PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science Department

Ordre N° : IA7/M2/2021

Thesis

Submitted in fulfilment of the requirements for the Masters degree in

Computer science

Option : **Artificial Intelligence**

A Deep Learning-based approach for Chatbot: medical assistance a case study

---

By :

GUERRI MOHAMED FADHLALLAH

Members of the jury :

Full Name	Grade	President
MERIZIG Abdelhak	MCB	Supervisor
Full Name	Grade	Member

Session 2021

Acknowledgements

First and foremost, thanks to Allah the most graceful and merciful for his showers of blessings throughout my research work to complete it successfully.

I'm extremely grateful to my parents for their care, love, encouragements and prayers. Special thanks to all my family members, for contributing by their love and support.

I would like to express my deep and sincere gratitude to my research supervisor **Dr. Abdelhak Merizig** for the continuous support of my thesis study and research, for his immense knowledge, precious guidance. I really admire his support and understanding all along the way.

My sincere thanks also goes to all teachers of the **Computer Science Department** who helped me in my education.

I extend my appreciation to my precious friends and classmates, having such great people around me had a huge impact on my life and pushed me forward to always give my best.

To everyone i love and care about, thank you.

Guerri Mohamed Fadhlallah

ملخص

لقد حقق تطور الذكاء الاصطناعي (Artificial Intelligence) تقدمًا كبيرًا في المجال الطبي. باستخدام الذكاء الاصطناعي ، يمكن للمهنيين الطبيين والممرضات تحسين عملية اتخاذ القرار وقضاء وقت أقل في محاولة جمع معلومات المرضى. يمكن للذكاء الاصطناعي تحسين التواصل في المستشفى من خلال دعم دليل للمرضى ، خاصة في الأزمة العالمية الحالية لوباء (COVID-19) حيث أن الاتصال المباشر بين المرضى والطاقم الطبي قد تكون خطيرة. تجعل هذه الظروف الكثير من المساعدين الطبيين قلقين بشأن حالتهم الصحية لأنهم مضطرون إلى الاتصال المباشر بالمرضى. إحدى تقنيات الذكاء الاصطناعي التي يمكنها حل هذه المشكلة تتمثل في المساعد الطبي اللآلي المتحدث (Chatbot) . في هذا المشروع ، نبني نظام متحدث آلي ذكي يمكنه التفاعل مع المريض لمساعدته في زيارات المستشفى. نجمع بين خوارزميات التعلم العميق (Deep learning) الذي يتمثل في نموذج الذاكرة طويل و القصير المدى ثنائي الاتجاه (Bidirectional Long-Short Term Memory) والنظام المتخصص (Expert System) للتنبؤ بنوايا المرضى بناءً على السؤال أو الطلب الذي يطرحونه ، ويقدم الهدف إجابة مفيدة. النظام الممثل في تطبيق أندرويد (Android) متصل بقاعدة بيانات المستشفى.

الكلمات المفتاحية : المتحدث الآلي ، *الذكاء الإصطناعي ، المساعد الطبي، التعلم العميق ، الذاكرة الثنائية الاتجاه طويلة و* قصيرة *المدى ، النظام* المتخصص

Abstract

The development of Artificial Intelligence (AI) has made great advances in the medical field. With AI, medical professionals and nurses can improve the decision-making process and spend less time collecting patients' information. AI can improve communication in the hospital by supporting guide to patients, especially in the current world crisis of the (COVID-19) pandemic, where the direct communication between patients and medical crew is restricted. These circumstances make many medical assistants worry about their health condition as they need to not get in direct contact with patients. One of the AI technologies that can solve this problem is represented in a medical assistant Chatbot. In this work, we build an intelligent Chatbot system that can interact with patients to assist them in hospital visits. Combining Deep Learning algorithms (Bidirectional Long-Short Term Memory model) and an Expert system to predict patient's intentions based on the question or request they ask and the goal provide a helpful answer. The system is represented in an Android Chatbot Application connected to a hospital knowledge base.

Keywords: *Chatbot, Artificial Intelligence, Medical Assistant, Deep Learning, Bidirectional Long-Short Term Memory, Expert System.*

Résumé

Le développement de l'intelligence artificielle (IA) a fait un grand progrès dans le domaine médical. Avec l'IA, les professionnels de la santé et les infirmières peuvent améliorer le processus de prise de décision et passer moins de temps à essayer de collecter des informations sur les patients. L'IA peut améliorer la communication à l'hôpital en soutenant le guide des patients, en particulier dans la crise mondiale actuelle de la pandémie (COVID-19), où la communication directe entre les patients et l'équipe médicale est restreinte. Ces circonstances font que beaucoup d'assistants médicaux s'inquiètent de leur état de santé car ils n'ont pas besoin d'entrer en contact direct avec les patients. L'une des technologies d'IA pouvant résoudre ce problème est représentée dans un assistant médical Chatbot. Dans ce travail, nous construisons un système Chatbot intelligent qui peut interagir avec le patient pour l'assister lors des visites à l'hôpital. Combiner des algorithmes d'apprentissage profond (modèle de mémoire bidirectionnelle à long-court terme) et un système expert pour prédire les intentions des patients en fonction de la question ou de la demande qu'ils posent, et l'objectif fournit une réponse utile. Le système représenté dans une application Chatbot Android connectée à une base de connaissances hospitalière.

Mots clés : *Chatbot , Intelligence artificielle, Assistant médical, L'apprentissage en profondeur, mémoire bidirectionnelle à long-court terme, Système expert*

List of Figures

List of source code

List of Tables

Contents

Chapter 1

General Introduction

1.1 General context

The medical domain is a critical section for human beings to take care of their health condition. Medical society always looks for new ways to improve the means that help medical crew assist them in daily routine operations. One of these is Artificial Intelligence (AI); this technology impacts the medical domain in many aspects: diagnosis, treatment, assistant, etc... A great deal of research in artificial intelligence was implemented in the medical field in the last decades, such as cancer prediction, chronic diseases tracker, radiology classifier. In addition, AI technologies provide numerous opportunities for innovation in the healthcare industry, such technologies like Chatbots can be developed to help and support both patients and medical crew.

In recent years, a huge amount of investment has been made to develop intelligent personal assistants (IPAs) such as Apple's Siri, Microsoft's Cortana, Google Assistant, Facebook Messenger, and Amazon's Alexa. These IPAs can be deployed on mobile devices to answer user's questions or respond to a request by providing in-time assistance such as reminding them of upcoming events or recommending a useful service Shum et al. [2018]. The huge development in artificial intelligence and a major shift from online social networks to mobile messaging applications such as Facebook Messenger, Telegram, Slack, Kik, and Viber led to the sudden interest in chatbots Brandtzaeg and Følstad [2018].

1.2 Problematic and Objectives

The COVID-19 pandemic has radically changed the world. Since the appearance of COVID-19 pandemic, the World Health Organization recommend to nations some precautions. One of these precautions is humans should not get in touch directly with each other to avoid the infects between human beings. To ensure communication in the hospital robot represents the ideal solution to this issue. Besides, handypersons are needed in this situation due to its short-handed when we talk about crew numbers. The answer to the problem, as mentioned earlier, is to use robots. In the literature, many works have used a real robot to facilitate workaround infected patients. However, some workers need to communicate with persons who can also be affected or visit the hospital for emergency cases for the reception desk. The main objective of this project is to create a virtual robot represented in Chatbot to communicate with patients and assist them in their routine needs. We propose a solution based on Deep learning algorithms to deal with the mentioned problem.

1.3 Outlines

Our work is organized as follows:

In the 2nd chapter, we talk about the literature review of Chatbot technology. We start with the Human-Computer interaction field, the case-based reasoning, and its limits. We defined the chatbot and its types and general architecture, followed by the problem statement.

In the 3rd chapter, we discuss the research methods and technologies of the chatbot and deep learning, and we start by defining Natural Language Processing and its history and methods. Finally, we discuss some related works about the chatbot projects and the used methods.

The 4th chapter is mainly about Design and Contribution. We present our proposed architecture for medical assistant chatbots and the used methods. The proposed architecture is explained along with a flowchart and UML diagram and the used deep learning algorithm. Implementation and results are discussed.

In the 5th chapter, We present the development tools and frameworks, the chatbot answer process, with screenshots of the application. Then we discuss and analyze the obtained results.

Finally, we conclude our work in the 6th chapter.

Chapter 2

Chatbot : Literature review

2.1 Introduction

The conversation is a human art that we practice every day, from discussing events or debating food options or deciding the best movie to watch after dinner... etc. For many years, AI researchers are interested in making an AI system that can interact the same way as humans can, asking and answering a wide range of questions, understanding and producing human language and intentions. So far, researchers have succeeded in building a specialized Human-like intelligent system that can perform intelligent tasks based on a natural language request like booking a flight. These Human-like AI systems are represented in a Human-Computer Interaction technology called Chatbot.

Many intelligent personal assistants (IPAs), like the mentioned in the general introduction, provide reactive and proactive assistance to users to accomplish a variety of tasks; for example, reactive assistance includes information consumption such as weather reports. Task assistance such as restaurant reservation and proactive assistance includes reminding the user of upcoming events or recommending specific products or services to the user, according to the user's profile and relevant contextual information Shum et al. [2018].

2.2 Human-computer Interaction

The development of new information technologies such as the Internet of things, cloud computing, big data, remote sensing telemetry, and geographic information systems led to the growth of the research of urban intelligence. Human-computer Interaction is a broadly researched field and studied with a long history as a front-end application technology in urban intelligence Qi et al. [2019], One of the major factors in the development of information and communication technologies and artificial intelligence is the advancement of effective tools of the interaction between man (user, operator, developer) and computer systems Karpov and Yusupov [2018].

2.2.1 Definition

Booth [2014] defined Human-computer interaction as : "*The study of the interaction between humans and computers*", According to Almansor and Hussain [2020] Human-computer interaction is the field of understanding the interaction between humans and machines, dialogue systems or conversational systems, including chatbots, voice control interfaces, and personal assistants are examples of HCI application that have been developed to interact with users using natural language.

2.2.2 Types

There are several types of HCI, and we can separate them by the mean of the interaction.

- **Contact interaction**

  Type of interaction using Keyboards and button bars Manipulators, including a mouse, a joystick, a trackball Touch screens, and touch panels Virtual-reality gloves and costumes Karpov and Yusupov [2018].

- **Textual and handwriting input**

  This type of interaction is generally done using Optical Character Recognition (OCR) which is mainly used for the electronic or mechanical transform of printed, handwritten, or typed characters into machine-encoded text. The process generates A text document as

output by recognizing the characters in a picture or scanned document, and it can be edited and manipulated by the computer program Manoharan [2019].

- **Speech interface**

  Speech interfaces are communication or interaction systems that use speech, and they can be pre-recorded or synthesized to communicate or interact with users Weinschenk and Barker [2000]. Interacting with automatic systems through Speech has become the dominant method of interacting. Voice-enabled intelligent personal assistants (IPAs) that mentioned previously are widely available on many devices that use speech as a main form of interaction, speech can be used as a primary input or output, or in a two-way dialogue, with the help of natural language understanding (NLU) and automatic speech recognition (ASR) user input can be recognized and identify the intents or commands Clark et al. [2019].

- **Gesture interface and behavior analysis**

  Gesture recognition the term collectively refers to the whole process of tracking human gestures to their representation and conversion to semantically meaningful commands, designing and developing systems that can identify explicit human gestures as input, and process these gesture representations for device control through mapping of commands as output are aided through two major types of enabling technologies for human-computer interaction namely contact-based and vision-based devices Rautaray and Agrawal [2015].

- **Facial expression interface**

  In this type of interaction, the most common method is through facial imaging using cameras. However, muscle contractions led to changes in facial features, detectable at 1000 times per second from electromyography (EMG) electrodes, in comparison with cameras with a sample at 30-60 frames per second. EMG can detect baseline muscle tone changes and the monitoring of micro-expressions that may not be observable. Indeed, EMG was used to calibrate camera-based facial expression algorithms Mavridou et al. [2017].

- **Neural Computer Interface**

  A person with a neural disorder leads to a locked-in state (LIS). These diseases affect the motor neuron in the brain and spinal cord and make them inoperative and lose their ability to begin action potentials to manage voluntary body movements. NeuralComputer Interface is a type of interaction used to convert the control signals of the human brain neuron into commands to make the communication in a more natural and human-friendly way in the absence of biological channels. Fang et al. [2019]

2.2.3 Case-based reasoning

Rabah et al. [2019] defined Case-based reasoning as: "*Case-based reasoning (CBR) is a paradigm of artificial intelligence and cognitive science that models the reasoning process as primarily memory-based. It uses the specific knowledge of past experiences, formalized in the form of concrete problem situations called cases*". The case-based Reasoning understanding intelligence approach is based on two tenets: the nature of the world problems tend to repeat, and similar problems have similar solutions. When the two tenets are required, Case-Based Reasoning is an effective reasoning strategy. Case-Based Reasoning is both a theory for modeling how people use memory to solve problems and a theory of how we can design machines that use past experiences to address new situations Goel and Diaz-Agudo [2017].

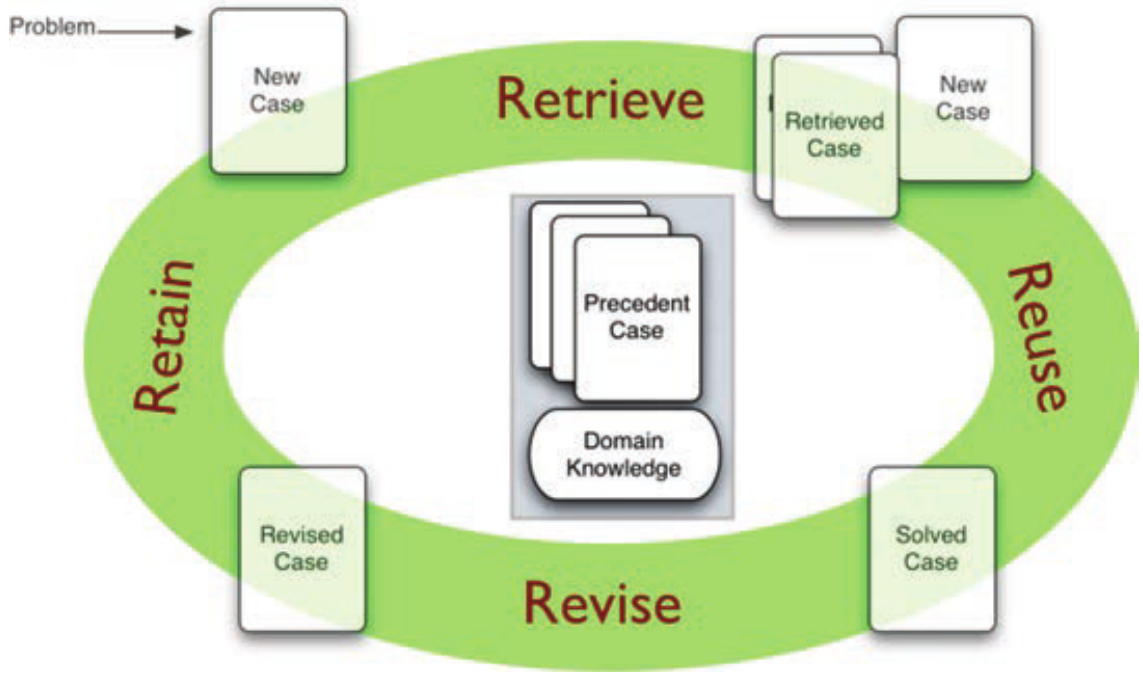


Figure 2.1: The Case Based Reasoning Cycle from Quijano-Sanchez et al. [2013]

As Figure 2.1 shows, Case Based Reasoning follows a cycle of four phases:

1. **Retrieve** from the case base the old cases that are the most similar to the query.
2. **Reuse** the information and knowledge embedded within similar resolved cases to produce a solution for the query case.
3. **Revise** the solution to adapt it to the query case.
4. **Retain** the query case with the selected solution in the case database.

Case Based Reasoning has been applied in many domains, including medicine. In a medical Case Based Reasoning system, a case usually corresponds to a patient and the problem to solve typically consists of classifying a new patient according to various classes. For example in a diagnostic system, targeting a given disorder, there are commonly two classes healthy and diseased. The case database contains previous patients for which the diagnosis or the treatment is known Lamy et al. [2019].

2.2.4 Limits

In the CBR system, there are no methods that suit every domain application, so there are several shortcomings and limits in this systems:

- Knowledge modeling is one of the CBR limits. Building initial knowledge modeling and the similarity measures can be complicated, especially on domains that require expert knowledge of the used domain Berka [2020].
- The Case Based Reasoning is heavily dependent on the content and structure of its set of cases. Sometimes fast access to the cases is required, which is impossible when dealing with a huge knowledge base Cordier et al. [2009].
- • When dealing with natural language problems, Case Based Reasoning suffers from knowledge representation. The human language can be represented in many different terms for the same situation, and the syntax of the input text may not be correct, or other languages are used D'Ávila et al. [2018].

2.3 New era of Human computer interaction

For many years, Human Computer Interaction researchers and practitioners working to improve the design and the interaction of the graphical user interfaces in a natural friendly way. Studies is now focused on designing and building more simpler and easy to interact user interfaces. The interaction with digital systems happens not through swiping, scrolling, or button clicks. Still, it happens through a new means of exchange represented in a natural language way, where the computer processes and understands human language. This interaction mainly appears in the recent developments of Chatbots Følstad and Brandtzæg [2017].

2.3.1 Chatbot

This section introduces a new concept of communication between machines and human beings, consisting of Chatbot technology.

Definition :

There are many definitions of chatbot. Smutny and Schreiberova [2020] defined it as : "*a software tool that interacts with users on a certain topic or in a specific domain in a natural, conversational way using text and voice*". According to the dictionary, a chatbot is "*A computer program designed to simulate conversation with human users, especially over the Internet*" Chatbot in English by Lexic Dictionaries [2019]. Chatbots are designed to provide a clone of how a human will chat, and thereby, it acts as a conversational partner rather than humans Divya et al. [2018]. Furthermore, popular Internet companies such as Google, Facebook, and Microsoft see chatbots as the next dominating technology Brandtzaeg and Følstad [2017].

Brief History of chatbots :

Chatbots have a long history starting from 1964. When the first Chatbot ELIZA was created. ELIZA analyzed the user's inputs and created its response based on the reuse rules methods associated with a decomposition of the input. After thirty years, the Chatbot ALICE was developedusingArtificialIntelligenceMarkupLanguage, themainmethodisusingCategorypatterns matched to find the most appropriate response to user input is based on categories containing patterns and a template in the response. Later in 2008, and unlike previous Chatbots, CLEVERBOT was launched. In addition, its answers were not predefined; instead, it learned directly from human input, where a user would type in a desired comment or question, and CLEVERBOT would find all keywords and entities matching the input. Nowadays, chatbots can understand a wide range of user questions and cases and guide the user in the desired direction Winkler and Soellner [2018].

2.3.2 Types of Chatbots

In this subsection, we introduce some types of Chatbots, that are differentiated by the method of processing.

1. **Menu/Button-Based Chatbots**

   The Menu/Button-Based Chatbots are the most basic and the simplest type of Chatbots existing today, which are presented in a decision tree hierarchies to the user in the form

of buttons. Like the automated phone menus that we interact with daily, where you make your decisions to get the perfect answers. The user is instructed to make these decisions by choosing their options and going deeper toward the AI's appropriate response. However, these menu-based Chatbots are the slowest in terms of getting the user to their desired answer, and they fall close in more advanced scenarios, in which there are too many variables or too much knowledge to predict, and how users should get to specific answers with confidence Gupta et al. [2020].

2. **Keyword recognition-based Chatbots**

   Unlike the previous Menu/Button-Based Chatbots, keyword recognition-based Chatbots recognizes specific keywords to produce the desired result. They listen to what the users type and respond appropriately. These Chatbots use a customized keywords list and an AI application to respond appropriately to the user. These types of Chatbots start to fail when they have to answer many similar questions, and there are keyword redundancies between several related questions. It is quite common to see Chatbots that are a hybrid of keyword recognition-based and menu-based. These Chatbots provide users with the choice to try to ask their questions directly or use the Chatbot's menu buttons if the keyword recognition functionality is giving poor results or the user requires some guidance to find their answer Gupta et al. [2020].

3. **Linguistic Based Chatbots (Rule-based Chatbots)**

   Linguistic or Rule-Based Chatbots, create informal chat using if/then logic. This type follows the defined language conditions of the Chatbot. Conditions can be created to evaluate the words and their order, synonyms, and more. If the incoming input matches the conditions defined by the Chatbot, users can receive the appropriate help in a short amount of time. However, the permutation and combination of each question must be defined, otherwise, the Chatbot will not understand the user's input. This is why a Rule-Based models like those discussed previously are limited. Even while incredibly common, they can be slow to develop. These Chatbots demand the ability to change and specificity D'Ávila et al. [2018].

4. **Contextual Chatbots (Machine Learning Chatbots)**

   Contextual Chatbots or Machine learning Chatbots are the most advanced of the three types discussed previously. These types of Chatbots use Machine Learning (ML) and Artificial Intelligence (AI) technologies such as Natural Language Processing, voice recognition, speech-to-text, Machine Translation, conversion algorithms .. etc. To remember, conversations state with specific users to learn and grow over time. The main idea of this type of bot is to figure out what the user's intentions are and correspondingly present a reasonable answer by analyzing the pattern in the database Gupta et al. [2020].

5. **Voice bots**

   Voice bots make conversational interfaces even more vernacular and easy to interact. Voice bots have been on the rise for the last couple of years. Major companies are now using voice-based Chatbots as daily assistant applications, such as Apple's Siri, Amazon's Alexa, Google assistant... etc. Due to their convenience, it's much easier for the user to speak rather than type. Voice bots bring enjoyable experiences directly to the user's Braun and Matthes [2019].

2.3.3 Architecture

Designing and developing a new generation AI-based Chatbot system requires a variety of preprocessing techniques such as (natural language processing, language generation, etc...) Ahmad et al. [2018],], to build accurate and meaningful Chatbot systems, developers need to understand what the Chatbot will offer and what category falls into to pick the right algorithms or platforms and tools. At the same time, it also helps the end-users to understand what to expect Nimavat and Champaneria [2017]. The first step in designing any Chatbot system is to divide it into parts according to a standard to make the development step much easier Ramesh et al. [2017]. Figure 2.2 represents a general Chatbot architecture.

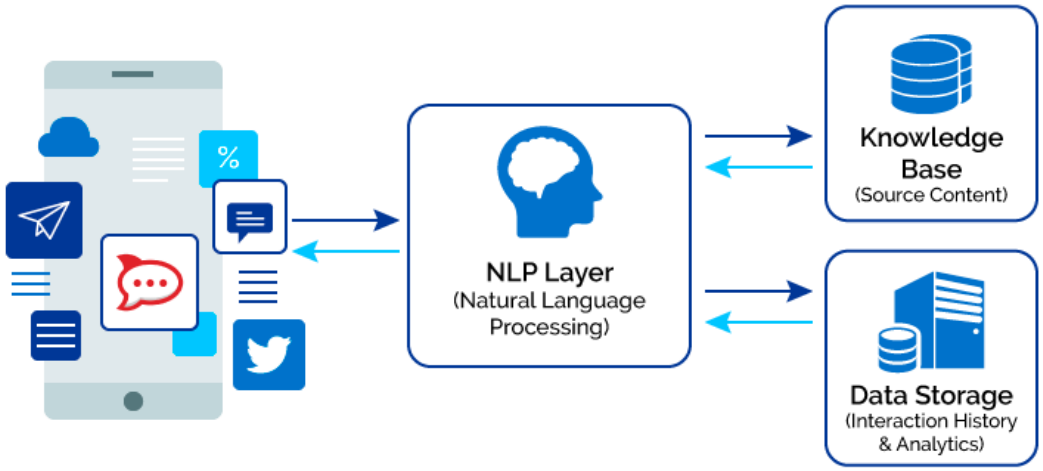


Figure 2.2: General Chatbot Architecture from Dilmegani [2021]

As Kucherbaev et al. [2018] described, the process of a general Chatbot architecture follows six steps:

1. **Getting** the user's input from the interaction interface.
2. **Natural Language Processing components** receive the input and parse it to infer the user's intention and the associated information, using Natural language processing methods.
3. **Action execution and information retrieval** request.
4. **Perform** the requested actions or retrieves the data of interest from its data sources, which may be a database, known as the Knowledge base of the chatbot, or external resources that are accessed through an API call
5. **Updates** the context of the conversation of the user intentions in the Dialogue Manager (also called Data Storage analyzer).
6. **Display** the Chatbot answer or report the performed actions.

2.4 Problem statement: Medical assistance

This section presents the use case of our problem applied as medical assistance, which consists of Coronavirus (COVID-19) Pandemic.

2.4.1 Definition

Since the discovery of the Coronavirus (COVID-19), it has become a global pandemic. The World Health Organization recommends to nations some precautions. One of these precautions is that humans should not get in touch directly to avoid infects between them. It has been a great challenge to hospitals or healthcare staff to manage the flow of the high number of cases, especially in remote areas. It is becoming more difficult to consult a medical specialist when the immediate hit of the epidemic has occurred. Hospital administrators are spending their day in appointment scheduling and answering routine questions of patients, Continuing or repeating the same actions and words is neither necessary nor productive.

2.4.2 Benefits of the Chatbots in the Medical Domain

there are several benefits and advantages of using Chatbots in the medical field.

- Nowadays, more people use smartphones, medical Chatbots help healthcare organizations reach their patients where they spend more time on their smartphones Shahid et al. [2021].
- Patients want more information about their medical conditions and treatments, they might need information about generic and prescription drugs, medical Chatbots have been proven as a useful tools to provide additional information to patients Shifa et al. [2020].
- The Healthcare industry has a significantly small amount of qualified professionals. Therefore, it takes time for patients to receive consultations one by one. Medical Chatbots can provide initial assistance over one by one conversation. Chatbots can do several tasks like reminding patients about their medicines, set up appointments with doctors, find the nearest pharmacy or doctor‘s office etc. Shahid et al. [2021].

2.5 Conclusion

Chatbot as a new Human Computer interaction technology shows a great assist in the medical domain. Case-based reasoning Chatbots became outdated and very hard to implement nowadays; moving toward a better method of building a Chatbot is required. The next chapter will discuss the new method of developing a Chatbot based on the Deep Learning and Natural Language Processing methods and advantages.

Chapter 3

Research methods : Natural language processing

3.1 Introduction

Human language is highly variable and ambiguous, and ever-changing and evolving. Humans are capable of express, recognize and explain very accurate meanings. While humans are great language users, they are also inferior at formally understanding and describing the principles that control the language Goldberg [2017]. Natural language processing is a widely discussed and researched topic nowadays. It is one of the oldest research areas in machine learning; it is used in significant fields such as machine translation, speech recognition and text processing Jain et al. [2018]. Natural language processing has become included in our daily lives, like the widespread of automatic machine translation on the web and social media, prevent our email inboxes from collapsing under a deluge of spam by text classification, the evolution of search engines to a high degree of linguistic, and dialogue systems provide a progressively common and effective way to get and share information. Eisenstein [2019]

3.2 Definition

There are many definitions of Natural language processing (NLP). Deng and Liu [2018] defined it as: "*Natural Language Processing investigates the use of computers to process or understand natural human languages to perform useful tasks*". According to Jain et al. [2018] Natural language

processing is an integral area of computer science in which machine learning and computational linguistics are broadly used. Natural Language Processing came into existence to relieve the user's work and satisfy the wish to communicate with the computer naturally. Since all the users may not be highly experienced in machine-specific language, Natural Language Processing serves users who do not have enough time to learn new languages Khurana et al. [2017]. Making human and computer interaction easy and efficient is the primary goal of the field. The machine learns and processes the syntax and meaning of human language to give the user output. Natural Language Processing involves making computer systems perform meaningful tasks with the natural and human understandable language Jain et al. [2018]. Natural language processing is usually used mutually with AI and Machine Learning (ML). However, they are different from each other. Natural language processing and Machine Learning are parts of AI. Figure 3.1 shows the relation between Natural language processing (NLP), Natural Language Understanding (NLU), Natural Language Generation (NLG), Machine Learning(ML) and AI. Gersil and Hilal [2020]

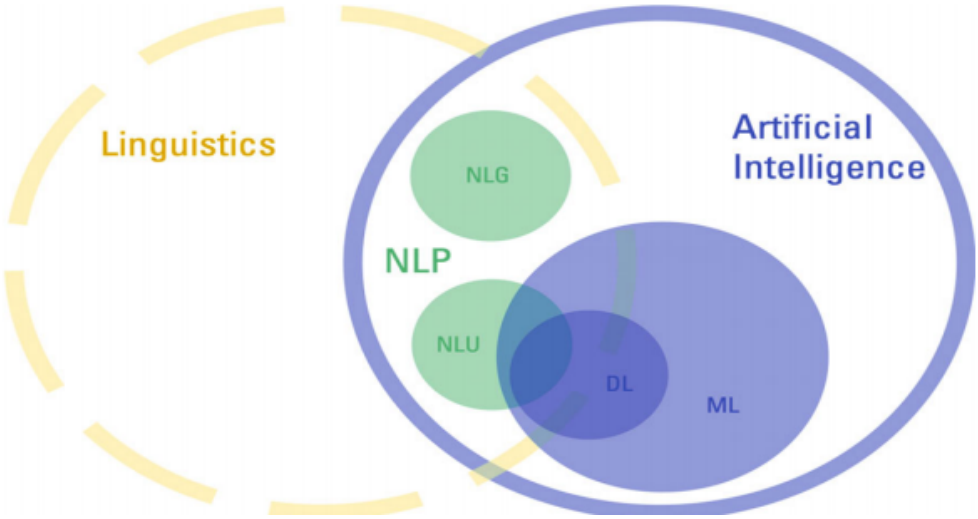


Figure 3.1: Representation of AI, ML and NLP from Gersil and Hilal [2020]

3.2.1 Machine Learning and Deep learning :

As a subcategory of AI, machine learning is quickly becoming ordinary in many of the applications we use daily. It can make us more productive, helping in decisions making, provide a per-

sonalized experience, and gain insights about the world by leveraging data. The AI field is broad, including significant application domains, such as computer vision, natural language processing, Big Data analysis, search algorithms, and many other areas Kamath et al. [2019]. Deep learning has recently been highly successful as a subset field of machine learning. Deep learning can be defined as neural networks with a large number of parameters and layers. These layers outperform and express more complex models than the traditional Neural Networks (used in machine learning). One of the most significant advantages of Deep learning is the ability to build next-generation neural networks capable of extracting features for themselves in a more intelligent way. However, More connections mean that the networks have more parameters to optimize, and this required extensive data and an explosion in computing power which is not always available Patterson and Gibson [2017].

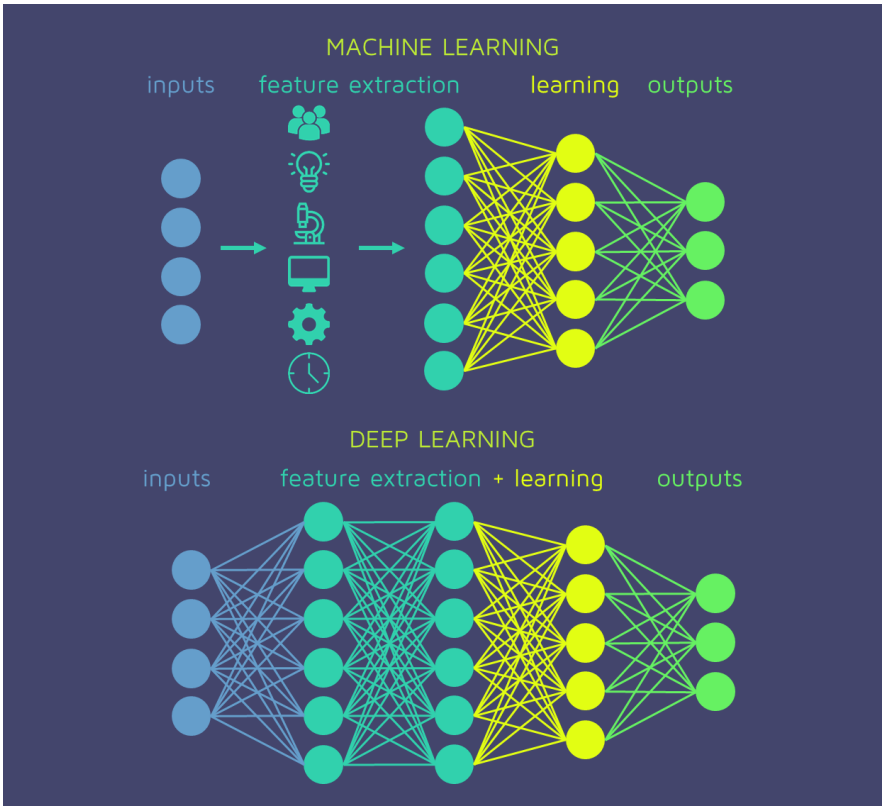


Figure 3.2: The automatic feature extraction in Deep learning over Machine learning

3.2.2 Natural Language Understanding :

Natural language understanding (NLU) is a subfield of Natural Language Processing, which involves transforming human language into a machine-readable format, which helps computers understand and interpret human language by breaking down the essential pieces of speech. While speech recognition records spoken language in real-time, converted into written form and returns text, NLU goes beyond recognition to determine a user's intent. Speech recognition is operated by statistical machine learning methods, which add numeric structure to large datasets. In NLU, machine learning models improve over time as they learn to understand language patterns, recognize syntax, context, sentiment, unique definitions and user intents Liu et al. [2019].

3.2.3 Natural Language Generation

Natural language generation(NLG)is a technique that uses raw structured data to convert it into plain English or any other desired language. It's also called data storytelling. This technique is beneficial in many organizations where a large amount of data is used. It converts structured data into natural languages to better represent patterns or detailed insights into any domain. Gatt and Krahmer [2018]

3.3 History : Birth of NLP

In the late of 1940s the work regarding machine translation (MT) and linguistic and the use of computers for literary studies had started Khurana et al. [2017]. NLP research lasted for a long time, dating back to 1950s , in 1950 Alan Turing proposed the Turing test to evaluate a computer's ability to exhibit intelligent behavior not able to be identified as different or distinct from that of humans. The test was based on natural language conversations between a human and a computer designed to generate human-like responses Johri et al. [2020].

Since 1960, researchers focused on solving the problems of representing meaning and developing computationally tractable solutions that when the existing theories of grammar were not able to produce before 1960 Joseph et al. [2016]. This period coincided approximately with

the early development of artificial intelligence Deng and Liu [2018] , characterized by expert knowledge engineering where domain experts devised computer programs according to the knowledge about the application domains they have. the first expert system was created was in 1970, the main algorithm used was the inference rules in the form of "if-then-else" Deng and Liu [2018].

By the 1970s, the first NLP computer program was developed by Terry Winograd at Massachusetts Institute of Technology (MIT) named SHRDLU, this program was able to perform the tasks like remembering names, and moving objects, determining the current state , it was successful AI achievement Johri et al. [2020].

In early 1980s computational grammar theory became a very active area of research linked with logic's for meaning and knowledge's ability to deal with the user's beliefs and intentions and with functions like emphasis and themes. Johri et al. [2020].

Since 1990s. New era of NLP characterised by the new paradigm of deep learning Deng and Liu [2018], deep learning brings hope for addressing the human feature engineering problems. The big success of speech recognition in 20102011 led to new era of NLP and artificial intelligence, Google announced the first of its move to neural machine translation in September 2016 and Microsoft made similar announcement 2 months later, Facebook had full deployment by August 2017 Deng and Liu [2018].

3.4 Methods

In this section we present the most popular Natural Language Processing methods.

- **Tokenization :**

  Tokenization is the task of separating a given character sequence with a defined document unit into parts, called tokens, and throwing away some of the unnecessary characters such as punctuation. These tokens are often loosely mentioned as terms or words, in the form a type/token distinction. A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. A type is the class of all tokens containing the same character sequence Solangi et al. [2018]. in Figure 3.3 below, we present a Tokenization method example.

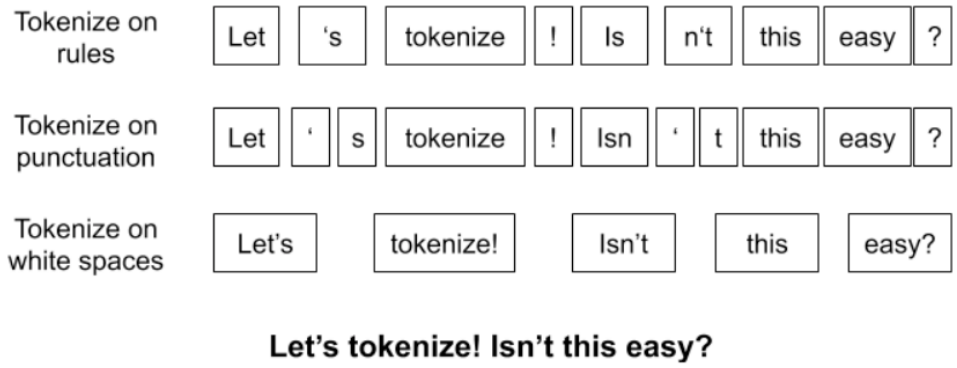


Figure 3.3: Tokenization method example

- **Text Normalization (Stemming and Lemmatization) :**

  Stemming and Lemmatization are Text Normalization or Word Normalization techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing. Stemming has been the most widely applied morphological technique for information retrieval. The method of Stemming is used to remove affixes from the word for the purpose of extracting the base form of it. With stemming, the searcher does not need to worry about the correct truncation point of search keys. Stemming also reduces the entire number of distinct index entries Solangi et al. [2018].

  With the use of a vocabulary and morphological analysis of words, Lemmatization is the process of removing inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. The goal of both stemming and Lemmatization is to scale back inflectional forms and sometimes derivationally related sorts of a word to a standard base form Solangi et al. [2018]. Figure 3.4 shows the Stemming and Lemmatization comparison.

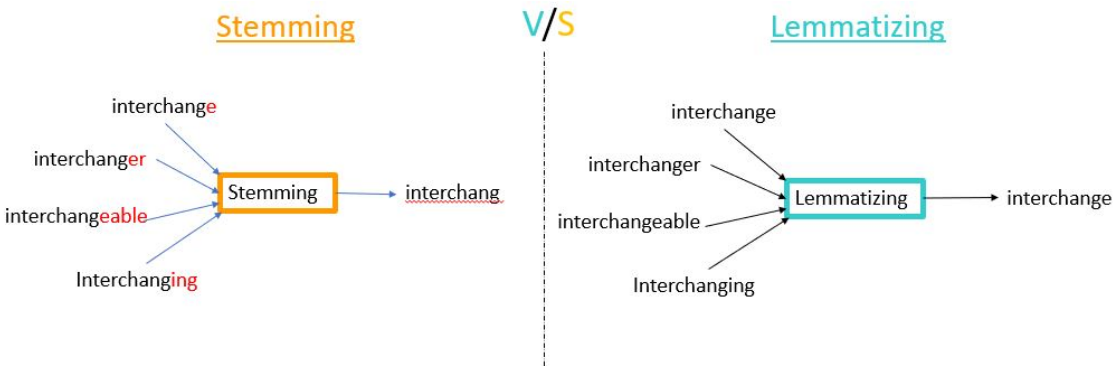


Figure 3.4: Stemming vs Lemmatization example from Solangi et al. [2018]

- **Named Entity Recognition :**

  Named entity recognition (NER), also known as entity chunking or extraction , is a popular technique used in information extraction to identify and segment the named entities and classify or categorize them under various predefined classes.Named Entity Recognition is commonly formulated as a word level tagging problem where each word in the sentence is mapped to a named entity tag. A common method is is to slide a window over each word position to extract features for a classifier that produces tags, moreover, one can allow the classifications at adjacent positions to interact by chaining local classifiers together and perform joint inference to achieve good performance Kuru et al. [2016]. Figure 3.6 shows an example of this method.

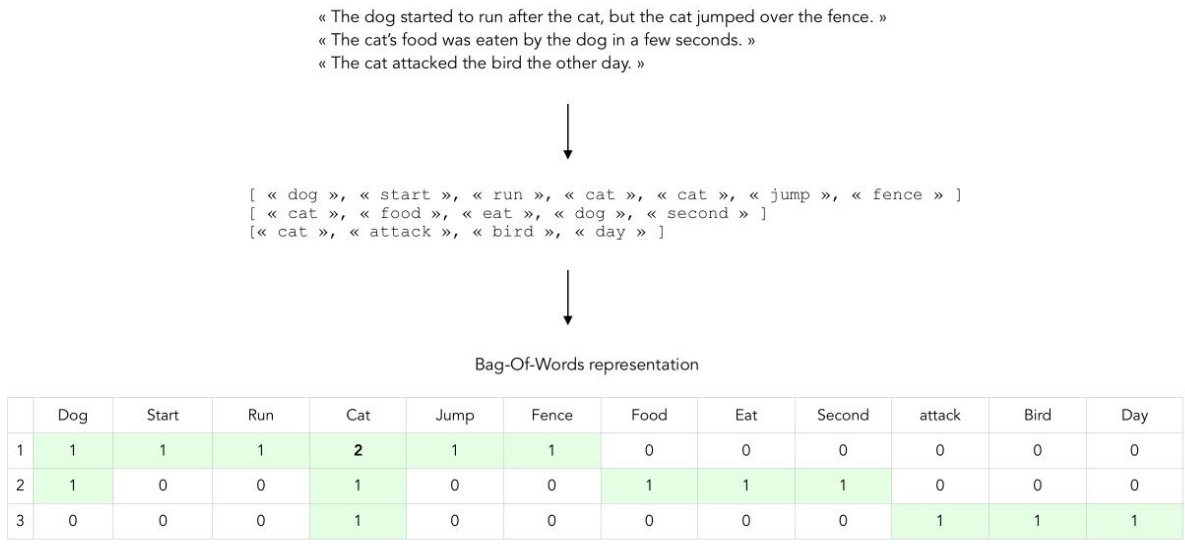
```
[(US, 'GPE'), (China, 'GPE'), (US, 'GPE'), (China, 'GPE'),
 (Sunway, 'ORG'), (TaihuLight, 'ORG'), (200,000, 'CARDINAL'),
 (second, 'ORDINAL'), (Sunway, 'ORG'), (TaihuLight, 'ORG'),
 (93,000, 'CARDINAL'), (4,608, 'CARDINAL'), (two, 'CARDINAL')]
```

US GPE unveils world's most powerful supercomputer, beats China GPE . The US GPE has unveiled the world's most powerful supercomputer called 'Summit', beating the previous record-holder China GPE 's Sunway TaihuLight ORG . With a peak performance of 200,000 CARDINAL trillion calculations per second ORDINAL , it is over twice as fast as Sunway TaihuLight ORG , which is capable of 93,000 CARDINAL trillion calculations per second. Summit has 4,608 CARDINAL servers, which reportedly take up the size of two CARDINAL tennis courts.

Figure 3.5: Visualizing named entities in a news article from Kuru et al. [2016]

- **Bag of Words :**

  Bag of words technique is used to preprocess text for the purpose of using it in Machine Learning modeling. The Bag of words model is mainly used as a tool of feature generation. After transforming the text into a "bag of words", various measures to characterize the text can be calculate, It is a representation of any text that explains the occurrence of the words within a document. Its called “Bag” due to its mechanism, it is only concerned with whether known words occur in the document, not the location of the words El-Din [2016].



	Dog	Start	Run	Cat	Jump	Fence	Food	Eat	Second	attack	Bird	Day
1	1	1	1	**2**	1	1	0	0	0	0	0	0
2	1	0	0	1	0	0	1	1	1	0	0	0
3	0	0	0	1	0	0	0	0	0	1	1	1

Figure 3.6: Bag of words method example from Walkowiak et al. [2018]

- **One-Hot Encoding :**

  One-Hot Encoding is the most generally used coding scheme. It compares each level of the specific variable to a detailed and fast reference level. A one hot encoding is the representation of the categorical inputs as a binary format sequence vectors. This first requires that the categorical values be integer representation values. Then, each integer value is represented as a binary value in the sequence vector which is all zero values except the index of the integer of the category, that is marked with. One-hot encoding allows the representation of categorical data to be more expressive. Many machine learning algorithms

cannot process categorical data directly. The categories must be converted into numbers. This is required for both input and output variables that are categorical Potdar et al. [2017]. Figure 3.7 show an example of the One-Hot encoding method.

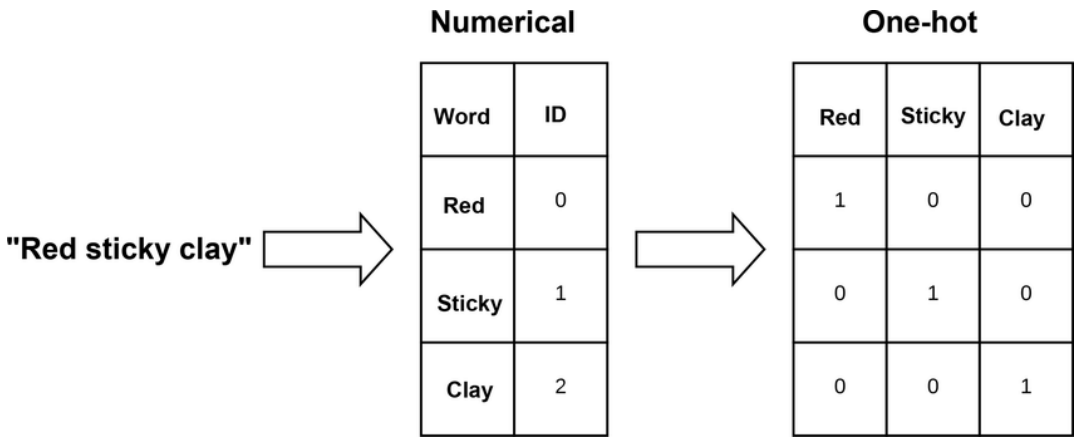


Figure 3.7: One-Hot encoding example from Padarian and Fuentes [2019]

3.5 Related works

Authors in Athota et al. [2020] developed an application to provide quality of answers in a short period of time. Using an expert system. The project is developed for the user to save their time in consulting the doctors or experts for the healthcare solution.Using the N-gram and TF-IDF algorithms in the application for extracting the keyword from the user query. The keywords are weighed down to get the proper answer for the query. They developed a Web interface for the users improved by the latest security technologies and effectiveness upgrades by ensuring user protection and characters and retrieving answers consequently for the questions.

Authors in Ouerhani et al. [2020] developed a smart pervasive chatbot for emergency case assistance based on cloud computing, called SPeCECA (Smart Pervasive Chatbot for Emergency Case Assistance) who is designed to help people react well in emergency cases. They used pre-trained word embedding GloVe model that works well according to their evaluation experiments. Additionally, in order to determine whether it is a true emergency alert or not, they recommend the emotion detection by facial expressions or voice analysis and give EMS more detailed insights and predictions of possible future emergency cases.

Authors in Oguntosin and Olomo [2021] designed a chatbot for Covenant University Shopping Mall. The chatbot's purpose is to have a smart, accurate, and real-time conversation with the students. In this way, students chat with the bot to inquire about particular items they seek to purchase and pay online for the items before they visit the mall. The chatbot is accessible via portable mobile devices or computers, which students can log in to anywhere and anytime on campus, thereby providing a 24-hour online service. They used Seq2seq model for feature extraction and sequence output.

3.6 Synthesis

In this section, we present a comparison between the mentioned work presented in this chapter. This comparison stands on five different parameters which are : domain, objective of the work, platform, the chatbot type and the used methods which are the main point in our work due to their importance especially in the process of decision making.

Table 3.1: Related work comparison

	Athota et al. [2020]	Ouerhani et al. [2020]	Oguntosin and Olomo [2021]	**Our work**
Domain	Medical	Medical	E-Commerce	Medical
Objective	Symptom check and diagnosis	Emergency case assistance	Provide a Covenant University Shopping Mall for students	Medical assistant
Platform	Web application	Android application	Web application	Android application
Chatbot type	Rule-based chatbot	Contextual chatbot	Contextual chatbot	Contextual chatbot
Method	Expert system N-gram and TF-IDF algorithms	word embedding GloVe model	Neural networks Seq2seq model	Bidirectional Long-Short Term Memory (BiLSTM RNN)

3.7 Conclusion

Natural Language Processing is widely discussed and researched topic of Artificial Intelligence and Linguistics and very important process of building a Chatbot system. the methods used in NLP shows extremely efficient way of understand human language. Finding a better combination of deep learning techniques and NLP methods ensures a better understanding Chatbot. Therefore, our work focuses on combining these methods to build an acuurate Chatbot system. In the next chapter we will discuss our system architecture as well as the used deep learning algorithms and the Natural Language Processing methods.

Chapter 4

Design and Contribution

4.1 Introduction

After presenting the Natural Language Processing methods in the previous chapter, our goal now is to build an Android Mobile Chatbot application for medical assistance, using Deep Learning algorithms and Natural Language Processing methods to understand user intents and requests and provide a meaningful response. In this chapter we discuss our system proposed architecture and the Deep learning algorithms and Natural Language Processing methods used to achieve our goal.

4.2 Proposed architecture

The objective of the project is to design the retrieval based model chatbot using recurrent neural networks. In this project our chatbot architecture contain 3 different layers responsible for different tasks.It begins with the user interface layer where the user inputs the intended question or request as a text input. The Treatments layer receives the input, detect user's input language, and translate it into English with Google API translator, then we apply the Natural language processing methods on the input for intents classification and entity extraction. Based on the data extracted, the chatbot response is retrieved from the Data Source layer that contain Medical assistance Prolog answers, the response translated into user's detected language and outputs it on user's interface. The Figure 4.1 below show's our system architecture.

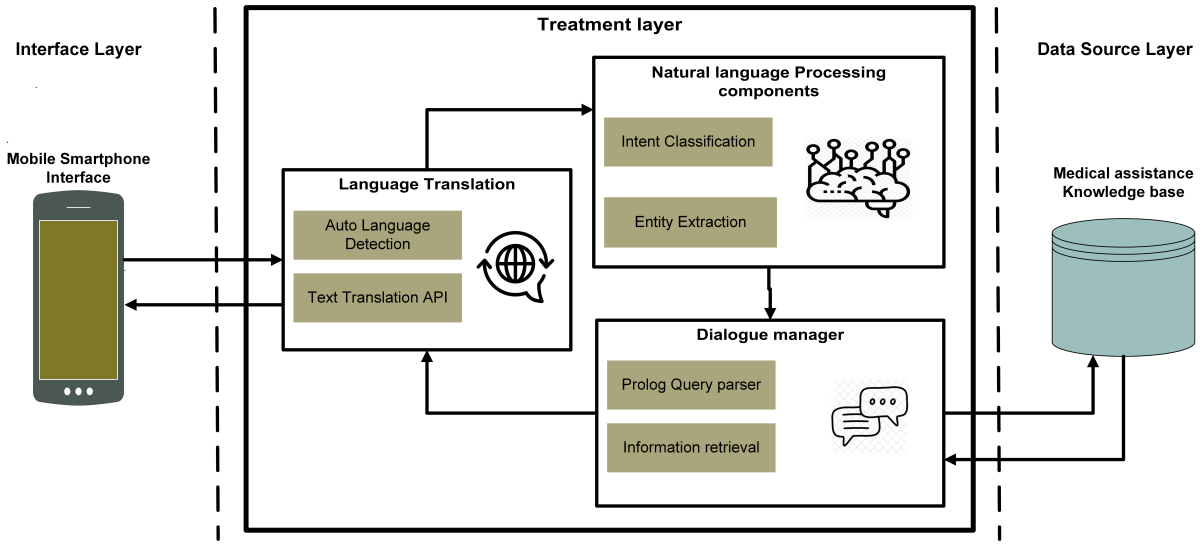


Figure 4.1: The proposed architecture

4.2.1 Architecture description

To understand the role of each component presented in the figure 4.1, this subsection presents the process of each layer.

1. **Interface layer :**

   This layer represents the main way of interaction between the user's and the chatbot.The interface contain 2 main parts, the first one is the input text bar where the user's can input the desired question through the android mobile screen to send it to the Treatment layer, the second part is an ordered swipe window that contain the whole conversation between the user's and the chatbot.

2. **Treatment layer :**

   This layer represents the whole chatbot system as this contain the most functions and the main methods. This layer composed by three different modules.

(a) **Natural language processing components :**

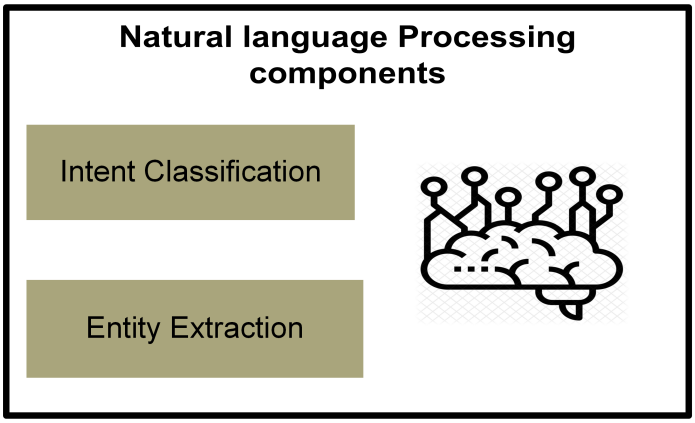


Figure 4.2: Natural language processing components

After receiving the translated text, this Module is responsible for understanding user's intention which consist of 3 tasks, text pre-processing, extracting entities and intents classification,

- **Pre-processing :**

  Lower case, clear and eliminate unwanted symbols such as punctuation's and stop words, and apply tokenization, stemming and One-Hot encoding.

- **Entity extractor :**

  Applying a part-of-speech tagging to the input text to extract entities such as (names, locations, dates, ...)

- **Intent classification :**

  Takes the One-Hot encoded sequence to apply our model prediction for intent classification.

(b) **Dialogue manager :**

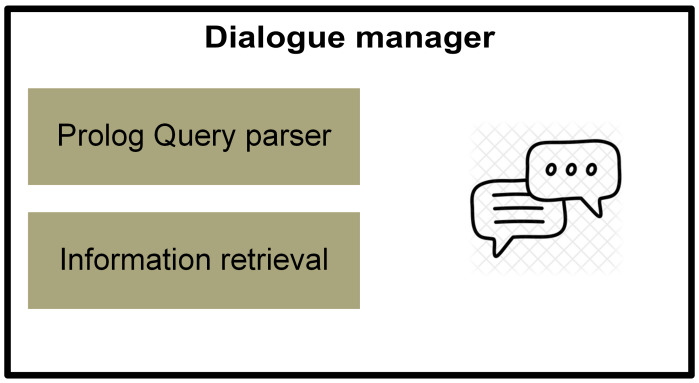


Figure 4.3: Dialogue manager

- **Prolog query parser :**

  This task is responsible for building the Prolog query, to retrieve the chatbot answer based on the intent and entities extracted.

- **Information retrieval :**

  After building the Prolog query, this task responsible for retrieving from the knowledge base the chatbot answer, the answer will be sent to the translation API.

(c) **Language Translation :**

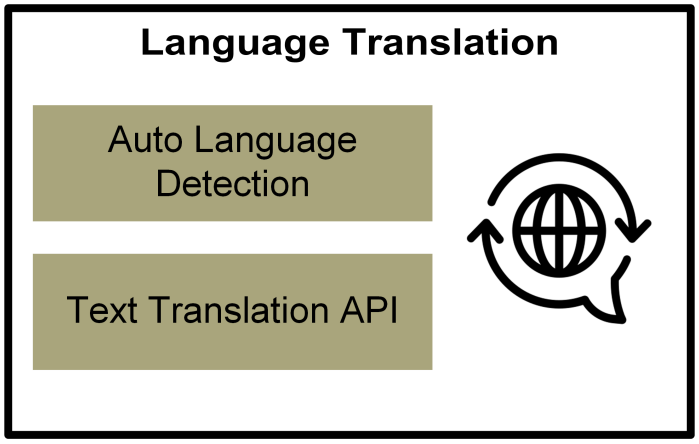


Figure 4.4: Language translation

This Module is responsible for the translating between the user's input and the chatbot system, an auto language detection method is used with three supported lan-

guages (English - Arabic - French). Request Google API translate server to translate from user's detected language to the chatbot system. We use the same tool when translating the chatbot system answer from English to the user's detected language.

3. **Data source layer :** This layer represented in a knowledge base Medical assistance intents based Questions answering Prolog file, with asserted rules and factes.

knowledge base.pl - Notepad

File Edit Format View Help

```
?- assertz(doctor('hakim','emergency room', '1','4')).
?- assertz(doctor('abdelhamid','treatments room', '1','3')).
?- assertz(doctor('missaoui','laboratory','2','11')).
?- assertz(doctor('bacha','administration','1','12')).
?- assertz(room('emergency room','1','4')).
?- assertz(room('treatment room','3','3')).
?- assertz(room('laboratory','2','11')).
?- assertz(room('administration','1','12')).
?- assertz(wait_time('3','10')).
?- assertz(wait_time('5','25')).
?- assertz(wait_time('10','50')).
?- assertz(floor('1')).
?- assertz(floor('2')).
?- assertz(floor('3')).
?- assertz(floor('4')).
?- assertz(room('1')).
?- assertz(room('2')).
?- assertz(room('3')).
?- assertz(room('4')).
?- assertz(room('5')).
?- assertz(room('6')).
?- assertz(room('7')).
?- assertz(room('8')).
?- assertz(room('9')).
?- assertz(room('10')).
?- assertz(room('11')).
?- assertz(room('12')).
```

Figure 4.5: A part of the Knowledge base prolog file

4.2.2 Medical assistance process

To respond the user's request, the proposed chabot system use the following steps presented in the next flowchart (see Figure 4.6).

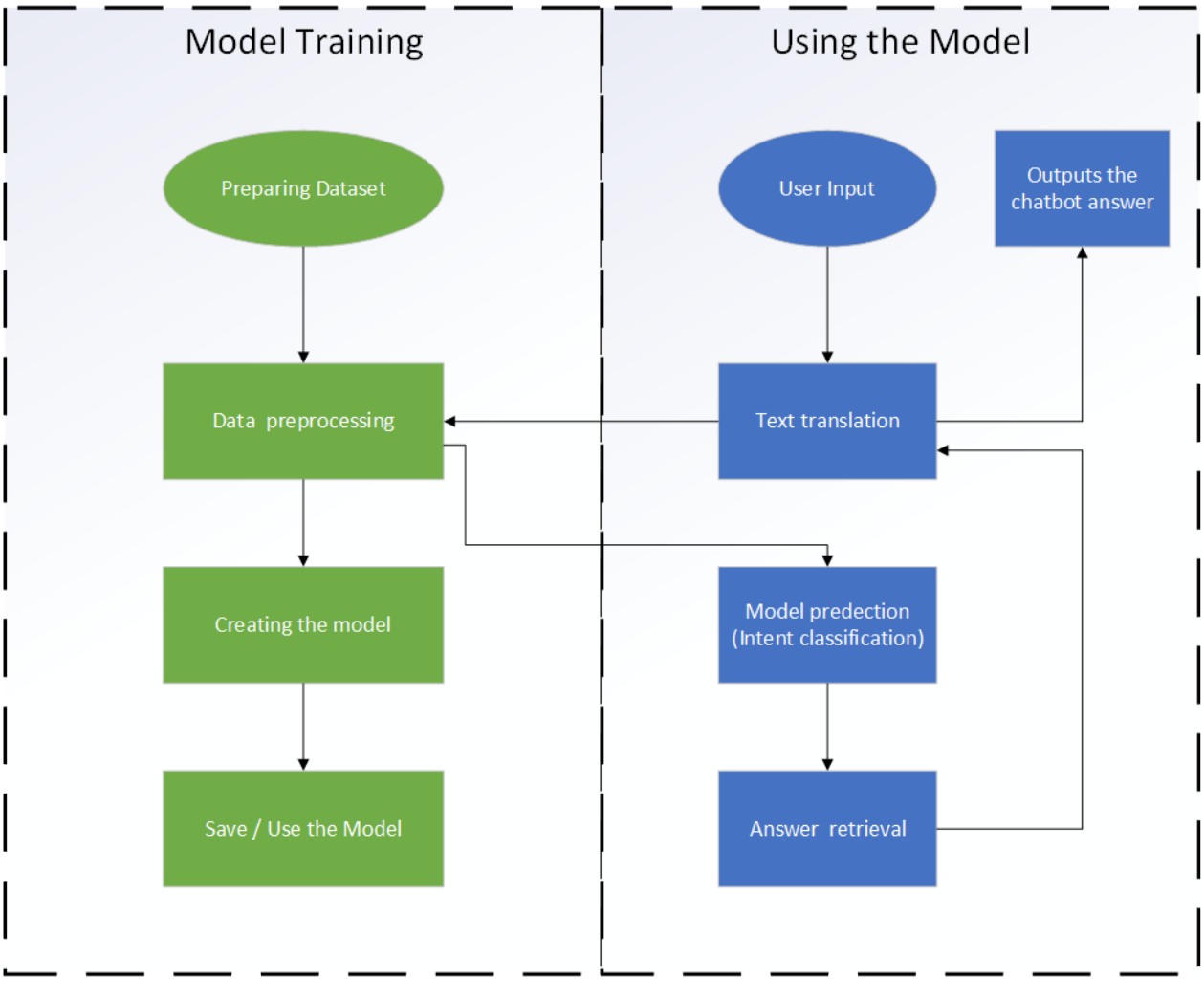


Figure 4.6: Medical assistance process

I. **Model Training :**

(1) **Preparing Dataset:**

Grouping all the dataset from multiple files into One CSV file of two rows : Question and Intention, that can be ready for machine learning process.

(2) **Data Preprocessing:**

before feeding the data to the model, the reliability of a machine learning model is highly dependent upon the quality of the data. therefore it must run through the preprocessing methods, Lower case, punctuation removal, stop words removal, to-

kenization, stemming, applying a one hot encoding to our data to get the integer representation with the same length as vocabulary size (number of unique words) to each sequence.

(3) **Creating the model :**

building the right model is important for creating an accurate learning model. This phase goes on 2 steps:

- configuration of some parameters: number of LSTM cells, number of layers, activation functions...etc.
- Training the model and testing it (making predictions to test the model).

In "Used Algorithm" section, we will discuss the detail of the algorithm used.

(4) **Save/Use the Model :**

After creating the model, it must be exported (saved) to later use for model prediction

II. **Using the model :**

(1) **User's input:**

After loading the user's input from the the application interface which consist of a text data, it will be sent to the local server as to the treatment backend python scripts using FLASK framework.

(2) **Text translation :**

After receiving the user's input, the python langdetect library is used to detect the input language. From the 3 supported languages (English - Arabic - French) the input text is translated to English if the text is not already in English with a request to Google API servers, then it will be sent to Data preprocessing.

(3) **Data preprocessing :**

After loading the translated user's text input, we clear and eliminate unwanted symbols such as punctuation's and stop words. then, we split it into individual words to apply tokenization and part-of-speech tagging to the text data to extract entity's. It goes through the following stages namely lower case, punctuation removal, and stop words removal, tokenization, stemming and, Named entity recognition.

(4) **Model Predection :**

After the preprocessing, the text data will be converted into numerical sequence values (One-Hot encoding) to apply our model prediction. The recognized intent and the extracted entity's will be sent to Dialogue manager for answer retrieval.

(5) **Answer retrieval :**

Build a Prolog query based on the intent and the entities extracted, the query used to retrieve from the knowledge base the chatbot anwer. then it will be sent to translation.

(6) **Translate to user's language :**

Translate to user's detected language the received chatbot answer and send it to the java FLASK loader in the local server as a text data.

(7) **Outputs the chatbot answer :**

After receiving the text data, the chatbot answer will be displayed in the conversation layout in the Android application.

4.2.3 UML Diagram

In this section we illustrate our chatbot functionalities where the user can receive answers. Steps and tasks are shown in the Sequence diagram in Fugure 4.7 below.

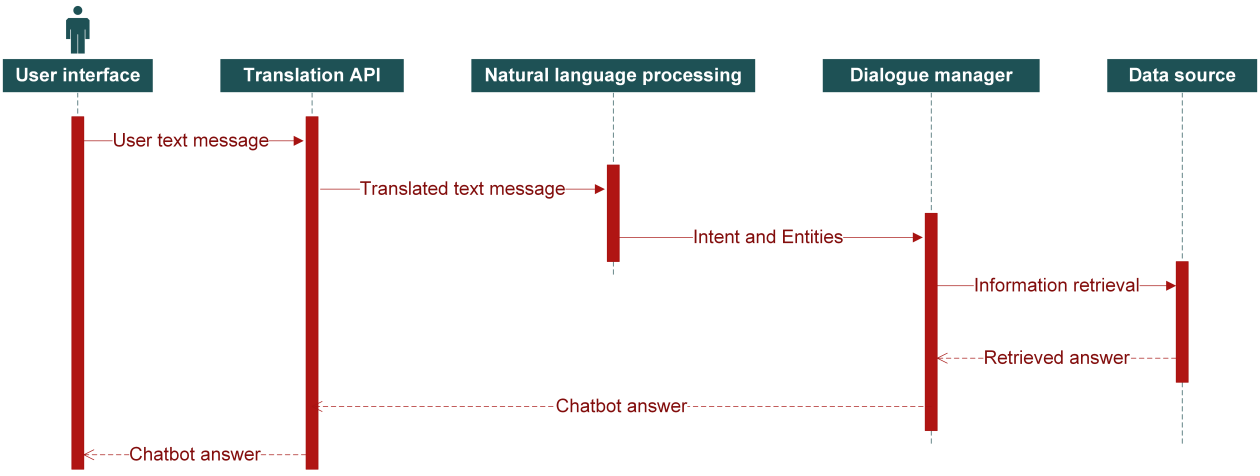


Figure 4.7: Sequence diagram of the chatbot application

4.3 Used Algorithm

4.3.1 Reccurent Neural Network (RNN)

Recurrent Neural Networks (RNN) represent a specific form of Deep learning models which possess several properties that make the neural networks suitable for sequence modeling and natural language processing tasks Goodfellow et al. [2016], they are capable of integrate input from past used events, allowing to output a wide range of sequence to sequence mappings Donkers et al. [2017], The difference between Recurrent Neural Networks and Feedforward Neural Networks also known as Multi-Layer Perceptrons (MLPs) is how information gets passed through the network. While Feedforward Networks pass information through the network without cycles, the RNN has cycles and transmits information back into itself. This enables them to extend the functionality of Feedforward Networks to also take into account previous inputs and not only the current input Schmidt [2019]. Figure 4.8 shows this difference.

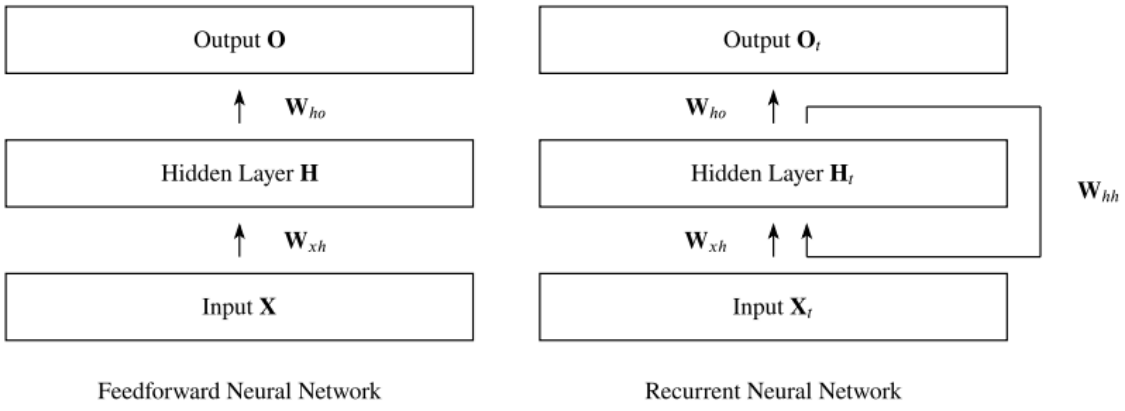


Figure 4.8: The differences between Feedfoward NNs und Recurrent NNs from Schmidt [2019]

4.3.2 Long-Short Term Memory network

RNNs suffers from some problems like the limited range of contextual information and the inability of time back propagation to work properly, also the exploding and vanishing gradients

problems, Long-Short Term Memory (LSTM) are designed to overcome these problems. Yildirim [2018] LSTM is a special type of recurrent neural network (RNN), which each traditional node in the hidden layer is replaced with memory cells. This feature is the major difference compared to standard RNNs. LSTM include memory cells and gate units and memory blocks. The gate units control the states of cells, for example, the input gate is responsible for the input flow to the memory cell for which the output gate for one or more conditions are met operates to control the output stream. This type of architecture is much better at capturing long-term dependencies than regular RNNs. Yildirim et al. [2019]. One of the LSTM versions is called Bidirectional LSTM (BLSTM). This version can improve LSTM model performance in classification processes. Unlike the standard LSTM structure, two different LSTM networks are trained for sequential inputs. Figure 4.9 shows a basic BLSTM structure.

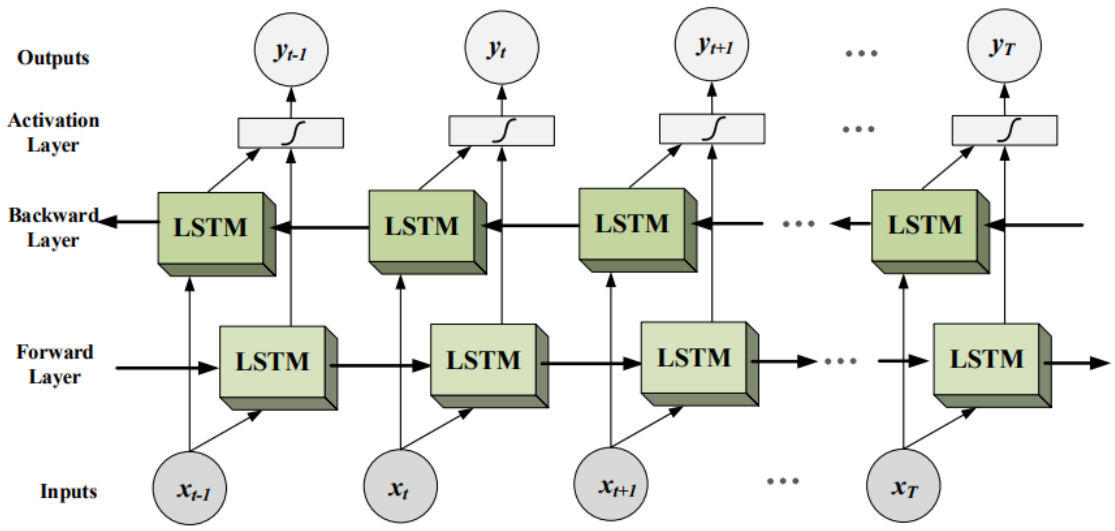


Figure 4.9: Basic structure of the BLSTM network from Yildirim [2018]

In our project , choosing BiLSTM architecture was based on the type of data we have. Since we are dealing with a natural language textual inputs, which means a sequential series, BiLSTM is much better in such scenarios. It is theoretically powerful in language modelling due to its capability of representing a sentence or text with sequence order information. where it can remember some information from previous inputs, and make predection based on multi-

ple outputs, This architecture suit our model for predicting user's intention based on the input sentence.

4.3.3 BiLSTM pseudocode and architecture

Our algorithm consist of several steps described as follows :

1. **Preparing data:** this process is accomplished as explained earlier, using a python script.

1. **Data preprocessing:** before feeding the data to the model, the dataset contains unusual text and symbols that need to be cleaned so that a machine learning model can grasp it, through this steps :

    - **Lower case :** A common approach to lowercasing everything for the sake of simplicity, it helps to maintain the consistency flow during the Natural Language Processing tasks.
    - **Punctuations removal :** Removing every unnecessary punctuation for a better and clean word representation.
    - **Tokenization :** Tokenization as explained in the previous chapter,a module breaks the words into tokens and these words act as an input for the normalization and cleaning process.
    - **Stemming :** There are many variations of words that do not bring any new information and create redundancy, ultimately bringing ambiguity when training machine learning models for predictions. Stemming as mentioned in the previous chapter used to solve this problem.
    - **One-Hot encoding** : As explained in he previous chapter, a one hot encoding allows the representation of categorical data to be more expressive. Our learning algorithm cannot work with categorical data directly. The categories must be converted into numbers. This is required for both input and output variables that are categorical.

1. **Creating the model :** Configuring neural networks is difficult because there is no good theory on how to do it. The model creation process done with trying different parameters: number of BiLSTM cells, number of layers, activation functions...etc.

1. **Training :** We split the dataset into 2 parts, Training, validation, in the Training process the weights adjusted to make the prediction to match the expected result, and the validation data is 20% from the dataset is used to know how the model can predict from unknown data.

1. **Evaluation :** The testing data is different than the training data will be used to test and evaluate the model, the testing data is located in a CSV file applying all preprocessing methods before evaluating.

1. **Prediction :** After reaching good results in training and evaluating, the model is ready for making predictions, which are the user's intention.

PseudoCode

Algorithm 1 BILSTM prediction model pseudocode

1: $data\_preprocessing(inputs, targets)$ ▷ Applying NLP methodes to convert text data into integer encoded
2: $inputs\_train, targets\_train, inputs\_validation, targets\_validation \leftarrow split\_data(data, 20)$ ▷ Split data to training and validation sets (20% validation)
3: $reshape\_data(data)$ ▷ Reshape data according to Embedding layer input
4: $model \leftarrow CreateSequentialModel()$ ▷ Creating and configuring the model
5: $model.add\_Embedding\_layer(vocabulary\_size, embeding\_dimension)$
6: $model.add\_Bidirectional\_LSTM\_layer(units, dropout = 0.2)$
7: $model.add\_NN\_layers(units, relu)$
8: $model.add\_Dropout(0.7)$
9: $model.add\_NN\_layers(target\_length, softmax)$
10: $model.compile()$
11: $history \leftarrow model.fit(inputs\_train, targets\_train, validation\_data = (inputs\_validation, targets\_validation), epochs = epochs)$ ▷ Training BILSTM model
12: $results \leftarrow model.evaluate(test\_data)$ ▷ Testing the model from testing data

Architecture

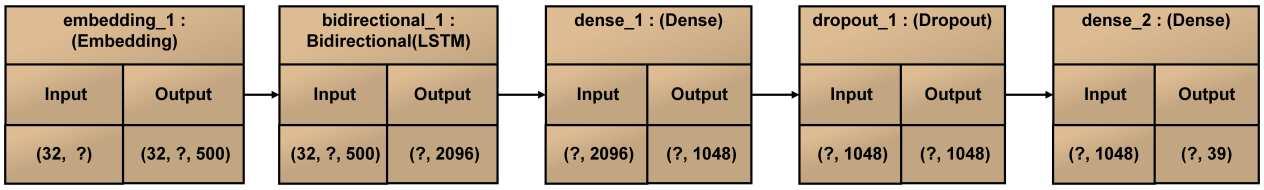


Figure 4.10: BiLSTM model architecture and layers

Our model was created with 5 layers, An embedding layer is the input layer that maps the words/tokenizers to a vector with embedding dimensions. Bidirectional, and Dense layers. The dropout layer is to drop the nodes so as to prevent overfitting. The output layer with 39 different feature which represent the intentions.

4.4 Conclusion

In this chapter we explained our architecture and the used algorithm. Our system is composed of two parts. Interaction part (Android application). Second part is backend server python scripts part (translation , natural language processing, model predection ...). In the next chapter we discuss the implementation and results.

Chapter 5

Implementation and results

5.1 Introduction

Application development process comes after a set of steps which their main goal is to develop an Android application capable of responding to the user's intentions about medical assistance. After presenting and discussing the theory part of our project and the details about the used approaches.In this chapter, We show the developed methods through some results. we present the used tools, platforms and librieries to perform chatbot for medical assistant. In the end, we present system interfaces and discussion about given results provided by our model.

5.2 Development tools and used platforms

Our system is composed, as mentioned before, an Android application and Backend process using Python scripts. The Android application represents the main interface of the interaction between the user and the Chatbot where the user type in the text bar his desired question and the received Chatbot answers are displayed in the main interface. The Backend process represents the treatment layer that proposed in the previous chapter that contain all the methods and algorithms used to extract user's intent and entities to build the Chatbot answer. The Sotfwares and libraries used in this project are listed below with a brief definition and explanation.

5.2.1 Android application

The Android application developed using Android studio software with combination of JAVA programming language and the Extensible Markup Language (XML) and other libraries

Android Studio

Android Studio is the Google official Integrated Development Environment (IDE) for Android app development [1], based on IntelliJ IDEA a powerful and intelligent code editor and developing tool. Android Studio offers unique features that enhance the productivity when building Android apps, such as:

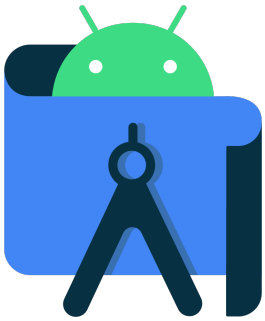


Figure 5.1: Android Studio

- A flexible Gradle based build system
- Cover a large testing tools and frameworks.
- Provide a fast emulator that has many options and functional capabilities.
- A grouped and united environment where you can develop for all Android devices.
- Support code templates and GitHub integration to help you build common app features and import sample code.
- You can Apply Changes to your running app without the need to restart it by pushing code and resource changes.
- Lint tools to catch performance, usability, version compatibility, and other problems

Each project in Android Studio contains one or more modules that contain source code files and resource files, such as Library modules, Android app modules, and Google App Engine modules. Each app module contains Manifests,Java where Java source code files and res file where Contains all non-code resources, such as XML layouts.

[1] Android Studio official Website https://developer.android.com/studio/intro

Extensible Markup Language (XML)

XML is used to define documents with a standard format that can be read by any XML compatible application, XML is easy to read and understand both by human and machines. Also, it is scalable and simple to develop.XML is used In Android development for designing layouts, due to his lightweight language so it doesn't make layouts heavy.

Figure 5.2: XML

JAVA

Java is a popular programming language that is class-based object-oriented, general purpose, concurrent and strongly typed, that is designed to have as few implementation dependencies as possible. It is normally compiled to the bytecode instruction set and binary format defined in the Java Virtual Machine Specification [1].

Figure 5.3: Java

5.2.2 Back-end Treatments

Google Colab

Colaboratory, or "Colab" for short, is a Google research project created to help students, data scientists and AI researchers for machine learning education and research. It's a Jupyter notebook environment that requires no setup to uses, free access to GPUs, easy sharing and runs entirely on cloud. allows to write and execute Python in the browser [2]. Training a deep learning model can require extensive CPU/GPU setup, that's why we need Google Colab cloud platform for this task.

Figure 5.4: Google Colab

Pycharm (Python code editor)

[1] https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html
[2] https://colab.research.google.com/

PyCharm is an integrated development environment (IDE) used in computer programming, It provides code analysis, an integrated unit tester, a graphical debugger, integration with version control systems (VCSes), and supports web development as well as data science with Anaconda [1]. In our project we use Pycharm due to his simplicity and provide a support for web framework FLASK for our server.

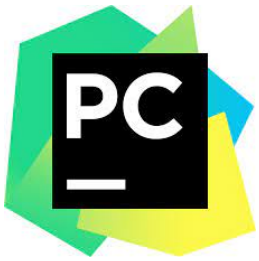


Figure 5.5: Pycharm IDE

Python

Python is an interpreted high level, general purpose programing language. Its language constructs as well as its object-oriented approach aim to help programmers write clear and logical code for small and large-scale projects. For developing our model huge set of helpful python libraries makes it the best choice for the Treatment process as mentioned in previous chapter, Some of the most important libraries we used:

Figure 5.6: Python

- **Flask**

  Flask is a micro web framework written in Python. Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools [2]. For our project we use FLASK for the connection between the Android application and the Treatments process Back-end python scripts that have been explained in the previous chapter.

- **Tensorflow and Keras :**

  Used to create ,train and predict in our deep learning model.

- **Pandas :**

  Used for preparing the data from the CSV dataset to train the model and perform analysis.

[1] https://www.jetbrains.com/pycharm/
[2] https://flask.palletsprojects.com/en/2.0.x/

- **NLTK :**

  NLTK (Natural Language Toolkit) is a leading library for building Python programs to work with human language data. It provides easy to use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial strength Natural Language Processing libraries [1].

- **langdetect :**

  langdetect is an implementation of Google's language-detection library in Python, We use it in our project to get the user's language so chatbot can converse with user's language.

- **Googletrans :**

  Googletrans is a free and unlimited python library that implemented Google Translate API. This uses the Google Translate Ajax API to make call to translate method to officiel google servers.

- **Matplotlib :**

  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object oriented API for embedding plots into applications using general purpose GUI toolkits. We use it in our project for model analysis and plotting acuraccy and loss graphs .

- **PySwip :**

  PySwip is a Python and SWI-Prolog bridge enabling, to query SWI-Prolog in Python programs. It features the SWI-Prolog foreign language interface, a utility class that makes it easy querying with Prolog and also a Pythonic interface [2]. We use PySwip in our project to retrieve chatbot answers from a prolog knowledge base.

[1] https://www.nltk.org/
[2] https://pypi.org/project/pyswip/

5.3 Chatbot process :

- initializing the back-end server:

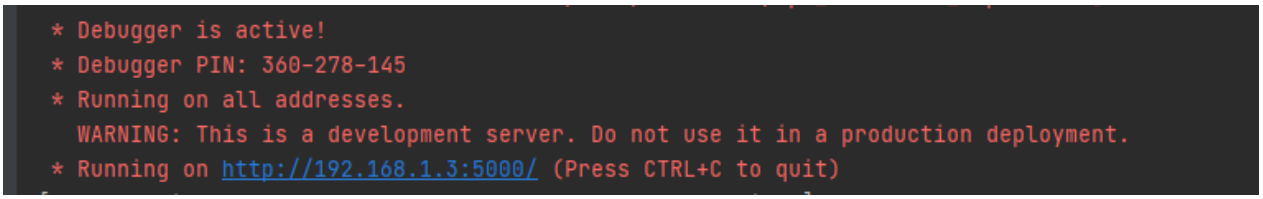


Figure 5.7: Initializing the FLASK server

Initializing the FLASK server for the back-end chatbot treatments, running on my local machine PC in the same network as the android application.

- **User input :**

The android application connect to the FLASK API automatically when opening the Application, User can type any desired question about medical assistant through the text bar in the main layout. Send the user's text message through the send button presented in Code 5.1 below

```
1 //Creating the connection instance and set timout to 30 seconds
2 OkHttpClient okHttpClient = new OkHttpClient.Builder().connectTimeout
    (30, TimeUnit.SECONDS).build();
3 //User input request form build
4 RequestBody formbody = new FormBody.Builder().add("chatInput",
    inputmessage).build();
5
6 //Sending a POST request to the bound IP of the FLASK server
7 Request request = new Request.Builder().url("http://192.168.1.3:5000/
    chat").post(formbody).build();
8
```

Listing 5.1: Message request build code

Figure 5.8 below, shows the main application interface.

Figure 5.8: Chatbot Application main layout

- **Text translation :**

  Our chatbot system model and knowledge base formalized with the English language, so a translation to English is needed when receiving non-English user language. The application support three different languages (English - Arabic - French)

```
    lng = detect(sentence)
    if lng == 'ar':
        sentence = ar_to_en(sentence)
    elif lng == 'fr':
        sentence = fr_to_en(sentence)

```

Listing 5.2: Detect user language

As the above code shows, user language detected using **langdetect** library, then translating into English through an API request to Google translation servers as the following pseudocode 5.3 shows.

```
from googletrans import Translator
translator = Translator()

def ar_to_en(input):
    global translator
    translation = translator.translate(input, dest='en')
    return translation.text

def en_to_ar(input):
    global translator
    translation = translator.translate(input, dest='ar')
    return translation.text

def fr_to_en(input):
    global translator
    translation = translator.translate(input, dest='en')
    return translation.text

def en_to_fr(input):
    global translator
    translation = translator.translate(input, dest='fr')
    return translation.text


```

Listing 5.3: Text translation

- **Tokenization :**

  The following pseudocode 5.4 represents the Tokenization method.

```
sent_seq = []
sentence = nltk.word_tokenize(sentence)
  for token in sentence:
        if token in tokenizer.word_index:
            sent_seq.append(tokenizer.word_index[token])
        else:
            sent_seq.append(0)
```

```
8 print("Tokenization : ",sentence)
9
```

Listing 5.4: Tokeniation code

The theory part of the Tokenization explained in the previous chapters, using the **NLTK** library Tokenizer for the iput sentence. Then we match each token to the model tokenizer if exist the sequence add the category label, if not add a zero, the output of the method on the sentence shown in Figure 5.9 .

```
Where can i finded doctor hakim
Tokenization :  ['Where', 'can', 'i', 'finded', 'doctor', 'hakim']
```

Figure 5.9: Tokenization output

- **Stemming :**

  We perform the next pseudocode 5.5 to ensure the stemming step.

```
1 stemmed = []
2   for token in sentence:
3         stemmed.append(token)
4  print("Stemmed :",stemmed)
5
```

Listing 5.5: Stemming code

Stemming was explained in the prievious chapters. Using the **NLTK** library Porter Stemmer to perform stemming method on the tokenized sentence. The output of this method in Figure 5.10 below.

```
Stemmed : ['where', 'can', 'i', 'find', 'doctor', 'hakim']
```

Figure 5.10: Stemming output

Stemming correct most of the spelling mistakes by eliminating suffixes and affixes as shown in Figure 5.10 in the word "Finded" to "Find".

- **Entities extraction :**

  Using this pseudocode 5.6 to extract entities.

```
pos = nltk.pos_tag(sentence)
print(pos)

```

Listing 5.6: Entities extraction (Part of speech tagging)

To extract the entities, We use built in **NLTK.pos_tag** function, which is a part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word. The output of the Part of Speech tagging is matching each word with a entity abbreviation [1] as Figure 5.11 shows :

```
[('where', 'WRB'), ('can', 'MD'), ('i', 'VB'), ('find', 'VB'), ('doctor', 'NN'), ('hakim', 'NN')]
```

Figure 5.11: Entities extraction output

The identified word tags:

  - **WRB** : Adverb
  - **MD** : Modal
  - **VB** : Verb
  - **NN** : Noun

- **Model prediction :**

  The next pseudocode 5.7 we perform our model prediction.

```
#Convert into 1D array
sent_seq = tf.expand_dims(sent_seq, 0)
```

[1] http://www.nltk.org/book/ch05.html

```
#Predict the category of input sentence
pred = model(sent_seq)
pred_class = np.argmax(pred.numpy(), axis=1)
arry = y_encoder.categories_
intent = arry[0][pred_class[0]]
print("Intent predicted : ",intent)

```

Listing 5.7: Model predection code

Before applying a model prediction, we need to convert the numerical tokenized sentence into 1 Dimensional array, with the built in tensorflow function **expand_dims**. The Figure 5.12 , shows the predicted intent from the input sequence "Where can i finded doctor hakim".

Figure 5.12: Predicting intent output

- **Answer retrieval :**

  In the answer retrieval phase, we apply a Prolog query function based on the user intent and extracted entities, using library **pyswip** on the knowledge base that is represented in Prolog file. the code below shows an example of two functions of building a Prolog query.

```
prolog = PrologMT()
  prolog.consult("knowledge base.pl")

  def find_doc():
      for soln in prolog.query("find_doc(X,Y,Z)"):
          res = f'{soln["X"]} is Currently on Floor {soln["Y"]} Room
  {soln["Z"]}'
      return res
  def wait_time():
      for soln in prolog.query("wait_time(X,Y,Z)"):
```

```
10              res = f'There are  {soln["X"]} people on queue right now,
    approximately {soln["Z"]} minutes waiting for you to meet {soln["Y
    "]}'
11          return res
12
```

Listing 5.8: Prolog query example code

An existing knowledge base stored in a Prolog file that can be consulted, and queried. consulting is the process of reading programs into Prolog.The Prolog interpreter responds to queries about the facts and rules represented in the knowledge base. Making a query is asking Prolog whether can prove that query or not. If so, return any variable bindings.

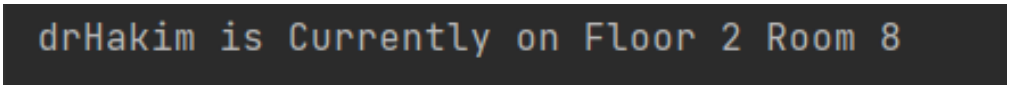


Figure 5.13: Retrieved answer from knowledge base

The retrieved answer will be translated to user language and sent to the android app with the POST method in the FLASK server.

```
1  app= Flask(__name__)
2
3  @app.route('/chat',methods=['GET','POST'])
4
5  def chatBot():
6      chatInput = request.form['chatInput']
7      chatBotReply = chatbotAnswer(chatInput)
8      return chatBotReply
9
10
11 if __name__ == '__main__':
12     #Start the API Server on the local machine address with the
    default port
13     app.run(host="0.0.0.0", debug=True)
14
```

Listing 5.9: FLASK API code

- **Output the answer :**

  The android application receives the answer and display it on the main layout as shown in the code below.

```
1  public void onResponse(@NotNull Call call, @NotNull final Response
       response) throws IOException {
2       runOnUiThread(new Runnable() {
3           public void run() {
4               String botreply = null;
5               try {
6                   botreply = response.body().string();
7               } catch (IOException e) {
8                   e.printStackTrace();
9               }
10              if (botreply!=null){
11                  Message outMessage=new Message();
12                  outMessage.setMessage(botreply);
13                  outMessage.setId("2"); messageArrayList.add(outMessage
       );
14                  mAdapter.notifyDataSetChanged();
15                  if (mAdapter.getItemCount() > 1) {
16                      recyclerView.getLayoutManager().
       smoothScrollToPosition(recyclerView, null, mAdapter.getItemCount()
       -1);}}}});
17
```

Listing 5.10: Receive the chatbot answer code

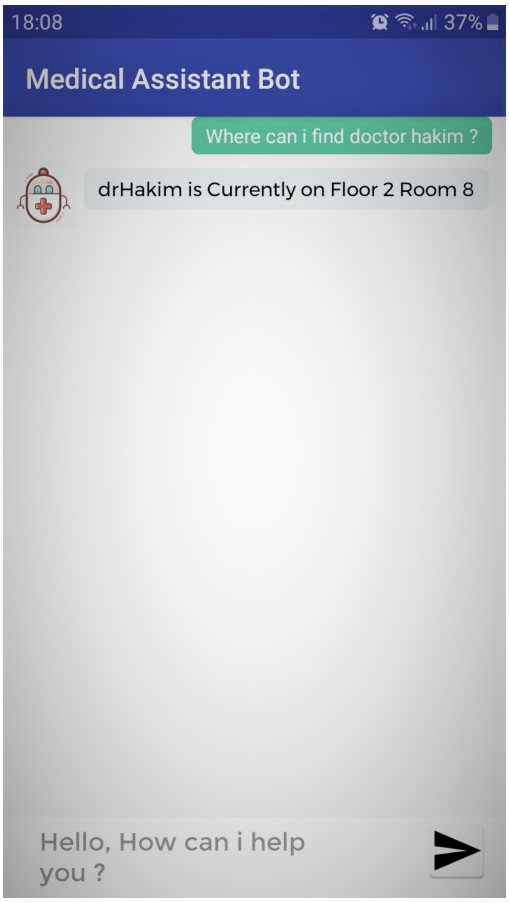


Figure 5.14: Chatbot reply display

5.4 System Interfaces and examples

In this section, we present some system usages and interfaces of our application with the three supported languages (Arabic - English - French)

- **English user**

  In this example in Figure 5.15 the Chatbot talks with an English person.

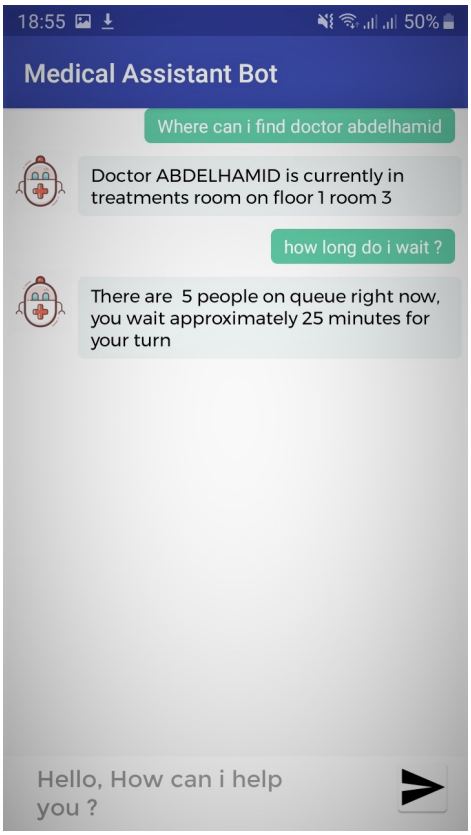


Figure 5.15: Chatbot test with English user

- **Arabian user**

  In this example in Figure 5.16 the Chatbot talks to Arabian person.

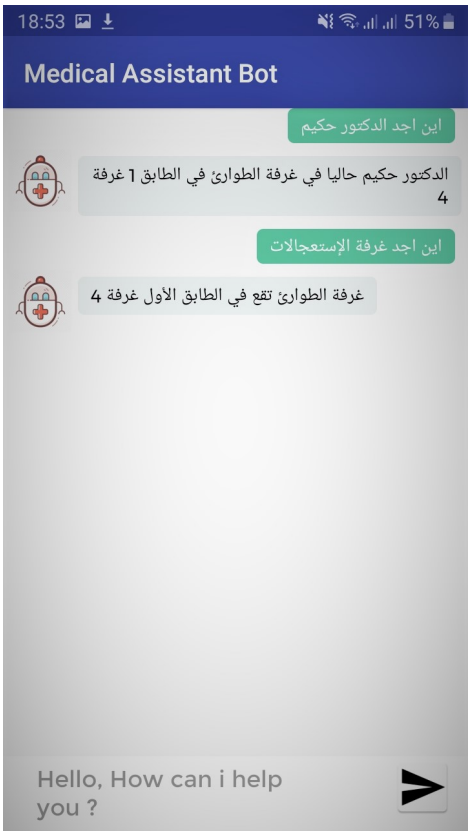


Figure 5.16: Chatbot test with Arabic user

- **French user**

  In this example in Figure 5.17 the Chatbot talks to French person.

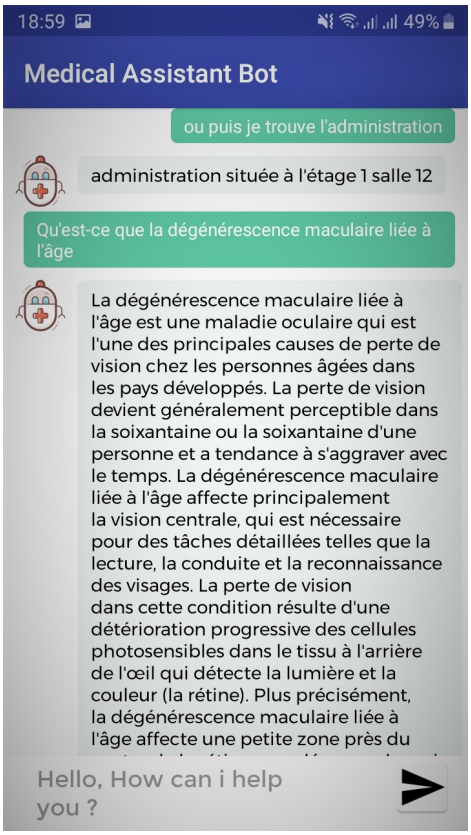


Figure 5.17: Chatbot test with French user

5.5 Obtained results and discussion

The input dataset from Ben Abacha and Demner-Fushman [2019], grouped into a CSV file contain two columns Intent, Question in a total of 39 different intent for classification. The dataset was transformed into One-Hot encoding categorical targets by given the value of One or Zero for each label.

- **Training** :

  The training process of the model, was in google colab. Training went for 60 epochs with the default batch size of 32. Our BiLSTM model was trying to learn the pattern between the question data and intents labels, in order to define and predict the intent correctly. the code 5.11 below shows the training process.

```
epochs=60
vocab_size=len(tokenizer.word_index) + 1
embed_dim=500
print(vocab_size,embed_dim)
units=1024
target_length= ytr_encoded.shape[1]
# build bidLSTM Model with tensorflow
model = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(vocab_size, embed_dim),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units, dropout
   =0.2)),
    tf.keras.layers.Dense(units, activation='relu'),
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(target_length, activation='softmax')
])

optimizer = tf.keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
   metrics=['accuracy'])
model.summary()
history = model.fit(train_padded, ytr_encoded, validation_split= 0.20,
    epochs=epochs)
model.save('/content/drive/MyDrive/Model')


```

Listing 5.11: Model training architecture

Figure 5.18 below shows the start of the training process. We can see at the beginning of the training,the accuracy value was very low with a high loss value.

```
_________________________________________________________________
Epoch 1/60
1084/1084 [==============================] - 39s 34ms/step - loss: 8.4615 - accuracy: 0.0015 - val_loss: 8.3736 - val_accuracy: 5.3571e-04
Epoch 2/60
1084/1084 [==============================] - 36s 33ms/step - loss: 7.8450 - accuracy: 0.0047 - val_loss: 7.8883 - val_accuracy: 0.0045
Epoch 3/60
1084/1084 [==============================] - 36s 33ms/step - loss: 7.1903 - accuracy: 0.0217 - val_loss: 7.1414 - val_accuracy: 0.0304
Epoch 4/60
1084/1084 [==============================] - 36s 33ms/step - loss: 6.2381 - accuracy: 0.0700 - val_loss: 6.1160 - val_accuracy: 0.1102
Epoch 5/60
1084/1084 [==============================] - 36s 33ms/step - loss: 5.0648 - accuracy: 0.1602 - val_loss: 4.9001 - val_accuracy: 0.3075
```

Figure 5.18: Beginning of the training with BiLSTM model

After 60 epochs,the training process accuracy reach 98% and validation accuracy about 97%, and a lower loss value in the training (0.05) and (0.11) in validation loss, each epoch took about 36-37 seconds as shown in Figure 5.19.

```
 855/1084 [======================>.......] - ETA: 7s - loss: 0.0673 - accuracy: 0.9798
Epoch 00047: saving model to /content/drive/MyDrive/ModelCheckoints6112021/cp-0047.ckpt
1084/1084 [==============================] - 37s 34ms/step - loss: 0.0706 - accuracy: 0.9792 - val_loss: 0.1009 - val_accuracy: 0.9762
Epoch 48/60
1084/1084 [==============================] - 36s 34ms/step - loss: 0.0741 - accuracy: 0.9777 - val_loss: 0.1051 - val_accuracy: 0.9766
Epoch 49/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0684 - accuracy: 0.9786 - val_loss: 0.1320 - val_accuracy: 0.9734
Epoch 50/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0739 - accuracy: 0.9781 - val_loss: 0.0988 - val_accuracy: 0.9771
Epoch 51/60
1084/1084 [==============================] - 37s 34ms/step - loss: 0.0684 - accuracy: 0.9788 - val_loss: 0.0919 - val_accuracy: 0.9777
Epoch 52/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0683 - accuracy: 0.9790 - val_loss: 0.0885 - val_accuracy: 0.9780
Epoch 53/60
 691/1084 [==================>...........] - ETA: 12s - loss: 0.0641 - accuracy: 0.9800
Epoch 00053: saving model to /content/drive/MyDrive/ModelCheckoints6112021/cp-0053.ckpt
1084/1084 [==============================] - 37s 34ms/step - loss: 0.0655 - accuracy: 0.9800 - val_loss: 0.1142 - val_accuracy: 0.9741
Epoch 54/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0673 - accuracy: 0.9798 - val_loss: 0.1108 - val_accuracy: 0.9752
Epoch 55/60
1084/1084 [==============================] - 36s 34ms/step - loss: 0.0641 - accuracy: 0.9800 - val_loss: 0.0910 - val_accuracy: 0.9777
Epoch 56/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0587 - accuracy: 0.9820 - val_loss: 0.1129 - val_accuracy: 0.9734
Epoch 57/60
1084/1084 [==============================] - 37s 34ms/step - loss: 0.0614 - accuracy: 0.9808 - val_loss: 0.1314 - val_accuracy: 0.9741
Epoch 58/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0627 - accuracy: 0.9808 - val_loss: 0.1049 - val_accuracy: 0.9770
Epoch 59/60
 527/1084 [=============>................] - ETA: 17s - loss: 0.0608 - accuracy: 0.9816
Epoch 00059: saving model to /content/drive/MyDrive/ModelCheckoints6112021/cp-0059.ckpt
1084/1084 [==============================] - 38s 35ms/step - loss: 0.0585 - accuracy: 0.9821 - val_loss: 0.1037 - val_accuracy: 0.9768
Epoch 60/60
1084/1084 [==============================] - 36s 33ms/step - loss: 0.0579 - accuracy: 0.9826 - val_loss: 0.1150 - val_accuracy: 0.9761
```

Figure 5.19: Ending of the training with BiLSTM model

The total time of the training took around 36 minutes, resulted in a high scores in both accuracy and validation.

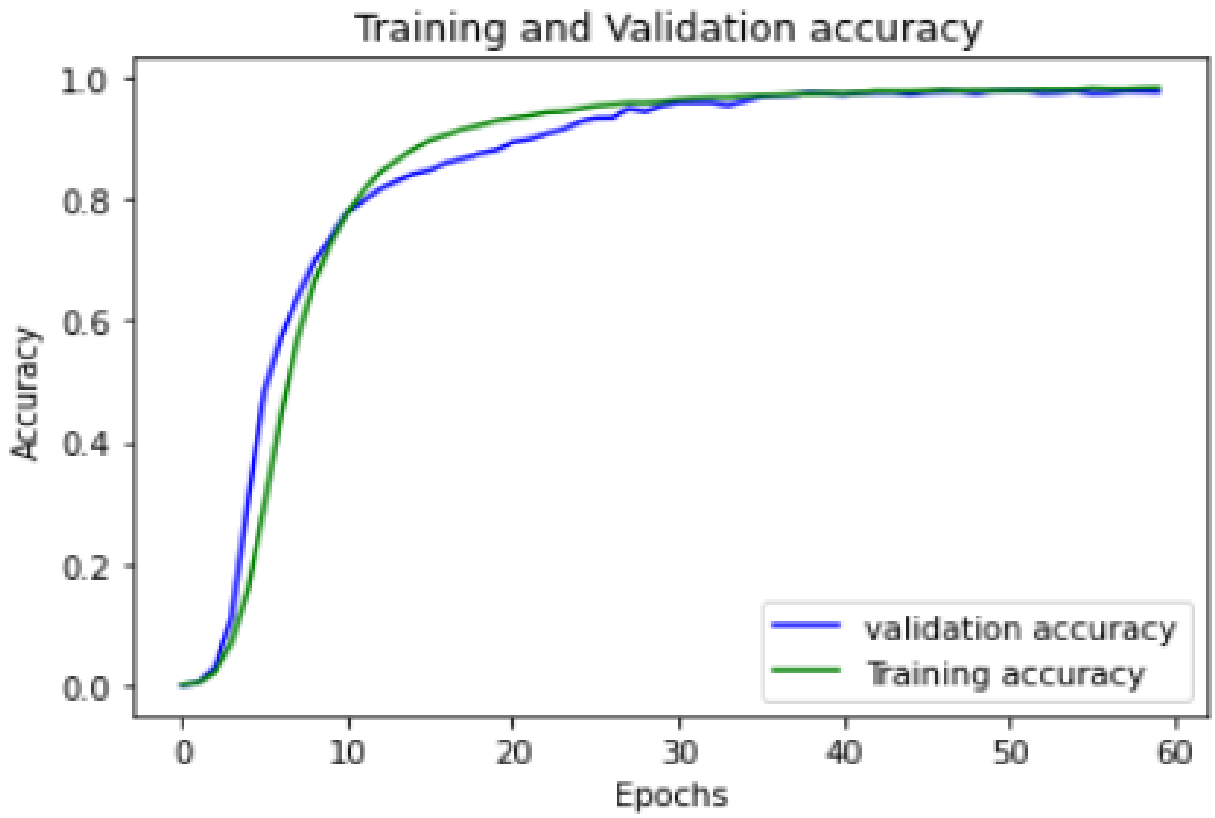


Figure 5.20: Training and validation accuracy

Model accuracy is a very important component in assessing model performance, After running a couple of dozen tests and parameters changing,This model accuracy improved over time, from 0.0015 to 0.98 means that the BiLSTM model was able to fit the data. Predicting correctly most of the time the correct intents, the validation accuracy is also considered very high, 0.97. The trained model uses a different set of data for validation that was not included in the training set, thus it was able to predict correctly for completely different set of data., it is concluded that the validation accuracy of the dataset lies around 97% and the training set accuracy rises to 98%. From these results we can say that with the current size of the dataset being around 41480 total of question, These results are good.

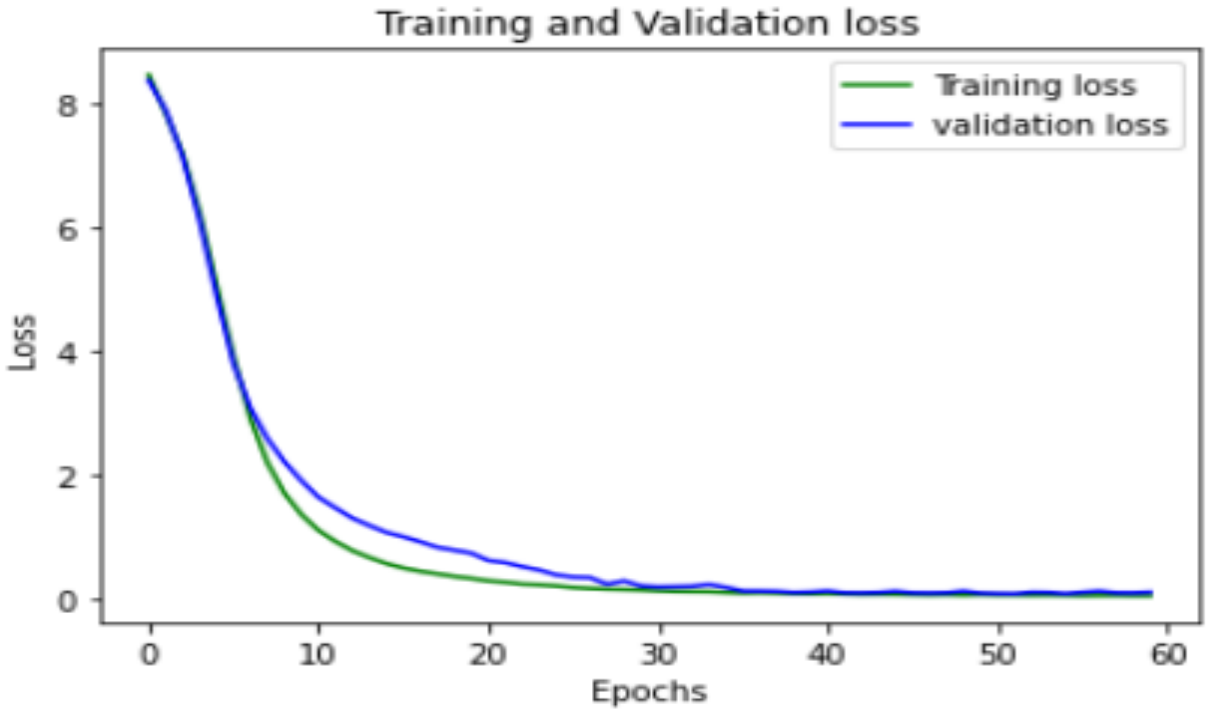


Figure 5.21: Training and validation Loss

For calculating the loss, we used Categorical Cross Entropy function. It's the most common function when it comes to classification problems. Categorical Cross Entropy increases as the predicted probability diverges from the actual label. The model works on minimizing the loss function, which process all aspects of our algorithm into a single numerical value that describes how efficient our model is. At the earlier stages of the training, loss function had a high value (8.46), BiLSTM tries to learn form the data and thus minimizing the loss by using the back propagation as training goes. After few epochs the value drops significantly to 0.006, which indicates that the model succeeded at extracting features from the data set and identifying the pattern to correctly predict intents.

5.6 Conclusion

In this chapter we presented the implementation of our system as well as the obtained results. we chose our software programs and libraries based on the programming needs and the best methods achieve our goal. The results were promising, our model had a high accuracy in prediction, which reflects the well structured system and architecture.

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

Medical assistant presents a very important side of the medical domain, our work mainly focuses on providing the best support and assistant to both medical crew and patients. On that matter, the proposed architecture which is a combination of Deep Learning and Expert system, showed efficiency in predicting user's intentions and provide accurate answer. Compared to the related works on different chatbots, our solution method uses Bidirectional Long Short Term Memory (BiLSTM) instead of classic Artificial Neural Networks for intent recognition as sequence patterns inputs rather than considering the data just scattered values, Our solution also uses a Prolog expert system to retrieve the answer from the knowledge base. However, the knowledge base needs to be constantly updated each time a change on the database occurs.

6.2 Perspectives

Our work can be extensible and enhanced for better and improved medical assistant process. As a future work, we aim to introduce voice recognition to our system using speech-to-text and text-to-speech methods, patients should not waste time typing when they need to save time speaking directly to the Chatbot. Additionally we want to add a symptom based diagnosis system using larger medical dataset, this system should predict possible diseases through an image taken by patient mobile phone using Convolutional Neural Networks (CNN) for image processing and extracting useful information about possible patient injuries, we also wish to add

a health condition tracker system for chronic diseases, so the application can remind patient about their pharmaceutical and medicine's by saving all patients on a database.

References

Nahdatul Akma Ahmad, Mohamad Hafiz Che, Azaliza Zainal, Muhammad Fairuz Abd Rauf, and Zuraidy Adnan. Review of chatbots design techniques. *International Journal of Computer Applications*, 181(8):7–10, 2018.

Ebtesam H. Almansor and Farookh Khadeer Hussain. Survey on intelligent chatbots: State-of-the-art and future research directions. In Leonard Barolli, Farookh Khadeer Hussain, and Makoto Ikeda, editors, *Complex, Intelligent, and Software Intensive Systems*, pages 534–543, Cham, 2020. Springer International Publishing. ISBN 978-3-030-22354-0.

Lekha Athota, Vinod Kumar Shukla, Nitin Pandey, and Ajay Rana. Chatbot for healthcare system using artificial intelligence. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 619–622. IEEE, 2020.

Asma Ben Abacha and Dina Demner-Fushman. A question-entailment approach to question answering. *BMC Bioinform.*, 20(1):511:1–511:23, 2019. URL https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3119-4.

Petr Berka. Sentiment analysis using rule-based and case-based reasoning. *Journal of Intelligent Information Systems*, pages 1–16, 2020.

Paul Booth. *An introduction to human-computer interaction (psychology revivals)*. Psychology Press, 2014.

Petter Bae Brandtzaeg and Asbjørn Følstad. Why people use chatbots. In *International conference on internet science*, pages 377–392. Springer, 2017.

Petter Bae Brandtzaeg and Asbjørn Følstad. Chatbots: changing user needs and motivations. *Interactions*, 25(5):38–43, 2018.

Daniel Braun and Florian Matthes. Towards a framework for classifying chatbots. In *ICEIS (1)*, pages 496–501, 2019.

Chatbot in English by Lexic Dictionaries. Lexico dictionary, 2019.

Leigh Clark, Philip Doyle, Diego Garaialde, Emer Gilmartin, Stephan Schlögl, Jens Edlund, Matthew Aylett, João Cabral, Cosmin Munteanu, Justin Edwards, et al. The state of speech in hci: Trends, themes and challenges. *Interacting with Computers*, 31(4):349–371, 2019.

Amélie Cordier, Bruno Mascret, and Alain Mille. Extending case-based reasoning with traces. In *Grand Challenges for reasoning from experiences, Workshop at IJCAI*, volume 9, page 31, 2009.

Thiago Carvalho D'Ávila et al. Kino: an approach for rule-based chatbot development, monitoring and evaluation. 2018.

Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.

Cem Dilmegani. Natural language platforms: Top nlp apis & comparison. https://research.aimultiple.com/natural-language-platforms/, 2021.

S Divya, V Indumathi, S Ishwarya, M Priyasankari, and S Kalpana Devi. A self-diagnosis medical chatbot using artificial intelligence. *Journal of Web Development and Web Designing*, 3(1):1–7, 2018.

Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160, 2017.

Jacob Eisenstein. *Introduction to natural language processing*. MIT press, 2019.

Doaa Mohey El-Din. Enhancement bag-of-words model for solving the challenges of sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 7(1), 2016.

Shao Fang, Ahmed Faeq Hussein, S Ramkumar, KS Dhanalakshmi, and G Emayavaramban. Prospects of electrooculography in human-computer interface based neural rehabilitation for neural repair patients. *IEEE Access*, 7:25506–25515, 2019.

Asbjørn Følstad and Petter Bae Brandtzæg. Chatbots and the new world of hci. *interactions*, 24 (4):38–42, 2017.

Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.

Tuna Gersil and Ismail Hilal. The role of conversational interfaces in the future of digitaland technology, 2020.

Ashok Goel and Belen Diaz-Agudo. What's hot in case-based reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Aishwarya Gupta, Divya Hathwar, and A Vijayakumar. Introduction to ai chatbots. *Int. J. Eng. Res. Technol*, 9(7), 2020.

Aditya Jain, Gandhar Kulkarni, and Vraj Shah. Natural language processing. *International Journal of Computer Sciences and Engineering*, 6(1):161–167, 2018.

Prashant Johri, Sunil K Khatri, Ahmad T Al-Taani, Munish Sabharwal, Shakhzod Suvanov, and Avneesh Kumar. Natural language processing: History, evolution, application, and future work. In *Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020*, page 365. Springer Nature, 2020.

Sethunya R Joseph, Hlomani Hlomani, Keletso Letsholo, Freeson Kaniwa, and Kutlwano Sedimo. Natural language processing: A review. *Natural Language Processing: A Review*, 6:207–210, 2016.

Uday Kamath, John Liu, and James Whitaker. *Deep learning for NLP and speech recognition*, volume 84. Springer, 2019.

AA Karpov and RM Yusupov. Multimodal interfaces of human–computer interaction. *Herald of the Russian Academy of Sciences*, 88(1):67–74, 2018.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *arXiv preprint arXiv:1708.05148*, 2017.

Pavel Kucherbaev, Alessandro Bozzon, and Geert-Jan Houben. Human-aided bots. *IEEE Internet Computing*, 22(6):36–43, 2018.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921, 2016.

Jean-Baptiste Lamy, Boomadevi Sekar, Gilles Guezennec, Jacques Bouaud, and Brigitte Séroussi. Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. *Artificial intelligence in medicine*, 94:42–53, 2019.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

Samuel Manoharan. A smart image processing algorithm for text recognition information extraction and vocalization for the visually challenged. *Journal of Innovative Image Processing (JIIP)*, 1(01):31–38, 2019.

Ifigeneia Mavridou, James T McGhee, Mahyar Hamedi, Mohsen Fatoorechi, Andrew Cleal, Emili Ballaguer-Balester, Ellen Seiss, Graeme Cox, and Charles Nduka. Faceteq interface demo for emotion expression in vr. In *2017 IEEE Virtual Reality (VR)*, pages 441–442. IEEE, 2017.

Ketakee Nimavat and Tushar Champaneria. Chatbots: An overview types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev*, 5(7):1019–1024, 2017.

Victoria Oguntosin and Ayobami Olomo. Development of an e-commerce chatbot for a university shopping mall. *Applied Computational Intelligence and Soft Computing*, 2021, 2021.

Nourchène Ouerhani, Ahmed Maalel, and Henda Ben Ghézela. Spececa: a smart pervasive chatbot for emergency case assistance based on cloud computing. *Cluster Computing*, 23(4):2471–2482, 2020.

José Padarian and Ignacio Fuentes. Word embeddings for application in geosciences: development, evaluation, and examples of soil-related concepts. *Soil*, 5(2):177–187, 2019.

Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International journal of computer applications*, 175(4):7–9, 2017.

Jinxian Qi, Guozhang Jiang, Gongfa Li, Ying Sun, and Bo Tao. Intelligent human-computer interaction based on surface emg gesture recognition. *IEEE Access*, 7:61378–61387, 2019.

Lara Quijano-Sanchez, Juan A Recio-Garcia, Belen Diaz-Agudo, and Guillermo Jimenez-Diaz. Social factors in group recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):1–30, 2013.

Nourhène Ben Rabah, Ines Ben Tekaya, and Moncef Tagina. Cbr-odaf: A case-based reasoning for the online diagnosis of all internal faults in automated production systems. In *Proceedings of 32nd International Conference on*, volume 63, pages 43–52, 2019.

Kiran Ramesh, Surya Ravishankaran, Abhishek Joshi, and K Chandrasekaran. A survey of design techniques for conversational agents. In *International conference on information, communication and computing technology*, pages 336–350. Springer, 2017.

Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial intelligence review*, 43(1):1–54, 2015.

Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

Osama Shahid, Mohammad Nasajpour, Seyedamin Pouriyeh, Reza M Parizi, Meng Han, Maria Valero, Fangyu Li, Mohammed Aledhari, and Quan Z Sheng. Machine learning research towards combating covid-19: Virus detection, spread prevention, and medical assistance. *Journal of Biomedical Informatics*, 117:103751, 2021.

Ghare Shifa, Shaikh Sabreen, Shaikh Tasmia Bano, and Awab Habib Fakih. Self-diagnosis medical chat-bot using artificial intelligence, 2020.

Heung-Yeung Shum, Xiao-dong He, and Di Li. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1): 10–26, 2018.

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

Yasir Ali Solangi, Zulfiqar Ali Solangi, Samreen Aarain, Amna Abro, Ghulam Ali Mallah, and Asadullah Shah. Review on natural language processing (nlp) and its toolkits for opinion mining and sentiment analysis. In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–4. IEEE, 2018.

Tomasz Walkowiak, Szymon Datko, and Henryk Maciejewski. Bag-of-words, bag-of-topics and word-to-vec based subject classification of text documents in polish-a comparative study. In *International Conference on Dependability and Complex Systems*, pages 526–535. Springer, 2018.

Susan Weinschenk and Dean T Barker. *Designing effective speech interfaces*. John Wiley & Sons, Inc., 2000.

Rainer Winkler and Matthias Soellner. Unleashing the potential of chatbots in education: A state-of-the-art analysis. 2018.

Özal Yildirim. A novel wavelet sequence based on deep bidirectional lstm network model for ecg signal classification. *Computers in biology and medicine*, 96:189–202, 2018.

Ozal Yildirim, Ulas Baran Baloglu, Ru-San Tan, Edward J Ciaccio, and U Rajendra Acharya. A new approach for arrhythmia classification using deep coded features and lstm networks. *Computer methods and programs in biomedicine*, 176:121–133, 2019.