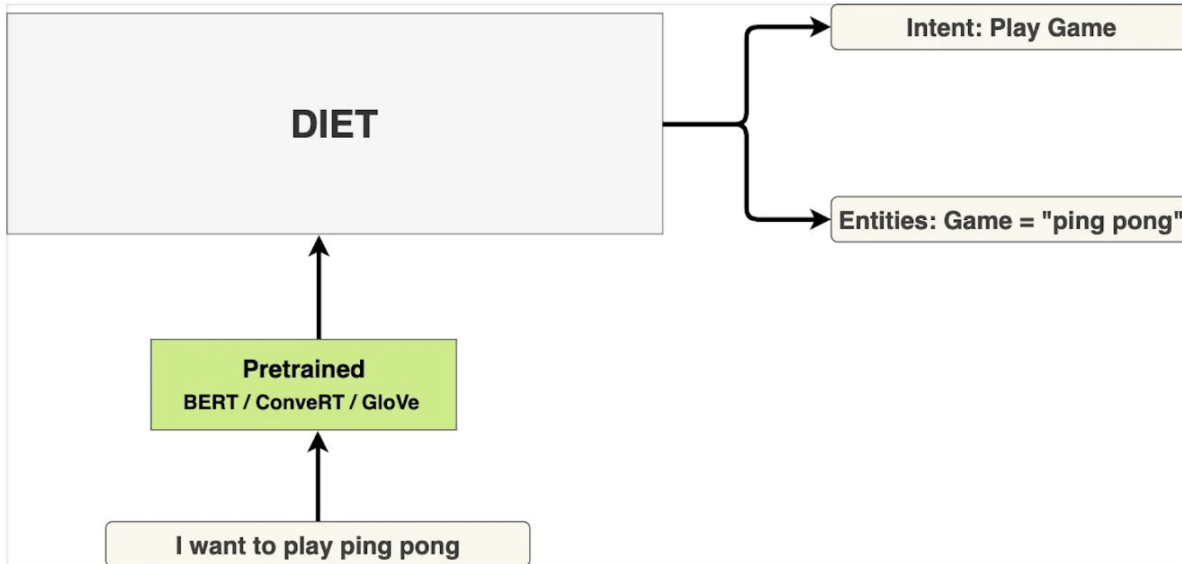


Figure 2.4: High-level illustration of DIET [Rasa, 2022]

- Entity Extraction:** As we mentioned earlier, despite the possibility of DIET in discovering and identifying entities, there are some types of entities that it is preferable not to use DIET in identifying, and among these types are entities that have a clear pattern and structure such as phone numbers or email that can be identified and discovered, for example, by `RegexEntityExtractor`. Hence, Rasa allows the use of multiple entity extractors because each one has its characteristics and advantages

2.2.2 Dialogue Management (Rasa Core)

In the previous item, we saw how we can define the intentions and entities of the user's message. This section is responsible for determining the appropriate response and the necessary steps, based on three policies that are present by default in Rasa. There are machine-learning and rule-based policies that can use in tandem. These policies are located in a file `config.yml`, this policies are:

- Rule-based Policies:**

²<https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance>

- (a) *Rule Policy*: Rule Policy is a dialog management policy used in Rasa Core, based on a set of predefined rules for selecting the next action based on the current dialog state and user input.

In a rule policy, rules are defined by developers and specify a set of conditions that must be met for a particular action to be selected. These conditions may be based on user intent, entities, conversation history, or other relevant information.

For example, a rule could specify that a chatbot should respond with a welcome message (*utter_greet*) when the user's intent is "*greeting*" and the conversation history does not contain previous greetings. Similarly, another rule could specify that when the user intent is "*book_flight*" and the required entity information is provided, the chatbot should start the flight booking process.

The listing 2.1 show an example of how rules are defined in Rasa:

Listing 2.1: Definition of rules in Rasa

```
rules.yml

rules:

- rule: rule name
  steps:
  - intent: intent X
  - action: utter_respond_X
```

2. Machine learning Policy:

- (a) *MemoizationPolicy*: MemoizationPolicy is a dialog management policy used in Rasa Core³ that caches previously selected actions and associated dialog states to improve chatbot responsiveness and reduce the need for expensive dialog state computations.

In the MemoizationPolicy, the policy first checks that the current dialog state and user input match the previously seen state. If there is a match, the operation associated with that state is returned without further computation being performed. If there is no match, the policy falls back to the next policy in the pipeline, such as a machine learning-based policy like the Transformer Embedding Dialogue (TED) (that we will discuss later) or a rule-based policy.

³<https://rasa.com/docs/rasa/reference/rasa/core/policies/memoization/>

The main benefit of MemoizationPolicy is the ability to quickly retrieve previously selected actions and associated dialog states without heavy computation. This can significantly improve the chatbot's response time, especially in the case of common questions or frequent conversation paths.

However, MemoizationPolicy is only effective if the chatbot frequently encounters similar conversation paths. If the conversation paths vary too much or are unusual, the MemoizationPolicy may not find a suitable state, and the policy falls back to the next policy in the pipeline, possibly resulting in slower response times.

- (b) *Transformer Embedding Dialogue (TED) Policy*: It is one of the policies that Rasa uses to choose what action the wizard should take next. It uses a transformer architecture to decide which conversations to pay attention to and which to selectively ignore when making predictions.

We know that Rasa uses many policies that have a role in predicting and choosing the appropriate action, as all existing policies provide predictions, and the priority of each policy differs from the other. The policy that has the highest degree of confidence is the one that is most likely to be predicted and is a strong TED policy in unfamiliar and new conversations, which are called **non-linear conversations**. Figure 2.5 shows an example of a non-linear conversation

- **How TED works?** During every exchange of dialogue, the TED strategy incorporates three components of data into its processing: the user's input, the previously anticipated system action, and any stored slot values in the assistant's memory.⁴ All of these elements are converted into features and combined before being input into the transformer model.

The self-attention mechanism plays a crucial role in this context. At each dialogue turn, the transformer dynamically evaluates and adjusts the importance of various segments within the dialogue history. This enables the TED policy to consider a user's input during one turn while disregarding it entirely during another [Vlasov et al., 2019], making the transformer an effective framework for handling dialogue histories.

Subsequently, the transformer's output undergoes a dense layer operation to obtain embeddings, which are numerical representations used to approximate the meaning of the text within the dialogue contexts and system actions. The embeddings are then compared, and the TED policy aims to maximize the similarity with the target label while minimizing similarities with incorrect ones. This technique draws inspiration from the Starspace algorithm.

- **StarSpace algorithm:** The StarSpace algorithm is a machine learning algorithm that is used for natural language processing tasks such as recommendation systems,

⁴<https://rasa.com/blog/unpacking-the-ted-policy-in-rasa-open-source/>



Figure 2.5: Example of a non-linear conversation

text classification, and entity recognition. It was developed by Facebook AI Research in 2018, [Wu et al., 2018]. The algorithm is based on the idea of representing words, phrases, and sentences as points in a high-dimensional space. This representation allows the algorithm to capture the semantic relationships between words and phrases, which is useful for many natural language processing tasks. One of the key features of the StarSpace algorithm is its ability to learn from both labeled and unlabeled data. This means that it can use large amounts of unstructured text data to improve its performance on specific tasks.

During the prediction of the next system action, a ranking is established for all potential system actions based on their similarity. The action exhibiting the highest similarity is then chosen as the selected system action.

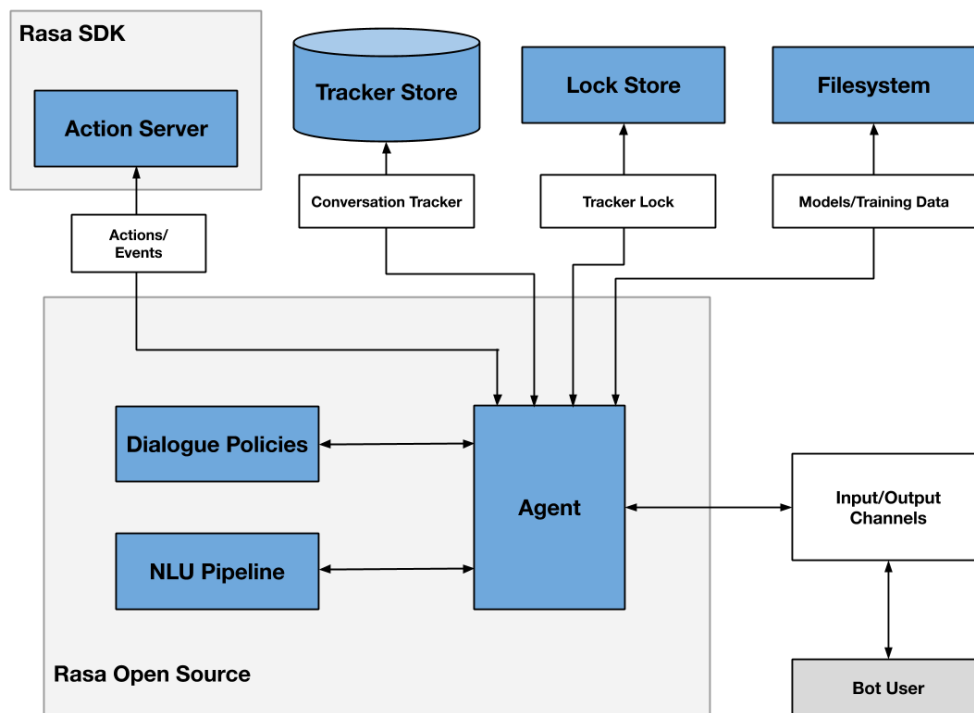


Figure 2.6: Architecture of Chatbots that is developed using Rasa [Rasa, 2023]

2.3 Rasa General Architecture

Figure 2.6 shows the general architecture of the Chatbots developed using Rasa. Rasa SDK is an action server. This takes care of creating the required custom actions based on the chatbot design and is optional. Tracker store is responsible for storing chatbot conversations. Bots store context, which helps personalize interactions. Rasa uses a ticket-locking mechanism to ensure that incoming messages are processed for a specific conversation ID and to lock conversations while messages are being actively processed. The file system consists of models and training data. The ML model and data list will be in separate files. Depending on the application, different ML models and datasets can be used to train chatbots.

2.4 Building Chatbots with Rasa

This section describes the steps involved in creating a chatbot with Rasa. We will examine how Rasa's fundamental components are connected and programmed.

2.4.1 Rasa Installation

Before installing upside down on the device, it is necessary to have Python installed first on the device. After installing Python, the framework is installed upside down by the command that is written on the command port:

```
1 pip3 install rasa
```

In this way, Rasa has been installed upside down on the device, and it is possible to start using it.

2.4.2 Creating Rasa Project, Intents, Entities, and Actions

1. **Creating rasa project:** The first project in Rasa is built by the following command:

```
1 rasa init
```

After executing the command, the project files will be created as shown in Figure 2.7.

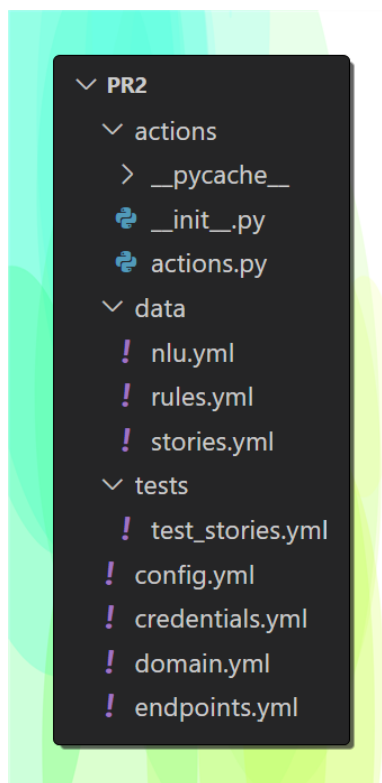


Figure 2.7: Project structure in Rasa

- (a) *Data folder*: In this folder, the data that the model will be trained on is placed, and we find three files in it. The first is *nlu.yml* used to define the NLU training data, the second is *rules.yml* used to define the rules that the chatbot should follow in certain scenarios, and the last one is *story.yml* used to define sample conversations between a user and the chatbot.
- (b) *tests*: Inside this folder, there exists a file *test_stories.yml* used to define a set of test stories that are used to evaluate the performance of the trained Rasa model. These stories are similar to the ones defined in the *stories.yml* file but are used specifically for testing purposes.
- (c) *config.yml*: is used to define the configuration of the various components that make up the Rasa pipeline. This file is used to define the various NLU and dialogue management components that are used to train the Rasa model which we referred to earlier, And the writing syntax in the config file is:

```
language: ar

pipeline:
- name: pipeline_name_1
- name: pipeline_name_2
...
- name: pipeline_name_n

policy:
- name: policy_name_1
  configuration ..
- name: policy_name_2
...
- name: policy_name_n
```

- (d) *credentials.yml*: is used to store the credentials for various external services that the Rasa bot may need to connect to, such as a messaging platform or a database. This file is used to securely store sensitive information such as API keys and access tokens and is typically excluded from version control to prevent unauthorized access.
- (e) *domain.yml*: is used to define the domain of the conversational AI assistant. It specifies the intents, entities, actions, templates, and slots that the bot can understand and respond to.
- (f) *endpoint.yml*: is used to configure the endpoints that the Rasa server will connect to.
- (g) *action.py*: In this file, custom actions are defined, the purpose of which is to create responses to user questions in a flexible manner and with additional properties.

2. **Creating intents:** The intent represents the user's intention and goal in the conversation. For instance, is he looking for a hotel or wants to book a flight, or otherwise? To create an intent in rasa, it can be added to the `nlu.md` file as follows:

```
\nlu.yml
- intent: intent_name
  examples: |
    - Training example 1
    - Training example 2
    ...
    ...
    - Training example n
```

Example:

```
\nlu.yml
- intent: order_food
  examples: |
    - I want to order a pizza
    - Can you deliver sushi to my address?
    - I'm hungry and looking for burgers

- intent: goodbye
  examples: |
    - bye
    - goodbye
    - see you around
    - see you later
```

3. **Creating entities:** An entity represents a specific piece of information that the chatbot needs to extract from the user's input. entities in Rasa have the following format: `[entity_name](entity value)`. The listing 2.2 present examples of creating entities.

Listing 2.2: Creating entities in Rasa

```
\nlu.yml
- intent: visit_city
  examples: |
    - I want to travel to
      the city of [Ghardaia](city)
      How can I get to [Ghardaia](city)?
    - What is the way to go to [Ghardaia](city)?
```

```

    - [Ghardaia](city) is the city I want to go to
- intent: my_name
  examples: |
    - my name is [Mohammed](name)
    - they call me [yacine](name)
    - I am [Ali](name)

```

To extract entities, more additional information must be specified for the intent training data. For example, when we train the model that the intent of the following sentence "My name is Muhammad" is that it falls within the username information, then extracting the entities is done by training the model on the location of the name in the sentence from through several examples. The model will be able to know the presence of the name, the city, the type of food or the destination on tourist trips, ...

4. **Creating Actions:** The procedure represents the response that the chatbot must take based on the user's input, and this is done in Rasa in two ways.

- (a) *Pre-built actions:* are built-in actions that come with Rasa and can be used without any additional configuration. Some examples of pre-built actions include *utter_greet* which returns a welcome message, and *utter_goodbye*. Pre-built actions can be used in the chatbot by defining them in the domain file, and they are executed automatically by Rasa when the corresponding intent is recognized.

In conclusion, the actions are responsible for executing a specific task based on user input, and the message the chatbot transmits to the user based on the results of these actions is known as **Responses**:

- i. **What are Responses?** Responses are messages that your assistant sends to the user⁵. A response is usually only text, but can also include content like images and buttons.
- ii. **Defining Responses:** In Rasa, responses are defined in the domain file using the *utter* prefix. The listing 2.3 explain the syntax and how defining a response , is defined in the domain file.

Listing 2.3: Creating responses in Rasa

```

responses :
  utter_response_name :
    - "Response message."

```

For example, you could add responses for greeting and saying goodbye under the response names *utter_greet* and *utter_bye* and this is what it explains listing 2.4

⁵<https://rasa.com/docs/rasa/responses/>

Listing 2.4: Examples of responses in Rasa

```
\domain.yml

responses:
  utter_greet:
    - text: "Hi there!"
  utter_bye:
    - text: "See you!"
    - text: "See you soon"
    - text: "Bye!"
```

Several things can be used inside responses, such as variables, conditional clauses, and many additions.

iii. Additions to responses:

A. *variable*: Here is an example of using variables 2.5:

Listing 2.5: Using variable in responses

```
\domain.yml
utter_greet:
  - text: "Hey, {name}. How are you?"
```

We point out that the *name* variable stores the name of the user that Rasa has recognized in previous conversations

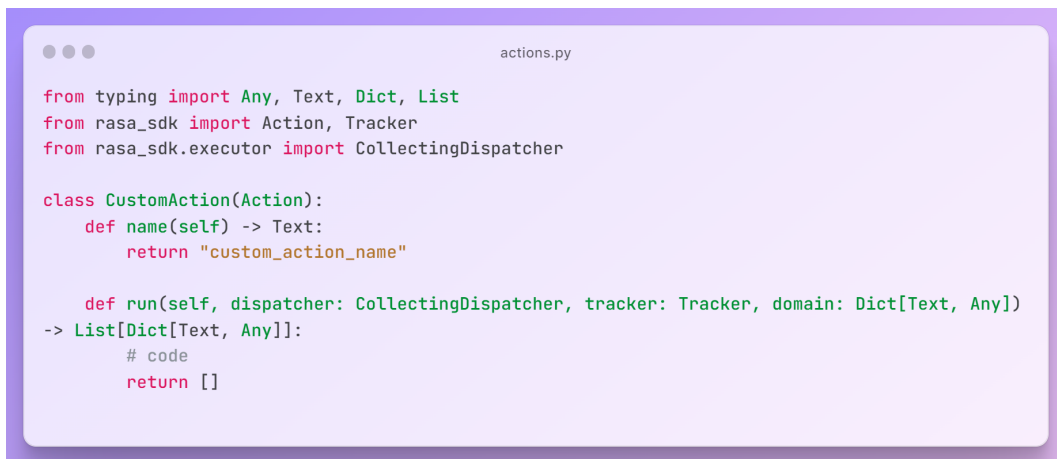
B. *Buttons*: are very important in the conversation process, as they save effort and time. Instead of typing, it is possible to directly specify the button that means what the user wants, the listing show how buttons can be included in responses .

Listing 2.6: Using buttons in responses

```
utter_greet:
  - text: "Hey! How are you?"
    buttons:
      - title: "great"
        payload: "/mood_great"
      - title: "super sad"
        payload: "/mood_sad"
```

Each button in its definition contains two keys, the first is "title", which represents the text that will appear on the button, and the second key, "payload", represents the message sent by the user when he pressed the button.

C. *Images*: It is a great feature provided by Rasa by providing the ability to reply with pictures, the listing 2.7 shows how to provide responses with pictures.



```

actions.py

from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher

class CustomAction(Action):
    def name(self) -> Text:
        return "custom_action_name"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any])
-> List[Dict[Text, Any]]:
    # code
    return []

```

Figure 2.8: Define Customer action

- **The name()** method in the custom action class defines the name of the action. This name is used to identify and call the action in the Rasa domain file and stories.
- **dispatcher:** Allow to send responses message to the user.
- tracker:** An instance that provides access to the conversation history and user inputs.
- **domain:** A dictionary that represents the Rasa domain, containing information about intents, entities, slots, etc.

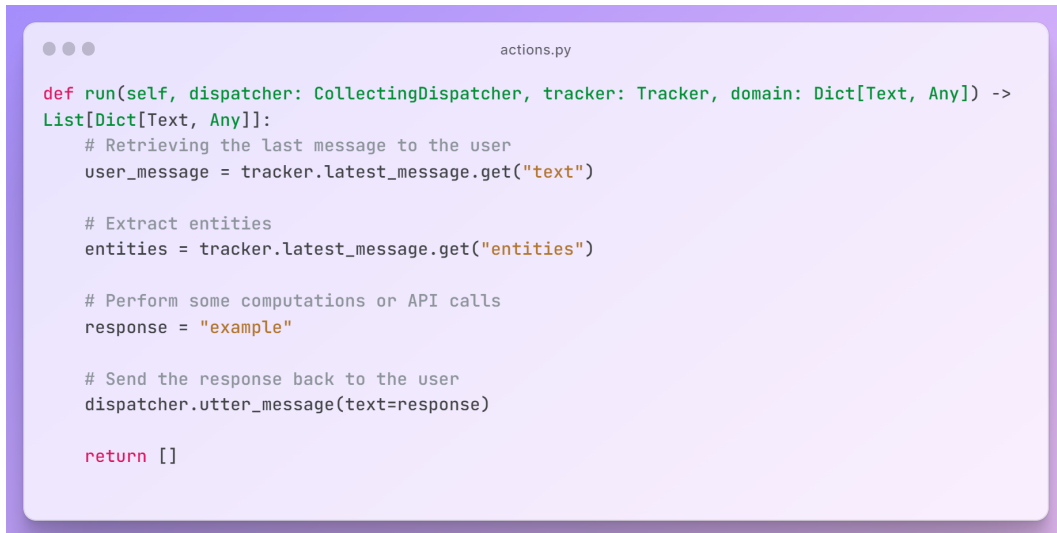
Listing 2.7: Using Images in Response

```

utter_ghardaia_pic:
- text: "This is a picture of Ghardaia."
  image: "https://ghardaia_test_image.png"

```

- (b) *Customer actions:* A custom action is a user-defined action that allows the writing of custom logic code in Python to handle specific tasks, such as querying a database, calling an external API, or performing calculations. All these operations that can be performed are intended to obtain information and return it to the user in response to his question. Custom actions are defined in the *actions.py* file in the Rasa project.
- i. **Defining a Custom Action:** To create a custom action in Rasa, first, we must define a custom action class. This class inherits from the Action class provided by Rasa. These steps are shown in Figure 2.8. In the run() method, we have the ability to retrieve the user's message, extract entities, access slots, and carry out any required computations or *API requests to generate a response*. We can also include the elements shown in Figure 2.9 inside the run method.
 - ii. This feature (Customer Actions) provided by Rasa can be exploited through the use

A screenshot of a code editor window titled 'actions.py'. The code defines a 'run' method that interacts with a Rasa dispatcher and tracker. It retrieves the latest message, extracts entities, performs a computation (returning 'example'), and sends the response back to the user.

```
def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
    # Retrieving the last message to the user
    user_message = tracker.latest_message.get("text")

    # Extract entities
    entities = tracker.latest_message.get("entities")

    # Perform some computations or API calls
    response = "example"

    # Send the response back to the user
    dispatcher.utter_message(text=response)

    return []
```

Figure 2.9: Customer Actions - Run method

of the GPT-3 model and its properties to provide answers to user questions. Figure 2.10 shows an example of that.

2.4.3 Rules and Stories

Stories and rules are two important elements in the process of building a chatbot, especially in the training phase, as they clarify and help the chatbot to provide appropriate responses in the flow of dialogue.

1. **Stories:** In Rasa, stories are used to define the dialogue flow. A story represents a sample conversation between a user and a chatbot. It helps the chatbot understand how to respond to the user's messages based on the context of the conversation. The Listing 2.8 show an example of how to define stories in Rasa:

Listing 2.8: Define stories in Rasa

```
\data\stories.yml

- story: visit ghardaia story
  steps:
  - intent: greet
  - action: utter_greet
  - intent: moode_great
```



```

actions.py

def Gpt(question,histo,data):
    file = open(datapath,encoding="utf8")
    csvreader = csv.reader(file)
    header = []
    header = next(csvreader)
    rows = []
    for row in csvreader:
        rows.append(row)
    model= 'text-davinci-003'
    prompt = f"Answer the user's question based on this data:
    Data : {data}
    previous conversations : {histo}
    user question ; {question}"
    data = {'model':model , 'prompt': prompt , 'temperature':0, 'max_tokens' :256, 'stop':["
    Human:", " AI:"]}
    response=requests.post(url, headers=headers, json=data,verify=False)
    Answer =response.json()['choices'][0]['text']
    return Answer

```

Figure 2.10: Using GPT-3 Model

- action: utter_how_can_i_help_you
- intent: ghardaia_hotel
- action: ghardaia_hotel_list
- intent: thanks
- action: utter_welcome1
- intent: goodbye
- action: utter_bye

- story: ghardaia_pic
 - steps:
 - intent: greet
 - action: utter_greet
 - intent: ghardaia_pic
 - action: utter_ghardaia_pic

As we noticed in an example about a conversation about visiting the state of Ghardaïa, this is how the stories are developed by which the model is trained.

2. **Rules:** Rules are a type of training data used to train the Assistant's dialog management model. Rules describe short parts of dialogue that should always follow the same process. Rules are useful for managing small, specific patterns of dialogue, but unlike stories, rules cannot travel down unfamiliar dialogue paths.

Before starting to use the rules, they must be defined in the `config.yml` file as follows in listing 2.9:

Listing 2.9: Define RulePolicy

```
policies :  
- ... # Other policies  
- name: RulePolicy
```

After adding a rule to the config file, it will be taken into account in the model training process and will play a role in making predictions. The following is an example of creating a rule that states that if the user message is classified as a welcome message, it must be answered with a welcome message:

```
rules :  
- rule: Say `hello` whenever the user sends a message  
  with intent `greet`  
  steps :  
  - intent: greet  
  - action: utter_greet
```

2.4.4 Training and Evaluating Chatbots

After adding and creating the training data (intents, story, rules) and appropriate responses (pre-built in and customer actions), the most important step is to train the model and evaluate its efficiency.

1. **Training:** First, the settings for training and the policies that will be used in the chatbot are determined, the chatbot's language, the number of training times, and so on. In general, this preparatory stage is before starting to train the model and setting up the environment.

These settings are included in the `config.yml` file as shown in Listing 2.10 :

Listing 2.10: Set the pipeline and policy for the Rasa project

```
config.yml
```

```
language: ar  
  
pipeline :  
- name: pipeline used 1
```

```

.....

policies:
- name: policy_used 1
.....

```

Once the settings have been adjusted, the model is trained with the command *rasa train*. After the training procedure is complete, the model appears in the model folder.

2. **Testing and evaluating the model:** After completing the model training process, the next step is to conduct the evaluation process and test the efficiency of the model

As we mentioned earlier, Rasa consists of two main components, Rasa NLU, and Rasa Core, each of these two components has its data to test on.

- (a) *Testing data for Rasa NLU:* In Rasa, the test data for a component is derived from the intent data. The intent data consists of a collection of examples that signify a particular intention. During the testing process, one or two examples are selected from each intention and consolidated into a separate file to create the test data. The test data comprises 20% of the total examples in the intentions.

In RASA, this is easily done by the *rasa data split nlu* command, where the data is automatically divided into 80% for training and 20% for testing in a separate file.

- (b) *Testing data for Rasa Core:* The testing data for Rasa Core differs from the testing data for Rasa NLU. As we know that the Rasa Core section is responsible for choosing the appropriate response based on the entity extracted from the user's question, and therefore the training data are stories, and from it, we conclude that the test data is also a phrase about stories.

Testing a trained model against test stories is the best way to gain confidence in our assistant's behavior in a given situation. Test stories are written in a modified story format and allow us to provide full dialogue and test whether our model behaves as expected given specific user input. This is especially important as we start to include more complex stories from user conversations.

Test stories are similar to the story's training data but with user messages added, the Listing 2.11 show how to define and create test stories

Listing 2.11: Define test stories

```

-----
tests/test_stories.yml
-----

```

```

stories:
- story: A basic story test
  steps:
  - user: |
      hello
    intent: greet
  - action: utter_ask_howcanhelp
  - user: |
      show me [chinese>{"entity": "cuisine"} restaurants
    intent: inform
  - action: utter_ask_location
  - user: |
      in [Paris>{"entity": "location"}
    intent: inform
  - action: utter_ask_price

```

The test process is done by the `rasa test` command. After those tests are performed for Rasa NLU and Rasa Core components, the test results are stored in several formats in a result folder that is created after the end of the test process.

3. Interpretation and Evaluation:

- (a) *Rasa NLU*: After completing the testing process, Rasa generates a report (`intent_report.json`), confusion matrix (`intent_confusion_matrix.png`), and confidence histogram (`intent_histogram.png`) for our intent classification model (Rasa NLU).

The report records the precision, recall, and F1 score for each intent and provides an overall average. A confusion matrix shows which intents are confused with other intents. All incorrectly predicted samples are logged and stored in a file called `errors.json` for easy troubleshooting. An Histogram allows to visualize the confidence of all predictions with correct and incorrect predictions represented by blue and red bars, respectively. As you increase the quality of the training data, the blue histogram bars move up the graph and the red histogram bars move down the graph. It should also help reduce the number of red histogram bars themselves.

- (b) *Rasa Core*: Rasa generates a report (`response_selection_report.json`), confusion matrix (`response_selection_confusion_matrix.png`), confidence histogram (`response_selection_histogram`) and errors (`response_selection_errors.json`). If the pipeline includes multiple response selectors, they are evaluated in a single report.

The report logs precision, recall, and f1 measure for each sub-intent of a retrieval intent and provides an overall average

2.5 Deploying a Rasa Assistant

After completing the previous steps and preparing the chatbot, the next step is to publish it and make it available for testing on various platforms. There are many ways that Rasa provides to make the chatbot public in different social media platforms, but we will limit ourselves to one platform, we picked telegram for this purpose.

2.5.1 Telegram Application

Telegram is a messaging application, that is favored for its advanced privacy and encryption features⁶, along with its robust support for extensive group chats. Notably, unlike Facebook Messenger and WhatsApp, which are both owned by Facebook, Telegram maintains independence from other social media platforms, making it an attractive choice for many users.

Telegram boasts a multiplatform nature, it's available in various operating systems such as iOS, Android, Windows, Mac, and Linux. Additionally, users have the convenience of accessing Telegram through a web browser, further expanding its accessibility.

BotFather: The term pertains to the authorized bot furnished by Telegram, serving as a pivotal tool for generating and administrating additional bots within the platform. It operates as a platform for bot development, empowering users to establish, customize, and govern their personal Telegram bots.

BotFather presents a straightforward and intuitive interface for the generation of fresh bots. It assigns an exclusive API token to each bot, serving as the bot's validation and access code for engaging with the Telegram Bot API. This token allows developers to instruct their bots to dispatch and receive messages, react to commands, and execute diverse tasks within the Telegram ecosystem.

2.5.2 Deploy Rasa Assistant in Telegram

1. The first step to do is to search for botFather in Telegram and start a conversation with it.
2. Follow the steps and instructions provided by BotFather to create a bot, and after completing the bot creation process, it will generate the API token, which will be the main key for the linking process.
3. On the other hand, in the Rasa project, specifically in the credentials.yml file, the following code is placed as shown in Listing 2.12.

⁶<https://www.businessinsider.com/guides/tech/what-is-telegram>

Listing 2.12: settings for linking the Rasa project with Telegram

```
credentials.yml
```

```
telegram:
  access_token: "Telegram acces token"
  verify: "verification token"
  webhook_url: "https://our_domain/webhooks/
telegram/webhook"
```

It is possible to rely on some services as an intermediary to establish a communication channel between Rasa and Telegram, such as ngrok.

Ngrok: When integrating Telegram with a Rasa project, Ngrok proves to be a valuable asset by facilitating a secure tunnel that connects the local Rasa server to the internet. It enables incoming requests from Telegram's API to reach the Rasa server, even if it resides behind a firewall or on a private network.

The main role of Ngrok in this scenario is to establish a secure link between Telegram and the local Rasa server. It accomplishes this by generating a public URL that redirects incoming requests to the local server. As a result, there is no requirement to deploy the Rasa server on a publicly accessible domain.

4. the listing below show how to start the Rasa server :

```
rasa run -m models --enable-api --cors "*" --debug
```

or this :

```
rasa run
```


3

State of the art

3.1 Introduction

In recent years, chatbots have evolved into powerful tools in the field of human-computer interaction, changing the way we interact with technology and improving every aspect of our lives. These intelligent conversational agents have made remarkable progress in their development and capabilities, bringing us closer to a world where human-machine interaction is not only possible but seamless.

The purpose of this chapter is to provide an overview of the state-of-the-art in chatbot development and examine the development of these systems and the tasks they perform. We will examine the historical evolution of chatbots and highlight the key milestones and breakthroughs that have shaped their current landscape. Additionally, we will give special attention to the development of Arabic chatbots, emphasizing efforts to meet the unique linguistic and cultural needs of Arabic-speaking users.

3.2 Rule Based Chatbots

In 1966 [Weizenbaum, 1966] designed the first chatbot at the University of Massachusetts named ELIZA. It relied on pattern-matching techniques to allow humans to interact with computers using natural language. This chatbot is written in a list-processing computer programming language named MAD-Slip (Michigan Algorithm Decoder - Symmetric List Processor) and implemented in Multiplexed Information and Computing Service(MAC) time-sharing system that allows users to share a computer from different locations. It applies the transformation rule (decompositional rule and reassembly rule) to decompose the inputs into segments and then arrange the segments to create a sentence based on their importance. ELIZA doesn't have true context understanding and its responses rely on user engagement and pattern matching and it cannot retain long conversations. This program was a source of inspiration as it had resonance and admiration from the scientists in the field of artificial intelligence.

In the 1970s [Colby et al., 1971] with support from the National Institute of Mental Health and the Secretary of Defense of the U.S. developed PARRY a chatbot to simulate a psychiatric interview of paranoid linguistic behavior. Its task was to intercept the input and answer in paranoid mode, the program is written in MLISP a programming language that translates human syntax (M-expressions) to symbolic expressions which is a syntax structure in LISP, to evaluate the program the developers collaborated with psychologists to classify whether the program is paranoid or not. Then they collaborated with Robert P. [Colby et al., 1972] to test if the program succeeded to simulate a conversation with a patient. The results show the success of this chatbot to respond similarly to those from a paranoid human.

Another chatbot appeared named ALICE(Artificial Linguistic Internet Computer Entity) devel-

oped by [Wallace, 1995] in the 90s using the AIML which was introduced in the same paper. The basic unit of knowledge in AIML is categories (atomic categories, default categories, and recursive categories) each category consists of a pattern(the user’s input) and a template(the response generated by the chatbot). the developers enabled inputting new dialogue patterns into it with ALICE Free Software Technology, there are over 50,000 hand-coded categories in ALICE public domain as it allows. ALICE is capable to understand the inputs based on predefined responses and uses pattern-matching techniques with an AIML interpreter to respond. Graph master (a set of files and directories which has nodes representing the first word) is an important component in the pattern-matching process. With the use of the depth-first search technique to navigate the graph master, the pattern matching algorithm tries to find the best longest pattern match. After receiving an input AIML stores it as a category, which consists of a response template, with the decision tree the model preprocesses the response template and matches it to its nodes. when it matches the user’s input it executes an action or response. AIML uses recursive techniques to repeat the utterance of the input. This technique does not always reply with a meaningful response. This chatbot’s performance was tested in the “Loebner Prize in Artificial Intelligence Contest” which is an annual competition to evaluate chatbots and their capability to simulate humans (turing test) it won this prize multiple times.

3.3 Generative Chatbots

Pattern matching rules can be very specific and fragile, that’s why chatbots are so limited, so a new type has emerged, namely generative chatbots which depend mainly on artificial intelligence and machine learning. There are two types of generative chatbots(text and speech chatbots), defining rules to manually match patterns is no longer important because the chatbot is trained on a set of data, and using machine learning algorithms, patterns, and rules are known flexibly.

3.3.1 Generative Text Chatbots

After decades and the development that took place in the field of natural language processing, generative text chatbots appeared, we see today more advanced and sophisticated such as ChatGPT and BingAI. ChatGPT is a chatbot released in 2022 by [OpenAI, 2022] an AI research company. it was trained on a large amount of textual data, including web pages and documents. GPT models which are the basis of this chatbot is NLP models that use deep learning techniques known as transformers technology to generate texts. The first GPT model(GPT1) was introduced in 2018 it had 117 million parameters from the internet to train, GPT2 the second model got released to the public in February 2019 after being trained on a billion and a half parameters. GPT3 which is the model ChatGPT started with was released in 2020 after training it on 175 billion parameters. In

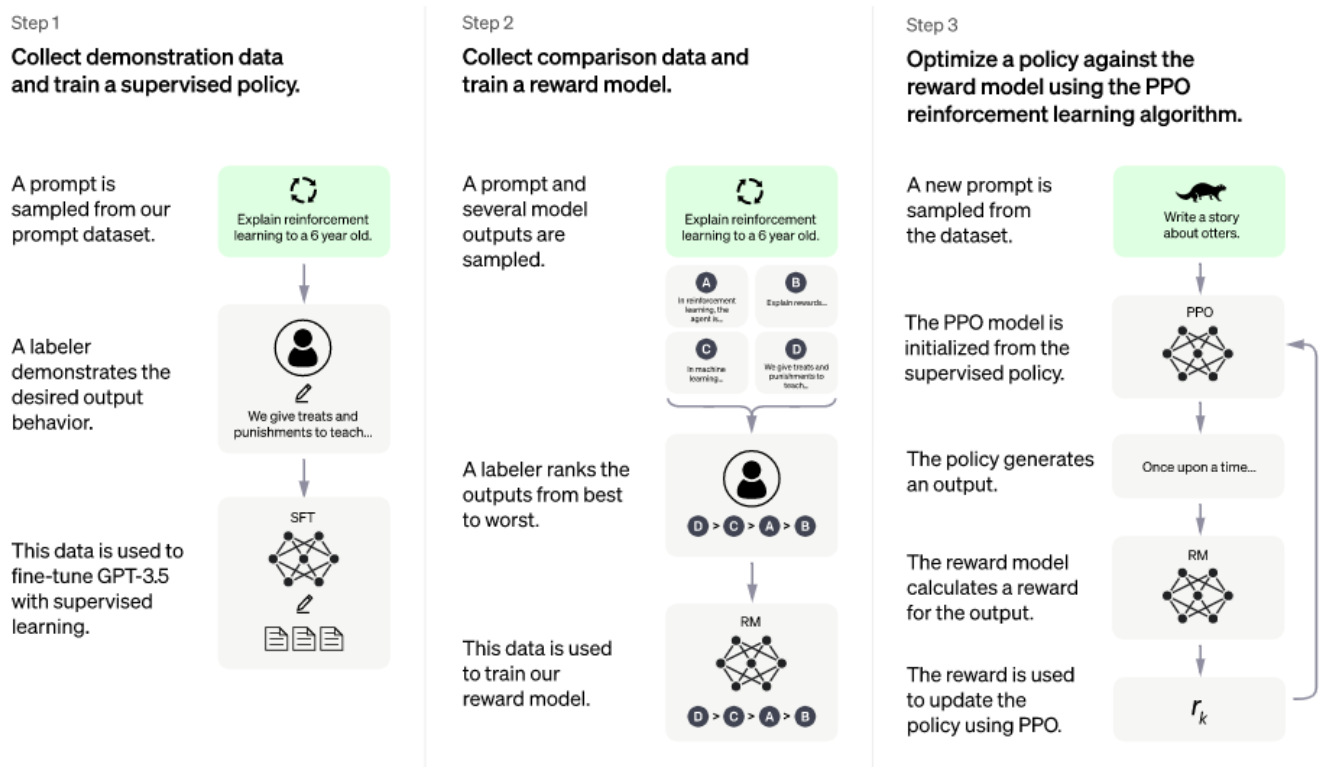


Figure 3.1: Steps of training ChatGPT [OpenAI, 2022]

March 2023 OpenAI launched GPT4 and made it available to users who subscribe to CHATGPT+ without providing any information about the data it was trained on [Leiter et al., 2023]. To create this chatbot, its model(GPT3.5 currently) was trained using Reinforcement Learning from Human Feedback (RLHF). To train the initial model developers used supervised fine-tuning where human AI trainers provided conversations as they were users and AI assistants at the same time, they mixed those conversations with the InstructGPT dataset(over one million textual prompts to train GPT models). To fine-tune this model using proximal policy optimization, they created a reward model after collecting comparison data from the AI trainer's conversations (figure). This chatbot is sensitive to the inputs and sometimes, it would give different responses for the same input, also it sometimes guesses the user's intent instead of asking for more information.

After the attention, ChatGPT gained more generative models and chatbots appeared. *Bing AI* is a generative chatbot released on February 2023 by [Microsoft, 2023] after the partnership with OpenAI, it was the first online chatbot that used GPT4. It was developed to provide users with a more personalized search experience. With the access to web and the data GPT4 was trained on, this chatbot provides more updated answers than ChatGPT, it also includes references when the answers are from the web so the user can learn more. This chatbot also provides fast access to other online websites such as OpenTable and TripAdvisor to make more relevant answers. Before the release of this chatbot, it was tested by a set of people to get feedback to learn, 71% of the answers were rated positive. Bing AI has some limitations for example it is not capable to understand long queries. it also provides a limited number of queries(50 sessions per day), but it gets repetitive and unhelpful after 15 queries on the same subject.

3.3.2 Generative Speech Chatbots

The other type of generative chatbot is the Speech chatbot which appeared in 2011 through Apple Inc as a speaking conversational system named *Siri*. It was developed by the SRI (International Artificial Intelligence Center). Its speech recognition engine was provided by Nuance Communications. It was launched as one of the distinctive features of the new phone iPhone 4S then it was included in the rest of Apple's devices. It works by analyzing the user's spoken input through voice recognition technology and interpreting it using NLP. Siri performs many tasks on Apple devices such as sending messages, making calls, and setting alarms. Its connection to the internet also allows it to answer various questions such as weather conditions. Siri had different reviews from its users, some part of them criticized its misinterpretations, lack of flexibility, and not understanding of certain English accents. But, it was praised for the contextual understanding and the speech conversation. It was a qualitative leap in the field of chatbots and artificial intelligence in general. As it is considered a basic benchmark for competing companies where led to the emergence of many voice assistants.

The attention Siri captured was attractive for the big companies to build their voice assistant.

Siri had many competitors one of them being Alexa. it's a voice assistant developed by *Amazon* it was launched in 2013 in a smart speaker named Amazon Echo. it supports more languages than siri(English, German, French, Italian, Spanish, Portuguese, Japanese, and Hindi). its speech technology is largely based on Ivona, a Polish speech synthesizer. Alexa can perform several tasks such as making calls, setting alarms, creating to-do lists, getting information about the weather, get information through access to online articles and web pages. and several restaurants used it for take-out food, people could use Alexa to make orders. Users are allowed to add new skills to their Alexa through the Alexa Skills Kit, in 2016 Alexa had 1000 skills and it became 90,000 functions in 2019.

In 2014 Microsoft introduced its voice assistant named Cortana. Its development started in 2009 by Zig Serfian and the Microsoft speech products team. It uses the results it gets from Microsoft Bing to answer the user's questions and perform tasks. It is available in several languages such as English, French, and Portuguese. In 2020 Cortana was getting decreasing focus, the CEO of Microsoft stated that he no longer thinks that Cortana would compete with SIRI and ALEXA. it was then removed from Microsoft products such as XBOX in July 2020, and Microsoft also reduced its existence in Windows 11.

3.4 Hybrid Chatbots

Another type of chatbot appeared named hybrid chatbot that combines both generative and rule-based systems.

One of the early hybrid chatbots is *Jabberwocky*, which is developed by [Carpenter, 1997] in 1997 and launched on the internet. it simulates human conversation entertainingly and interestingly, then stores them to collect large conversational data. It succeeded in its goal to pass the Turing test with AI after winning Loebner Prize twice (2005, 2006). It was the first chatbot that was capable to interact with voice inputs. this chatbot relied on the dynamic database, it collects from the conversations tokens and keywords and then adds them to the bank of conversations which is stored in the memory. To generate an answer it uses contextual pattern matching(AI technique) to generate responses. first jabberwocky searches and identifies keywords and similar conversations in the bank. it uses the search result to learn the human manner to respond to that input, then use it to generate its answer. To limit the results to conversations with similar keywords. With every input, the program conducts another research process and adds the user's input to the available statement bank in real time. its new version came in 2008 with a new name, '*Cleverbot*'. In 2010, it had nearly 42 million statements in its conversational bank.

SmarterChild is a chatbot that was released in june 2001 from *ActiveBuddy* . it used AI algorithms to detect keywords in the inquiries and offer predetermined responses based on them. and used the AIM platform to provide news and information in real-time (stock information, weather status,

news) as it could perform as a translator or calculator with its add-on functionality. This chatbot was programmed on PERL a family of dynamic programming languages. It was the first chatbot to perform as a virtual assistant. It also was one of the earliest chatbots that went public in messaging applications (MSN,...).

Mitsuku is a chatbot created by [Worswick, 2010] using AIML. It is designed for general types of conversations based on rules written in AIML and integrates with social media platforms like Twitter, Telegram, etc. The Mitsuku bot uses heuristic-based NLP, hosted on Pandorabot. The bot module abstracts most of the work needed to create a robust chatbot system. To integrate this module, some AIML classes need to be in. One of its hallmarks is the ability to reason about specific objects. For example, if someone says "Can you eat a house?" Mitsuku will look at the properties of "house", find that the value of "made_from" is set to "brick" and answer "no" because it is not a house to eat. Mitsuku is a multilingual robot that uses supervised machine learning. The data is sent to human curators for validation when it learns something new. The application can only further incorporate and use verified data. However, Mitsuku is ineffective without a large amount of training data and does not provide a dialog management component.

In 2018 a chatbot for healthcare service was introduced by [Chung and Park, 2019]. The chatbot's role is to provide fast responses and experienced assistance to handle accidents that may occur in everyday life and also respond to changes in the conditions of patients with chronic diseases. The framework was comprised of four levels, the first one is the data level which involves collecting data about the user's lifestyle, health condition, and medical history. The data is collected in real-time through sensors that capture different information (emotional status, physical activity,...) and user input. The second one is the information level, which involves processing the collected data. The third is the knowledge level, which consists of analyzing the information and generating knowledge about the user's health using AI-based algorithms. The last one is the service level which is making the chatbot accessible to users through the chatting interface. This chatbot has shown the capability to help patients and especially through emergency cases, users could send images of their injuries or affected areas and receive emergency treatment information, this chatbot was integrated with existing health systems such as the health information system(HIS).

In the tourism field chatbots appeared for different services like hotel reservations, plane ticket reservations, and travel recommendations... In "Chatbot-based Tourist Recommendations Using Model-based Reasoning", [Nica et al., 2018] developed a chatbot for hotel recommendation. Users express their preferences and need in a conversation with the chatbot, with adding other features such as the price the chatbot recommend the most suitable hotel. To develop this chatbot, the developers designed an algorithm close to ConDiag (Constraint-Based Diagnosis) to help tourists make decisions about the hotel that suits their criteria. They also used Shannon's information entropy to solve the recommendation problem. When the user inputs his requirements, the algorithm generates a constraint model. Then it checks the consistency of the model by calling a constraint solver, if the

model is consistent the algorithm generates recommendations based on the user's requirements only. In cases where there are too many suggestions or the user's preference is too restricted (the model is inconsistent), the algorithm generates responses based on model-based reasoning after enhancing the user experience during the chat.

In "The Application of AGNES Algorithm to Optimize Knowledge Base for Tourism Chatbot" paper, [Sano et al., 2018] introduced a chatbot for tourists assisting and places recommendations in Indonesia (Malang city, Malang regency, and Batu city) using AGNES. The main goal to achieve is to optimize the knowledge base of the chatbot, to assist tourists in getting information, on which possible sites they have to visit optimally if they are under limited vacancy-time constraints. AGNES stands for Agglomerated Nesting, which is a hierarchical clustering algorithm, it is used in the program to cluster tourism sites. The algorithm first puts each tourism site as a point, then calculates the distance between the points, then it merges the two closest clusters into a new cluster, then recalculates the distance between clusters until only one cluster is left. After that, the result clusters of each step are fed into the chatbot's knowledge base (Dialogflow environment in this chatbot), to recommend to tourists the maximum possible sites to visit in the time that tourists have.

3.5 Arabic Chatbots

Despite the complexity of the Arabic language and the scarcity of research available on it, this did not prevent the emergence of specialized chatbots that support the Arabic language in various fields.

Among the first Arabic chatbots, The authors of [Shawar and Atwell, 2004] propose an Arabic chatbot based on machine learning techniques. To generalize ALICE, in recent versions [AbuShawar and Atwe] they developed a Java program to automatically convert the corpus text (including Arabic text) to the chatbot language model format. As the Qur'an provides religious and other questions with guidance and answers, they adapt it as a training corpus for their chatbot. The developed chatbot ensures a conversation in which the input is in Arabic and the response is the appropriate Qur'anic ayyas. It can be used as a search instrument for ayyas with identical words but distinct connotations.

"Ibn Sina" is a multilingual robotic speaker that was developed in 2011, as it supports interaction between users, whether by voice or text. Either the robot's answers are by voice only, and the robot's responses depend on accessing Wikipedia databases and also the Holy Quran database, and Google translate, so it includes various aspects and many topics, It also informs the user if there is any missing information or erroneous spelling. Pattern matching approach was used to develop this conversational robot, figure 3.2 shows the modular subsystem of Ibn-Sina.

"ArabChat" is a chat agent dedicated to helping students at the Applied Science University of Jordan under the name "ArabChat". It was built and developed in 2014 [Hijjawi et al., 2014] and

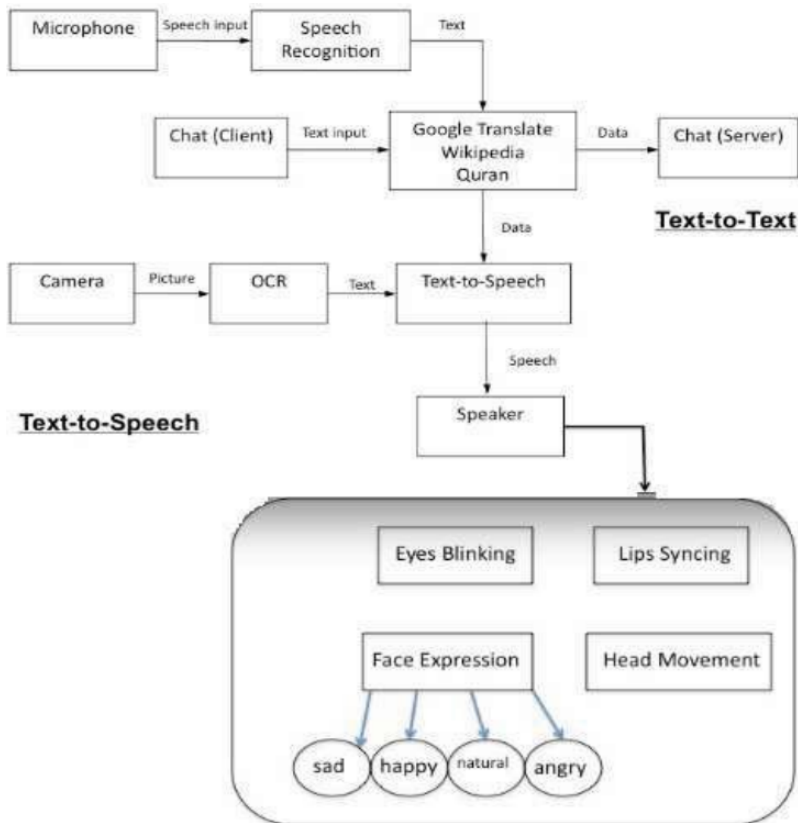


Figure 3.2: Modular subsystem of Ibn-sina

work on a pattern-matching approach to deal with text conversations in the Arabic language, as it consists of three main units that form the core of the robotic speaker and It is the programming language that was used in its construction and the engine that plays the role of responding to user conversations and providing appropriate answers based on the context in which the user speaks. As for the brain, it is the one that stores and contains context information.

In 2015, Mobile ArabChat was established by making some modifications to ArabChat and including it within the mobile applications to deal with Arabic conversations between Arab users. It was included in the Android system to work as a counselor for students at the University of Applied Sciences in Jordan, as we mentioned earlier.

The development did not depend on its inclusion in the Android system only, but it was restructured and comprehensively updated to obtain the best performance of the agent, and a new version was created in 2016, where it added new features of "speech classification" and "mixed rule", speech classification adds more words Basic to a question-based grammar pattern to distinguish between nonsensical speech and a question. A mixed rule focuses on how to handle requests with multiple

subjects that require several rules to be executed simultaneously, figure 3.3 shows more information about the Framework of the Enhanced ArabChat.

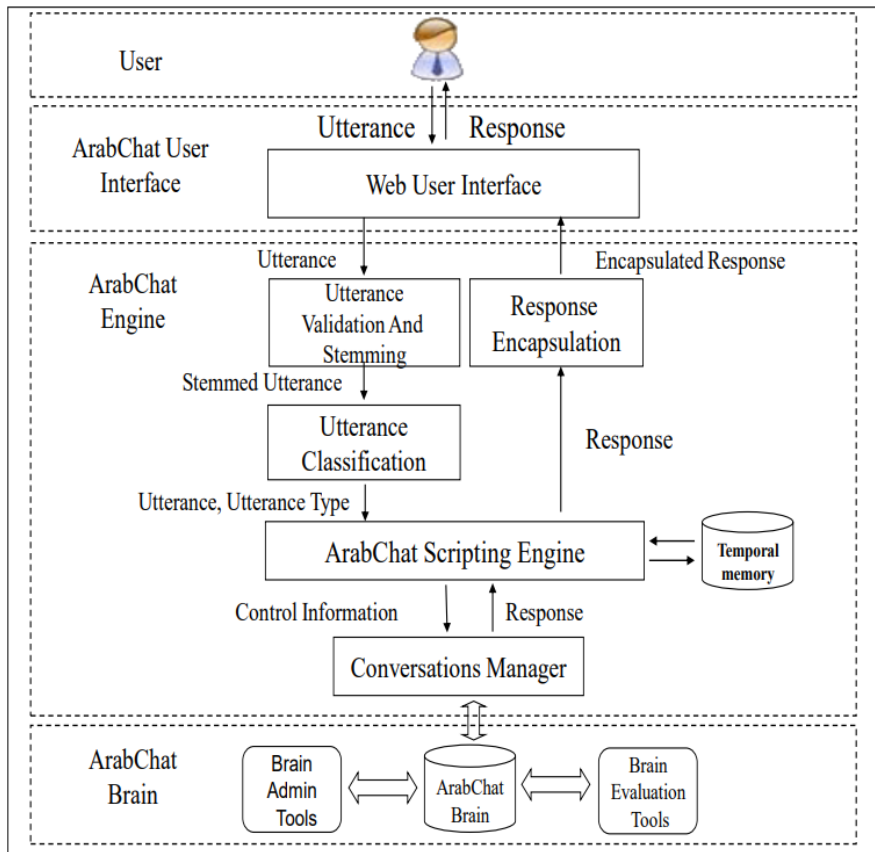


Figure 3.3: The Framework of the Enhanced ArabChat

Implemented by [Ali and Habash, 2016] in 2016, "BOTTA", an Arabic dialect chatbot, serves as a personable female companion who converses with users using the popular Egyptian Arabic (Cairene) dialect. It retains basic user information, specifically age, gender, and nationality temporarily by actively questioning users. BOTTA’s capability to converse about a wide variety of topics makes it an open dialogue between the user and the chatbot. The chatbot stores the categories under which it responds to user input in AIML files found within its knowledge base. Through text, users interact with it using a retrieval-based model that leans on a bank of predetermined responses and utilizes heuristics to produce a satisfactory reply, We note that BOTTA was developed by using AIML, Figure 3.4 shows an example of a conversation between a user and a BOTTA.

Ollobot is an Arabic chat agent whose mission is to help people provide a program and a healthy diet and manage their physical activities that was introduced in 2019 [Fadhil et al., 2019]. It also



Figure 3.4: A sample conversation between a user (U) and BOTTA (B).

tracks the user's daily nutritional intake, provides useful tips for a healthy lifestyle, and keeps a record of his eating habits, which was created based on the IBM Watson platform to take advantage of the advantages and Artificial intelligence technologies provided by the platform, and it has been linked with Telegram, the general architecture of OlloBot is shown in figure 3.5

More recently, in 2021, AL-Hanouf [Al-Ajmi and Al-Twairesh, 2021] created an Arabic chatbot for flights, as it applies information about available flights and the possibility of booking flights. It was created by the wit. ia platform and linked to the Telegram platform, Due to a lack of permission to connect this DS to real APIs, simulated flight booking systems were utilized. Enhancements to the system's functionality are underway with the help of user inquiries to form new training samples, the general structure of Flight Booking DS is described in figure 3.6

Rahal a chatbot that helps tourists arriving in the Kingdom of Saudi Arabia by providing a quick response to various questions and inquiries related to the field of tourism in the Kingdom of Saudi Arabia introduced in 2022 [Alhumoud et al., 2022], where three models of this chatbot were created and published, the first in the name of Saudi Rahhal Bot, which is intended for conversations in Arabic, it was created using the Java language, and the second model was programmed with IBM Watson, and the third, EngRahhalbot, which conducts conversations in English and is intended for

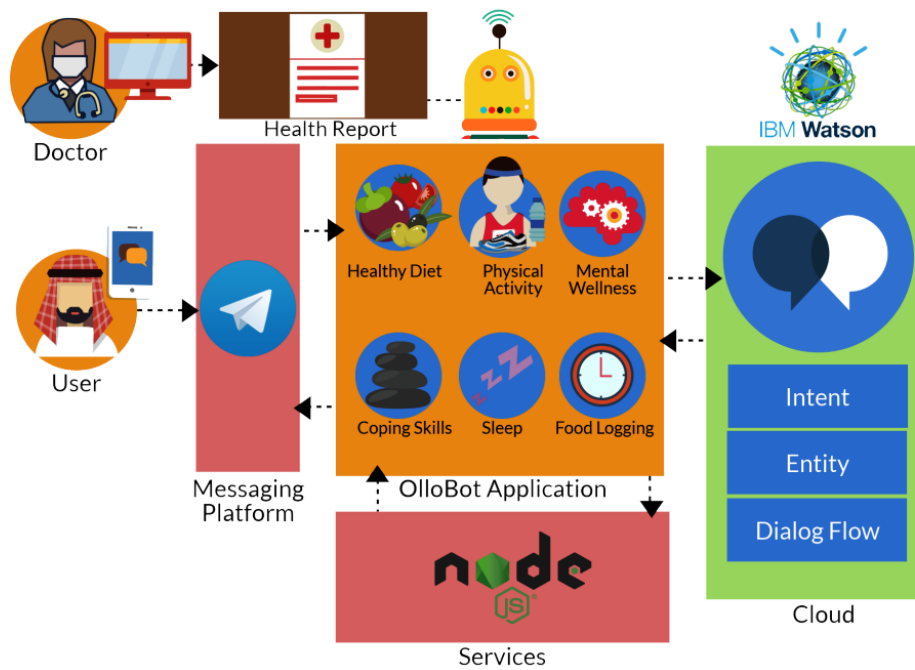


Figure 3.5: OlloBot High-level Architecture

foreign tourists and non-Arabic speakers, as well as it was programmed with IBM Watson platform. These models relied on a wide range of data collected from various available means. The chatbots are based on a dataset that covers 13 areas and 81 cities in Saudi Arabia and it has more than 845 data entries that could be provided to the user upon request [Alhumoud et al., 2022].

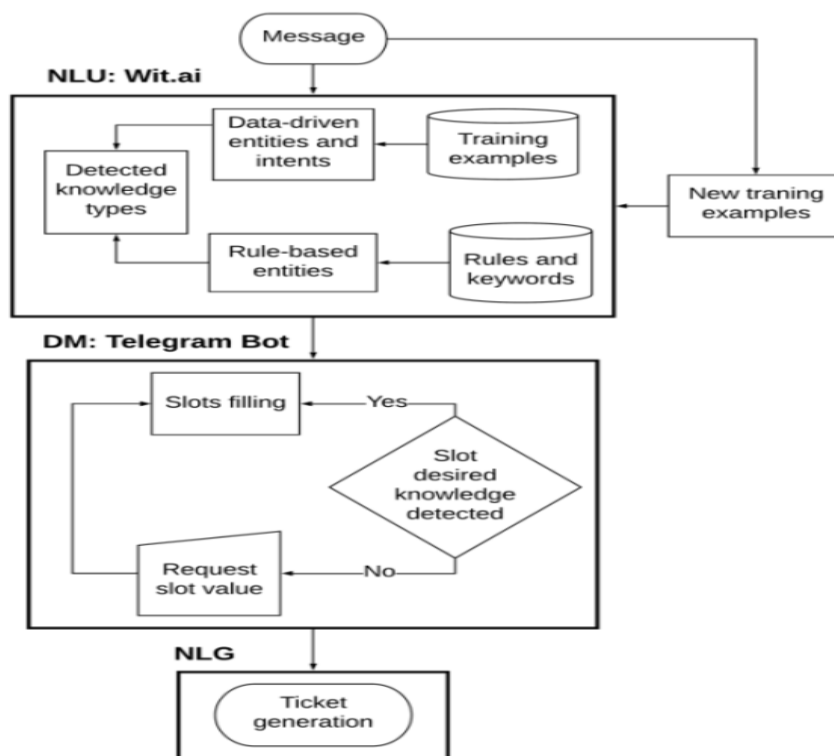


Figure 3.6: Flight booking DS architecture

4

Developing and building chatbot

4.1 Introduction

In this chapter, we explain how the chatbot is built and developed in practice and how the techniques that were explained in the previous chapters were used.

The process of building and developing a chatbot goes through stages that are related to each other. Construction stages begin first on defining the technology used in developing and building the chatbot then comes to the stage of data collection that the chatbot will train on and which it will answer based on. The next stage is the implementation stage, in which the programming and development of the chatbot begin, And then evaluate and test the performance of the chatbot, and then finally integrate it with various communication platforms.

4.2 Chatbot Goal

We recall that the main task of the chatbot that we are developing is to provide answers to user questions in the tourism field, that is, it will act as a tour guide. It will be able to provide information about existing hotels, tourist agencies, tourist areas, prices for transportation services, restaurants and existing tourism activities.

4.3 Development Tools

In the section 1.4, we have seen see some of the existing and available technologies and platforms that aim to build and develop a conversational agent, among these existing technologies and platforms, we chose and used the *Rasa platform*, in addition to relying partially on GPT-3, that is, we combined the characteristics and the features provided by Rasa with the features and characteristics provided by GPT-3 to try as much as possible to reach an integrated chatbot

Although it was possible to develop and build chatbots using the Rasa platform only or by relying on GPT-3 only, we decided to integrate them because some features are not supported by the GPT-3 model and we find them in Rasa, such as dealing with images and sounds, ... Hence, we decide to combine them to get a better result

4.4 Data Collection

It is a very important step in building a chatbot, and it plays a very interesting role in the effectiveness and efficiency of the chatbot. Since the chatbot will be in the tourism sector, the data we collect will be confined to this sector, and we have chosen the city of *Ghardaia*, due to the limited time and

the difficulty of fully collecting data on tourism in Algeria. Thus, our research was restricted to data collection for the city of Ghardaia only. In fact, this means that the chatbot cannot answer questions that are not related to Ghardaia.

4.4.1 Data Collection Method

The data collection method was done manually due to the multiplicity of sources that include tourist information and data. In the collection process, we relied on some websites that provide information about hotels and information about tourist agencies like tripadvisor¹ and booking.com² as well as the official website of the Directorate of Tourism in Ghardaïa-Wilaya³. We also relied on Facebook pages for facilities and hotels, as they provide some information such as pictures and the latest activities. We also relied on the data collection process on Google Maps, through which we were able to know the hotels located in a specific area and the facilities located near them, as well as their coordinates and locations. In some cases, it requires moving to some hotels and tourist facilities to collect data and information about them, as well as watching some tourist, cultural and historical videos to collect data related to the history and establishment of cities.

4.4.2 Data Structure

After carrying out the data collection process which included collect information about hotels, tourist agencies, tourist areas, information about transportation in the state, and some historical information pertaining to it, this data collected is stored according to a specific format and structure, in preparation for submitting it to the machine learning model to carry out training on it. The format and structure of storing the collected data depends mainly on the technique used in training the model.

In our case, we have data stored in two different ways, structured data, and unstructured data:

- *Unstructured Data:* Part of the data that was collected was stored in an unstructured way, that is, it is a question and answer only, as illustrated in Figure 4.1. This data is intended to be presented to Rasa for training purpose.
- *Structured Data:* A second part of the data that we stored in an organized and structured manner, due to the nature of this information that requires organizing its data according to tables. For example, the information related to hotels as shown in Figure 4.2

¹<https://ar.tripadvisor.com/>

²<https://www.booking.com>

³<https://ghardaia.mta.gov.dz/>



Figure 4.1: An example of storing unstructured data



Figure 4.2: An example of storing structured data

4.5 Implementation of the Chatbot

4.5.1 Create the Project

The creation of a Rasa project is done by executing the command *rasa init*. After executing this command, the project files necessary to start work are automatically created as shown in Figure 4.3.

4.5.2 Insert the Training Data

As we mentioned earlier, the data stored is in two forms, training data for RASA and training data for the GPT-3 model.

■ Rasa training data

The first step is to enter the collected data into the project for the model to train on. The training data takes several forms, it will be in the form of *intent*, *story* and *rule*

- **Intent Training Data:** As for the intent training data (that makes the model define the intent intended by the user), this data is placed in the *intent.yml* file, as shown in Figures

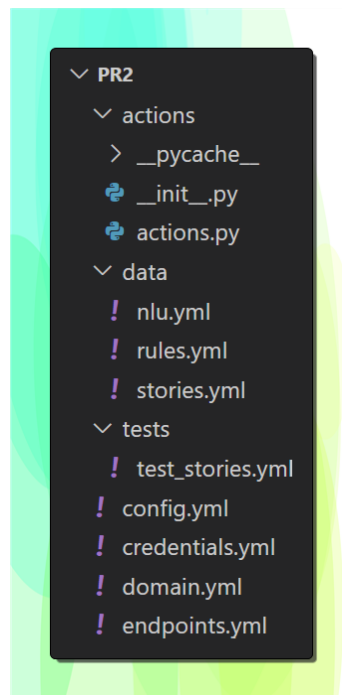


Figure 4.3: Project structure in Rasa

4.4 and 4.5.

There are also some training data to which we add additional details and information, such as letting the model know that the entered word represents the name of a hotel or that it represents the name of a city, and this helps the model to Understand the user's question well (Figure 4.6).

- **Responses:** When the chatbot discovers that the user is asking about a specific thing, now comes to the step of answering the question, so we put the answers to the expected questions that the user can ask, placed in the *domain.yml* file and some custom answers are placed in a file *actions.py*.
 1. **Default answer:** Figure 4.7 shows some of the ways to create responses to a user's question We may add other things to the responses, for example, after the user answers, we offer him the option to press the buttons instead of writing his question. Figure 4.8 shows an example when the user has asked, "How are you?" We expect that there will be two answers, either he is fine or in bad condition, and therefore two buttons will appear. It is also possible to provide answers with pictures (Figure 4.9).
 2. **Customer response:** Figure 4.10 shows two different answers, the first (`action_image_equestere`) is executed in case the user requested pictures of the equestrian club, and the second (ac-



Figure 4.4: Intent Training Data 1

tion_image_belvedere) if the user requested pictures of a *Belvedere* hotel.

- **Stories:** As previously explained, stories are training data that make the chatbot able to implement appropriate responses. Figure 4.11 Shows some examples of training stories that we used.
- **Rules:** Another type of training data is rules, which must be compatible with training stories. Figure 4.12 shows an example of a rule that we used. When a user's message is classified as a goodbye, the response should be a goodbye message.
- **GPT-3 Training Data:** In this part, we show how we use GPT-3 and how we provide its training data.

We mentioned earlier that the data for GPT-3 is stored in a structured way. Figure 4.13 shows an example of stored data for hotel contact numbers. It is stored in an xls file due to the ease of dealing with this type of file.

Figure 4.14 shows an example of how to provide an answer to a user's question using GPT-3.

```

nlu.yml

- intent: City_info
  examples: |
    - اريد معلومات حول مدينة [بريان]
    - اريد بعض المعلومات حول المدينة
    - اريد تعريف للمدينة
    - عطيلي بعض المعلومات على [بريان]
    - اريد معلومات حول بلدية [القرارة]

- intent: equestre_ghardaia
  examples: |
    - هل يوجد نوادي للفروسية في غرداية ؟
    - هل يوجد من يقدمون خدمات ركوب الخيل ؟
    - اريد ركوب الخيل في غرداية من يمكنهم تقديم ذلك
    - اريد تجربة ركوب الخيل ؟
    - ابحث عن نادي ركوب الخيل ؟
    - اين اجد مدارس تعليم ركوب الخيل ؟
    - ابحث عن نادي فروسية ؟

- intent: road_to_ghardaia
  examples: |
    - كيف اصل الى ولاية غرداية ؟
    - كيف يمكنني الوصول الى ولاية غرداية ؟
    - كيفاش نقدر نوصل لمدينة غرداية

- intent: zoo_ghardaia
  examples: |
    - هل يوجد حدائق حيوانات في ولاية غرداية ؟
    - ماهي حدائق الحيوانات الموجودة في ولاية غرداية ؟
    - ابحث عن حديقة حيوانات
    - اريد زيارة حديقة الحيوانات
    - ابحث عن حدائق حيوانات ؟

- intent: tafilaalt_info
  examples: |
    - ابحث عن معلومات حول منطقة تافيلالت
    - واه هي تافيلالت
    - عطيلي معلومات على تافيلالت
    - ماهي منطقة تافيلالت
  
```

Figure 4.5: Intent Training Data 2

4.5.3 Model Training

After the training data has been configured and entered, the next step is the training process. First, the settings for training and the policies that will be used in the chatbot are determined, as well as the language of the chatbot and the number of training times. In general, this preparatory stage is before starting to train the model and setting Model Configuration (like learning rate, batch size, optimizer, loss function, and number of epoch).

These settings are included in the *config.yml* file as shown in Figure 4.15. After adjusting the settings, the model is trained by the command *rasa train*.

After the completion of the training process, the model appears in the model folder. Figure 4.16 shows the completion of the training process for all components and policies used in the project.

4.5.4 Test and Evaluation

After completing the model training, the next step is to experiment and test the model. As we mentioned earlier, Rasa consists of two main components, Rasa NLU and Rasa Core, each of these two components has its data to test.

1. **Testing data for Rasa NLU:** The testing data for Rasa NLU consists of different examples

```

- intent: timing_visit_ghardaia
examples: |
- ماهو افضل وقت لزيارة ولاية غرداية ؟
- ماهو الجو المناسب لزيارة ولاية غرداية ؟
- ماهو الفصل المناسب لزيارة غرداية ؟
- ماهو افضل فصل لكي ازور غرداية؟
- واش من وقت مناسب نزور فيه غرداية؟
- واش هو لوقت المريح لي نزور فيه غرداية
- واش من شهر نزور فيه غرداية
- ماهو افضل شهر يمكنني ان ازور فيه غرداية
- ماهو الفصل او الفترة التي تقترحها للقيام بزيارة ولاية غرداية

- intent: equestre_ghardaia
examples: |
- هل يوجد نوادي للفروسية في غرداية ؟
- هل يوجد من يقدمون خدمات ركوب الخيل ؟
- اريد ركوب الخيل في غرداية من يمكنهم تقديم ذلك
- اريد تجربة ركوب الخيل ؟

- intent: hotel_adresse
examples: |
- اين يقع فندق [الجنوب] (hotel)
- وين بلاصة جاي فندق [الجنوب] (hotel)
- ماهو موقع الفندق ؟
- اين يتواجد هذا الفندق ؟
- اين يقع الفندق ؟
- كيف يمكن ان اصل الى فندق [الجنوب] (hotel)
- ماهو موقع هذا الفندق ؟
- ماهو عنوان الفندق ؟

- intent: City_carties
examples: |
- ماهي الاحياء الموجودة في مدينة [بريان] (city)
- ماهي الاحياء الموجودة فيها
- ماذا يوجد احياء في هذه المدينة
- ماهي تسمية الاحياء في مدينة [غرداية] (city)
- اريد الاحياء الموجودة في [القرارة] (city)
- واش هي الحومات الموجودة في [زلفانة] (city)

```

Figure 4.6: An example of storing unstructured data

from different intents (Figure 4.17).

2. **Testing data for Rasa Core:** Test data for Rasa Core are stories that represent examples of conversations, Figure 4.18 shows an example of the test story.

4.5.5 Result and Analysis

1. **Rasa NLU test results:** The testing results for this component are shown in the Table 4.1. We notice through the results that the model works very well with high accuracy, resilience, and F1 score. These measures reveal the developed chatbot effectiveness in correctly classifying intentions. Figure 4.19 shows the confusion matrix of intents.
2. **Rasa Core result :** We note through Table 4.3 and the confusion matrix depicted in Figure 4.20 that the model shows decent performance with fairly high average macro-precision, recall, and F1-score. However, the weighted average shows that the model struggles to address class imbalance due to the drop in precision and recall when class distributions are considered. A di-

A screenshot of a code editor window titled 'domain.yml'. The editor has a light purple background and three window control buttons (red, yellow, green) in the top-left corner. The code is written in a dark font and is organized into four sections, each starting with a green header: 'utter_welcome:', 'utter_bot_service:', 'utter_timing_visit_ghardaia:', and 'utter_best_place_in_wilaya_ghardaia:'. Each section contains one or two lines of text in Arabic, preceded by a hyphen and the word 'text:'. The text describes a chatbot's responses in Arabic, including a welcome message, a service description, a visit timing recommendation, and a list of best places in Ghardaia.

```
domain.yml

utter_welcome:
- text: العفو, نتمنى ان تكون الاجابات مفيدة
- text: عفوا

utter_bot_service:
- text: انا عبارة عن متحدث الي , هنا بمثابة مرافقك و مرشدك السياحي في ولاية غرداية,ساكون متوفر لك طوال لوقت لتزويدك بجميع المعلومات خصوصا السياحية التي تحتاجها و المتعلقة بولاية غرداية , لجعل تجربتك السياحية في الولاية ناجحة

utter_timing_visit_ghardaia:
- text: افضل فترة لزيارة ولاية غرداية تكون في فصل الربيع و الشتاء

utter_best_place_in_wilaya_ghardaia:
- text: زيارة القصور - سوق القديم - الاقامات السياحية - الحمامات المعدنية في زلفانة - مخيم الهدار في سبب
```

Figure 4.7: Response

alog accuracy of 50% indicates that the model needs further improvement to better understand and respond to the dialog. In fact, we will focus on this over time.

4.6 Deploy and Run the Chatbot in Telegram

Following are a summary of the steps required to operate the created chatbot in Telegram.

1. We created a bot in Telegram with the help of BotFather and got our bot tokens for the linking process (Figure 4.21)
2. After creating the bot on Telegram, the next step is to put its settings (Tokens) in Rasa, exactly in the *credential.yml* file as shown in Figure 4.22.
3. We relied on *ngrok* to create a bridge between Telegram and Rasa.
4. For the chatbot to start working, the upside-down server must be run with the *rasa run* command (Figure 4.23). Therafter, the chatbot is ready to respond to various user questions.

```
domain.yml

utter_how_are_you:
- text: ؟ مرحبا كيف حالك
  buttons:
  - title: بخير
    payload: /mode_happy
  - title: لمست بخير
    payload: /mode_sad
```

Figure 4.8: Responses - Buttons

```
domain.yml

utter_ghardaia_pic:
- text: "صورة لمدينة غرداية"
  image: "https://ghardaia_pic_example.png"
```

Figure 4.9: Responses - Images

4.7 Example of Chatbot Conversations

Figures 4.24-4.27 show some examples of conversations between the chatbot and a user on various topics on which it was trained.

```

actions.py

class equestreClub(Action):

    def name(self) -> Text:
        return "action_image_equestre_ghardaia"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        img = {"img1", "img2", "img3"}
        for im in img:
            dispatcher.utter_message(image=im)
        return

class ImageHotel(Action):

    def name(self) -> Text:
        return "action_image_belvedere"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        img = {"img1", "img2", "img3"}
        for im in img:
            dispatcher.utter_message(image=im)
        return [SlotSet("hotel", "بالفندق")]

```

Figure 4.10: Example response - Pictures of a club and a hotel -

Table 4.1: Rasa NLU Evaluation Results

		Precision	Recall	F1-Score
Macro Avg	Class Avg	0.9981	0.9972	0.9976
	Support	434		
Weighted Avg	Class Avg	0.9979	0.9977	0.9977
	Support	434		
Micro Avg	Class Avg	0.9977	0.9977	0.9977
	Support	434		


```
stories.yml

- story: happy path
  steps:
  - intent: greet
  - action: action_greet

- story: hotel ask story
  steps:
  - intent: asking_hotel_service
  - action: action_hotel_des_service_data
  - intent: hotel_price_reservation
  - action: action_hotel_price_reservation
  - intent: hotel_adresse
  - action: action_hotel_adresse
  - intent: hotel_contact
  - action: action_hotel_contact_data
  - intent: thanks
  - action: utter_welcome

- story: generale story 2
  steps:
  - intent: greet
  entities:
  - name: محمد
  - action: action_greet
  - intent: bot_challenge
  - action: utter_bot_service
  - intent: adv_visit_ghardaia
  - action: utter_beauty_ghardaia
  - intent: province_ghardaia
  - action: utter_ghardaia_province
  - intent: amayasse_restaurant
  - action: utter_amayasse_restaurant
```

Figure 4.11: Training stories

```
rules.yml

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

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

Figure 4.12: Rules

B	A	
	ارقام الاتصال	اسم الفندق 1
"0552152596"		بن حمودة 2
Fixe: 029-239-966 - Mobile: 0555-043-832 - Fax: 029-23		بالفيدار 3
'029286969		فندق مزاب 4
: tel		
029.28.14.04		
029.28.16.24		
029.28.59.07		
0770.93.95.02		
: Fax		
029.28.15.20		الجنوب 5
tel : 0550 66 88 84		فندق الاء 6
Tel :		
029.23.60.19		
0550.26.10.44		
0770.24.73.70		النخيل 7
tel : 029.23.12.37		
0774.03.08.02		فندق بلعجال 8
Tel : 029.88.32.84		
0770.60.99.69		
0671.47.34.88		
029.88.68.72		الكرامة 9
tel : 029.23.34.48		
029.23.34.46		
tel : 029.23.37.20		الطاسيلي 10
029.23.34.44		
tel : 029.88.49.01		الريم 11
029.88.32.84		المحطة 12

Figure 4.13: Hotel contact data

Table 4.2: Rasa Core Evaluation Results

		Precision	Recall	F1-Score
Macro Avg	Class Avg	0.8875	0.9392	0.9045
	Support	90		
Weighted Avg	Class Avg	0.8047	0.8667	0.8243
	Support	90		
Micro Avg	Class Avg	0.8667	0.8667	0.8667
	Support	90		

```

actions.py

class Hotelcontact(Action):
    def name(self):
        return "action_hotel_contact_data"
    def run(self, dispatcher, tracker, domain):
        hotel = tracker.get_slot("hotel")
        city = tracker.get_slot("city")
        user_question= tracker.latest_message['text']
        msg = Gpt( user_question, history , hotel_contact_data , city , hotel)
        history_conve = "-Que:"+str(user_question)+"\n-Answer:"+str(msg)+"\n"
        finHisto = str(history)+"\n"+str(history_conve)
        dispatcher.utter_message(text=str(msg))
        return [SlotSet("results_hotel_conv", finHisto)]

```

Callout box content: `hotel_contact_data = "C:\\Users\\ASUS\\Desktop\\Rasa Pro
Project\\Pr1\\dataset\\hotel_contact.csv"`

Figure 4.14: Customer Action - Hotel Contact response

When the user's question is related to searching for the contact number of a specific hotel, we ask GPT-3 to provide an answer to the user's question based on the following:

- (1) We provide him with data regarding the hotel list and their phone numbers in a CSV file (we convert the original xls file to CSV)
- (2) A user question
- (3) Previous conversations and the name of the last hotel the user talked about (because the user's question might be as follows: What is the hotel's contact number? Here is a question where the name of the hotel he is referring to can only be discovered through previous conversations)

Table 4.3: Conversation Accuracy

Conversation Accuracy	
Accuracy	0.5
Correct	3
With Warnings	0
Total	6

```
config.yml

language: ar

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
  analyzer: "char_wb"
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 100
- name: RegexEntityExtractor
  use_lookup_tables: True

policies:
- name: MemoizationPolicy
- name: TEDPolicy
  max_history: 10
  epochs: 200
- name: RulePolicy
```

Figure 4.15: Model Configuration

```

PS C:\Users\ASUS\Desktop\Rasa Pro Project\Pr4> rasa train
c:\users\asus\appdata\local\programs\python\python39\lib\site-packages\rasa\core\tracker_store.py:876: MovedIn2
0Warning: Deprecated API features detected! These feature(s) are not compatible with SQLAlchemy 2.0. To prevent
incompatible upgrades prior to updating applications, ensure requirements files are pinned to "sqlalchemy<2.0"
. Set environment variable SQLALCHEMY_WARN_20=1 to show all deprecation warnings. Set environment variable SQL
ALCHEMY_SILENCE_UBER_WARNING=1 to silence this message. (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e
/b8d9)
Base: DeclarativeMeta = declarative_base()
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'CountVectorsFeaturizer' from cac
he.
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'DIETClassifier' from cache.
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'EntitySynonymMapper' from cache.
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'LexicalSyntacticFeaturizer' from
cache.
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'MemoizationPolicy' from cache.
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'RegexEntityExtractor' from cache
.
2023-05-28 20:30:45 INFO      rasa.engine.training.hooks - Restored component 'RegexFeaturizer' from cache.
2023-05-28 20:30:46 INFO      rasa.engine.training.hooks - Restored component 'ResponseSelector' from cache.
2023-05-28 20:30:46 INFO      rasa.engine.training.hooks - Restored component 'RulePolicy' from cache.
2023-05-28 20:30:46 INFO      rasa.engine.training.hooks - Restored component 'TEDPolicy' from cache.
Your Rasa model is trained and saved at 'models\20230528-203043-flat-brad.tar.gz'.

```

Figure 4.16: Model training process

```

test_data.yml

- intent: tafilalt_rule
  examples: |
    - هل هناك اشياء محظورة عند زيارة تافيلالت
- intent: road_to_ghardaia
  examples: |
    - كيف اصل الى ولاية غرداية ؟
- intent: tafilalt_info
  examples: |
    - ماهي منقطة تافيلالت
- intent: province_ghardaia
  examples: |
    - ماهي البلديات الموجودة في ولاية غرداية

```

Figure 4.17: Testing NLU data

```
test_stories.yml

- story: te1
  steps:
  - user: |
    مرحبا
    intent: greet
  - action: action_greet
  - user: |
    ؟ مارأيك في ذلك (city) افكر في ان ازور ولاية [غرداية]
    intent: adv_visit_ghardaia
  - action: utter_beauty_ghardaia
  - user: |
    ؟ هل يوجد فيها فنادق
    intent: hotel_info
  - action: action_hotel_city_existe
```

Figure 4.18: Rasa Core Testing data

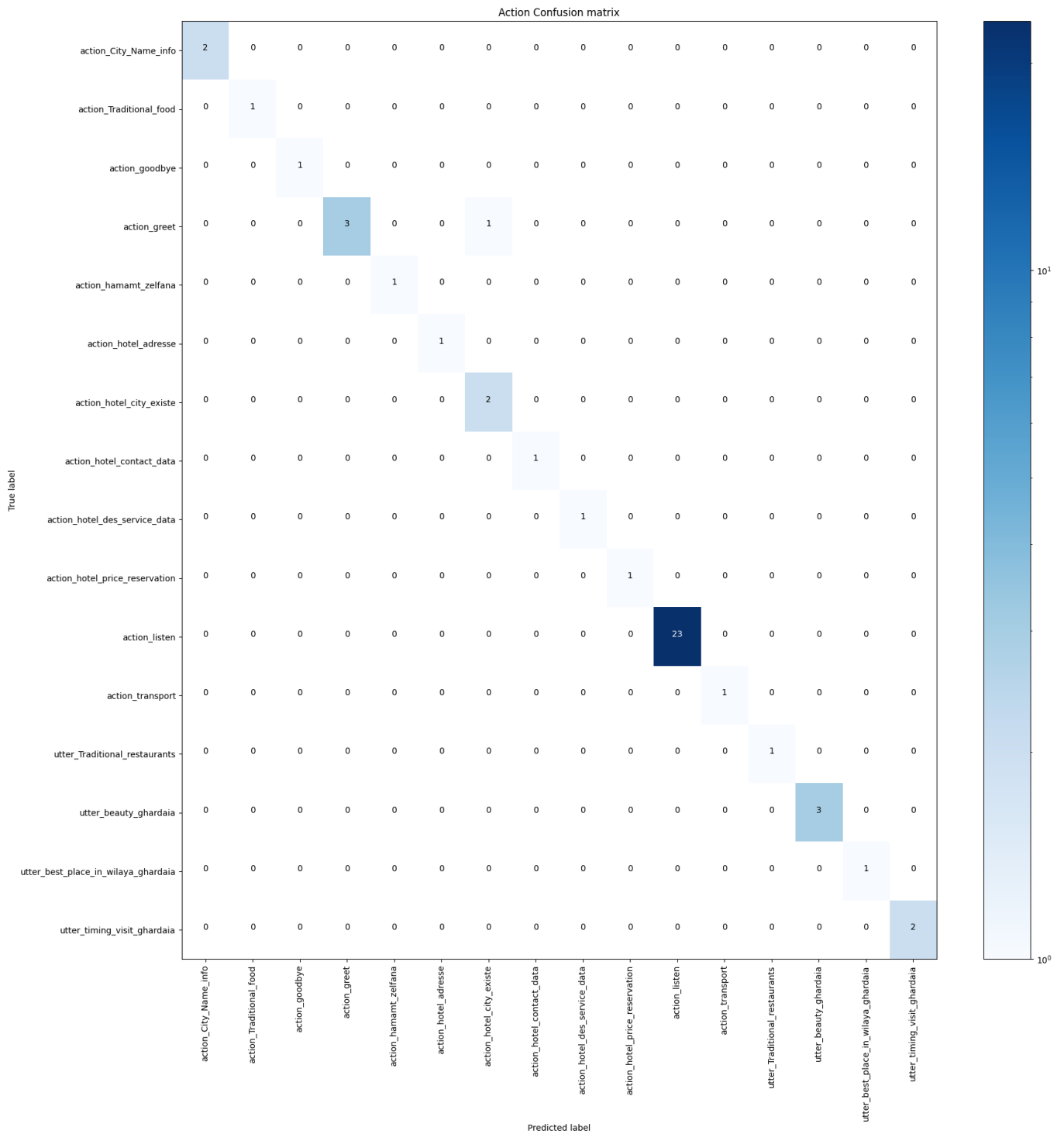


Figure 4.20: Confusion matrix for Actions

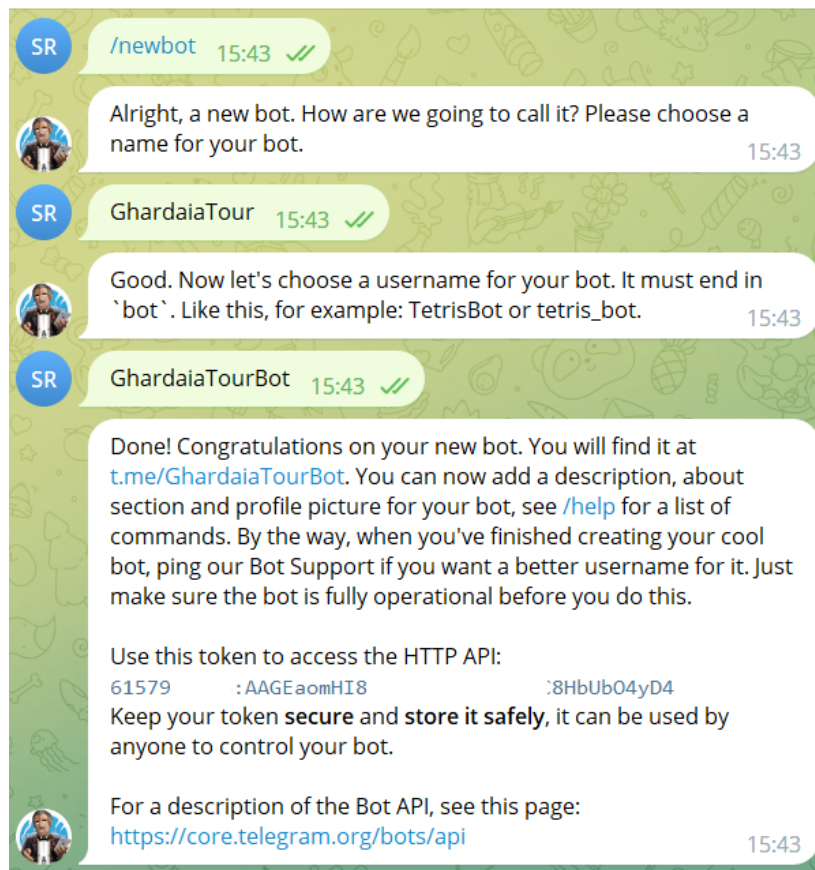


Figure 4.21: Creating a bot in Telegram with BOTFather

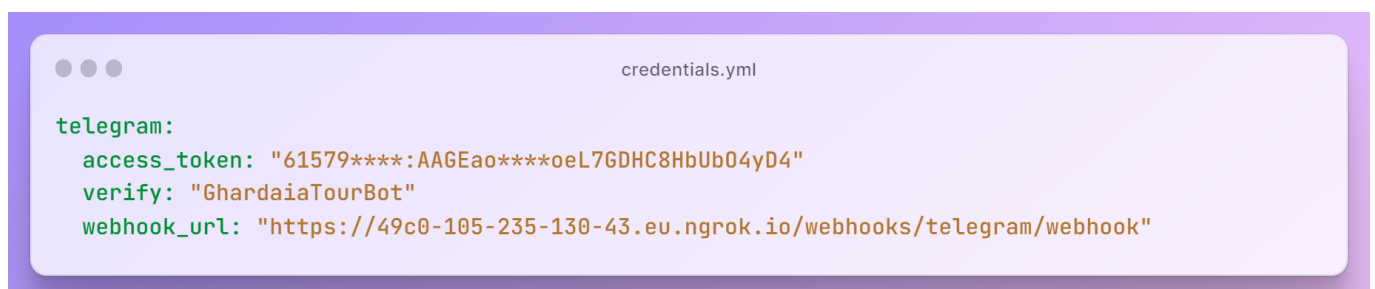


Figure 4.22: Set bot tokens in Rasa

```
PS C:\Users\ASUS\Desktop\Rasa Pro Project\Pr4> rasa run
c:\users\asus\appdata\local\programs\python\python39\lib\site-packages\rasa\core\tracker_store
.py:876: MovedIn20Warning: Deprecated API features detected! These feature(s) are not compatib
le with SQLAlchemy 2.0. To prevent incompatible upgrades prior to updating applications, ensur
e requirements files are pinned to "sqlalchemy<2.0". Set environment variable SQLALCHEMY_WARN_
20=1 to show all deprecation warnings. Set environment variable SQLALCHEMY_SILENCE_UBER_WARNI
NG=1 to silence this message. (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  Base: DeclarativeMeta = declarative_base()
2023-05-28 19:31:39 INFO      root - Starting Rasa server on http://0.0.0.0:5005
2023-05-28 19:31:40 INFO      rasa.core.processor - Loading model models\20230522-212353-creat
ive-chain.tar.gz...
c:\users\asus\appdata\local\programs\python\python39\lib\site-packages\rasa\utils\train_utils.
py:528: UserWarning: constrain_similarities is set to `False`. It is recommended to set it to
`True` when using cross-entropy loss.
  rasa.shared.utils.io.raise_warning(
2023-05-28 19:32:28 INFO      root - Rasa server is up and running.
```

Figure 4.23: Rasa server Execution



Figure 4.24: Conversation between a user and a chatbot - Information about Ghardaia Province - .

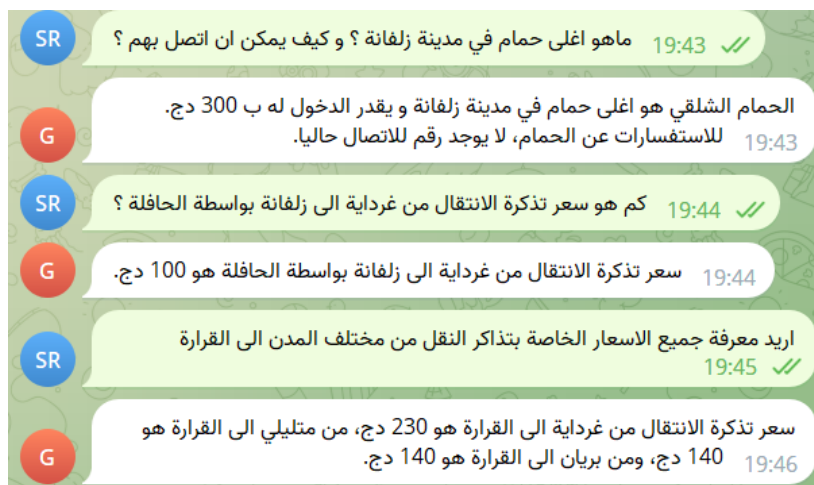


Figure 4.25: A conversation about transportation and the city of Zelfana



Figure 4.26: Conversation about the equestrian club



Figure 4.27: Conversation about a hotel

Conclusion

The significance of technology in the tourism industry is growing, and chatbots have the potential to revolutionize consumer service and engagement.

Our master's project contributes to addressing the gap in the availability of Arabic chatbots tailored to the Algerian tourism sector, concentrating on the city of Ghardaia. The targeted chatbot should effectively respond to users' queries about tourism in Ghardaia, enhancing their travel experience and promoting the city's tourism industry.

The research process included a thorough examination of chatbot technologies and their advantages and disadvantages, an in-depth understanding of the Rasa platform and integrating GPT-3 to Rasa platform, a state-of-the-art review of existing chatbots, and the development and implementation of the Arabic chatbot for tourism in Ghardaia.

However, there are certain limitations to this research. The scope of the chatbot is limited to the city of Ghardaia, it may require further development to cater to other tourist destinations in Algeria. Additionally, the chatbot's performance may be influenced by the quality and comprehensiveness of the training data used.

Despite these limitations, the study has yielded significant findings. The developed chatbot demonstrates promising results in addressing users' questions and enhancing their overall travel experience. These findings have both scientific and practical implications, as they contribute to the growing body of knowledge on chatbot development and its application in the tourism industry.

In terms of future perspectives, further research could focus on expanding the chatbot's scope to cover other Algerian cities or even other countries in the region. Additionally, incorporating advanced natural language processing techniques and machine learning algorithms could improve the chatbot's performance and enable it to handle more complex user queries. Finally, integrating the chatbot with other communication channels, such as social media platforms or dedicated mobile applications, could further enhance its accessibility and utility for tourists.

References

- [AbuShawar and Atwell, 2015] AbuShawar, B. and Atwell, E. (2015). Alice chatbot: Trials and outputs. *Computación y Sistemas*, 19(4):625–632.
- [Adamopoulou and Moussiades, 2020] Adamopoulou, E. and Moussiades, L. (2020). An overview of chatbot technology. In *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16*, pages 373–383. Springer.
- [Al-Ajmi and Al-Twairesh, 2021] Al-Ajmi, A.-H. and Al-Twairesh, N. (2021). Building an arabic flight booking dialogue system using a hybrid rule-based and data driven approach. *IEEE Access*, 9:7043–7053.
- [AlHumoud et al., 2018] AlHumoud, S., Al Wazrah, A., and Aldamegh, W. (2018). Arabic chatbots: a survey. *International Journal of Advanced Computer Science and Applications*, 9(8).
- [Alhumoud et al., 2022] Alhumoud, S., Diab, A., AlDukhai, D., AlShalhoub, A., AlAbdullatif, R., Alqahtany, D., Alalyani, M., and Bin-Aqeel, F. (2022). Rahhal: A tourist arabic chatbot. In *2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 66–73. IEEE.
- [Ali and Habash, 2016] Ali, D. A. and Habash, N. (2016). Botta: An arabic dialect chatbot. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 208–212.
- [Arionesei et al., 2014] Arionesei, G., Stanciu, P., Moroşan, S. A.-A., and Cosma, P. (2014). Tourism today: Why is it a global phenomenon. In *International Conference Sustainable Development in Conditions of Economic Instability*, volume 240, page 248.
- [Bocklisch et al., 2017] Bocklisch, T., Faulkner, J., Pawlowski, N., and Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*.
- [Carpenter, 1997] Carpenter, R. (1997). Jabberwacky 13.0—learning artificial intelligence—ai software applications. *Retrieved*, 15:2023.

- [Chung and Park, 2019] Chung, K. and Park, R. C. (2019). Chatbot-based healthcare service with a knowledge base for cloud computing. *Cluster Computing*, 22:1925–1937.
- [Colby et al., 1972] Colby, K. M., Hilf, F. D., Weber, S., and Kraemer, H. C. (1972). Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, 3:199–221.
- [Colby et al., 1971] Colby, K. M., Weber, S., and Hilf, F. D. (1971). Artificial paranoia. *Artificial intelligence*, 2(1):1–25.
- [Fadhil et al., 2019] Fadhil, A. et al. (2019). Ollobot-towards a text-based arabic health conversational agent: Evaluation and results. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 295–303.
- [Gössling, 2017] Gössling, S. (2017). Tourism, information technologies and sustainability: an exploratory review. *Journal of Sustainable Tourism*, 25(7):1024–1041.
- [Gupta et al., 2020] Gupta, A., Hathwar, D., and Vijayakumar, A. (2020). Introduction to ai chatbots. *International Journal of Engineering Research and Technology*, 9(7):255–258.
- [Hijjawi et al., 2014] Hijjawi, M., Bandar, Z., Crockett, K., and Mclean, D. (2014). Arabchat: An arabic conversational agent. In *2014 6th International Conference on Computer Science and Information Technology (CSIT)*, pages 227–237. IEEE.
- [Jafari et al., 2000] Jafari, J., Baretje, R., Buhalis, D., Cohen, E., Dann, G. M., Collison, F., Din, K. H., Fayos-Sola, E., and Fletcher, J. (2000). *Encyclopedia of tourism*. Taylor & Francis.
- [Leiter et al., 2023] Leiter, C., Zhang, R., Chen, Y., Belouadi, J., Larionov, D., Fresen, V., and Eger, S. (2023). Chatgpt: A meta-analysis after 2.5 months. *arXiv preprint arXiv:2302.13795*.
- [malki and chenini, 2022] malki, o. e. and chenini, a. (2022). Integrating the chatbot technology as one of the applications of artificial intelligence to enhance services in the hospitality and tourism sector. *imtiaze journal*, 6(1):341–358.
- [Microsoft, 2023] Microsoft (2023). Bing at microsoft build 2023: Continuing the transformation of search. Available at: https://blogs.bing.com/search/may_2023/Bing-at-Microsoft-Build-2023. Accessed on 2023.
- [Nica et al., 2018] Nica, I., Tazl, O. A., and Wotawa, F. (2018). Chatbot-based tourist recommendations using model-based reasoning. In *ConfWS*, pages 25–30.
- [OpenAI, 2022] OpenAI (2022). Chatgpt: Improving language generation with the clip vision system. Available at: <https://openai.com/blog/chatgpt>. Accessed on 2023.

- [Rasa, 2022] Rasa (2022). Introducing dual intent and entity transformer (diet): State-of-the-art performance on a lightweight architecture. Available at: <https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-l>. Accessed on May, 2023.
- [Rasa, 2023] Rasa (2023). Rasa Documentation: Architecture Overview. <https://rasa.com/docs/rasa/arch-overview/>. Accessed on May, 2023.
- [Sano et al., 2018] Sano, A. V. D., Imanuel, T. D., Calista, M. I., Nindito, H., and Condrobimo, A. R. (2018). The application of agnes algorithm to optimize knowledge base for tourism chatbot. In *2018 International Conference on Information Management and Technology (ICIMTech)*, pages 65–68. IEEE.
- [Shawar and Atwell, 2004] Shawar, A. and Atwell, E. (2004). An arabic chatbot giving answers from the qur’an. In *Proceedings of TALN04: XI Conference sur le Traitement Automatique des Langues Naturelles*, volume 2, pages 197–202. ATALA.
- [Song et al., 2018] Song, Y., Yan, R., Li, C.-T., Nie, J.-Y., Zhang, M., and Zhao, D. (2018). An ensemble of retrieval-based and generation-based human-computer conversation systems.
- [Trisha and Sahana, 2022] Trisha, K. R. and Sahana, A. (2022). Chatbot application for tourism using deep learning. *Ijrasnet Journal For Research in Applied Science and Engineering Technology*, 10.
- [Vlasov et al., 2019] Vlasov, V., Mosig, J. E., and Nichol, A. (2019). Dialogue transformers. *arXiv preprint arXiv:1910.00486*.
- [Wallace, 1995] Wallace, R. (1995). Artificial linguistic internet computer entity (alice). *City*.
- [Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
- [Worswick, 2010] Worswick, S. (2010). Mitsuku chatbot: Mitsuku now available to talk on kik messenger. Retrieved on 2023; Available from: <https://www.pandorabots.com/mitsuku/>.
- [Wu et al., 2018] Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., and Weston, J. (2018). Starspace: Embed all the things! In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [Yuan et al., 2019] Yuan, Y., Tseng, Y.-H., and Ho, C.-I. (2019). Tourism information technology research trends: 1990-2016. *Tourism review*.