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° : IA01/M2/2023

Thesis

Submitted in fulfilment of the requirements for the Masters degree in

Computer science

Option : **Artificial Intelligence**

Deep learning and Sentiment analysis techniques to ensure customer satisfaction through AI bots

By :
MGHEZZI BAKHOUCHE MOHAMED

Members of the jury :

TIBERMACINE Ahmed	MCA	President
MERIZIG Abdelhak	MCB	Supervisor
BELAALA Abir	MCB	Member

Session 2023

Acknowledgments

In the name of Allah, the Most Gracious, the Most Merciful.

I express my deepest gratitude and appreciation to Allah, the Most Merciful, for bestowing upon me the strength, guidance, and blessings throughout my academic journey.

I would like to extend my heartfelt thanks to my supervisor, MERIZIG Abdelhak, for his invaluable support, patience, and guidance. His unwavering dedication, expertise, and encouragement have played a pivotal role in shaping my research and academic growth. I am truly grateful for his mentorship and the trust he placed in me.

I would like to express my gratitude to my family and friends for their unwavering love, encouragement, and understanding. Their constant support and belief in my abilities have been a source of inspiration throughout this journey.

I would like to express my special thanks to my mother for her exceptional support and encouragement throughout this journey. Her unwavering belief in my abilities and constant motivation have been instrumental in my achievements. I am truly grateful for her love, guidance and unwavering presence in my life.

MGHEZZI BEKHCOUHE Mohamed

Abstract

With the rapid advancements of artificial intelligence , the field of human-machine interaction has experienced significant advancements. This project focuses on developing a chatbot system that facilitates human-machine interaction in the context of customer support in businesses. The chatbot aims to comprehend and react to human interactions by employing deep learning, transfer learning, and natural language processing (NLP) techniques. The objective of this work is to build a chatbot that helps in predicting customer intentions and sentiments based on their requests and provide relevant and helpful responses. The implementation involves training models using BERT for classifying intention and Glove-LSTM for sentiment classification in order to effectively analyze and comprehend customer inputs, as a result enhancing the customer support experience through efficient and personalized interactions.

Key word : Artificial intelligence, Human-machine interaction, Chatbot, Natural language processing, BERT, Glove-LSTM , Customer support.

Résumé

Avec les avancées rapides de l'intelligence artificielle, le domaine de l'interaction homme-machine a connu des progrès significatifs. Ce projet se concentre sur le développement d'un système de chatbot qui facilite l'interaction homme-machine dans le contexte du service client dans les entreprises. Le chatbot vise à comprendre et réagir aux interactions humaines en utilisant des techniques d'apprentissage profond, de transfert d'apprentissage et de traitement du langage naturel (NLP). L'objectif de ce travail est de construire un chatbot qui aide à prédire les intentions et les sentiments des clients en fonction de leurs demandes et fournir des réponses pertinentes et utiles. La mise en œuvre implique l'entraînement de modèles utilisant BERT pour la classification des intentions et Glove-LSTM pour la classification des sentiments afin d'analyser et de comprendre efficacement les donneés des clients, améliorant ainsi l'expérience de service client grâce à des interactions efficaces et personnalisées.

mot clé : Intelligence artificielle, Interaction homme-machine, Chatbot, Traitement du langage naturel, BERT, Glove-LSTM , Service client.

Contents

List of Figures

List of Tables

General Introduction

General Context

In the business world, customer support is a service that should not be neglected and communication between the business and customer plays a fundamental role in it, whether before or after the sale. Indeed we say that the customer is the master and we must take into consideration his needs, intentions and sentiments to guarantee a good service in a short time.

During the past few years, communication between businesses and clients has become considerably easier, developed and more efficient thanks to the latest innovations and technologies of Artificial Intelligence. Among these latest technologies are AI bots that can assist both customers and service providers.

The integration of AI bots, supported by deep learning algorithms and sentiment analysis models, has increased the effectiveness of customer interactions by enabling businesses to better understand their customer's needs, making creative recommendations, fixing issues quickly, and offering specific solutions. This tool led to customer satisfaction, therefore an increase in customer loyalty which is the key element to the success of any businesses.

Recently, the development of AI chatbots that can serve users in a number of disciplines has advanced significantly. These technologies are capable of understanding user requests, interpreting their intentions and sharing essential data. This includes communicating through websites, phones, and social media platforms.

Problematic and Objectives

a few years ago customer had to physically visit a service provider to request a service and express his needs. This was already causing problems with the opening hours of the establishment, as well as the time consumed and money spend by the client.

As digital platforms and E-services have grown quickly, the number of consumers is increasing, which led to difficulties in responding to all these requests because of the availability and the ability of human to process all these requests is very limited, even impossible and costly.

The solution to this problem, as previously mentioned, is to introduce a virtual robot ”the AI bots” which has the ability to assist the company and the customers without any loss of time or money and with unprecedented efficiency. Many work use chatbot in customer support to provide assistance and facilitating the customer service experience. Our work takes it a step further by not only assisting customers with their intentions, but also taking into consideration their sentiments.

In this project, our goal is to develop a virtual robot that improves customer interaction by focusing on their unique needs and sentiments. The goal is to provide a satisfying customer experience that fulfills their individual requirements. We proposed a solution based on deep learning and transfer learning to deal with this problem mentioned.

Structure of the thesis

Our work is structured as follows:

The first chapter is chatbot literature background : we define human-machine interaction and case-based reasoning and its limitations. We will also introduce chatbots and their role in customer support. We will then discuss the problem statements and present some related work.

The second chapter is Natural Language Processing methods : we define natural language processing, challenges, components and various NLP techniques. Then we will discuss transfer learning (transformers) with NLP. Finally, we will introduce the domain of sentiment analysis.

The third chapter is Design and Contribution :we will present the proposed architecture. Then we will present an overview of the modules and their respective tasks within the proposed architecture. Additionally, we will explain the development process of our customer assistance chatbot and the deep learning algorithms used.

The fourth chapter Implementation and results : we first introduce the development tools and frameworks. Then, we will explain the process of preparing our dataset and training the models. Next, we will discuss the results obtained from our experiments. Finally, we demonstrate the functionality of our chatbot and provide some examples.

Finally we conclude the manuscript and we present some perspectives.

Chapter 1

Chatbot literature background

1.1 Introduction

Human beings have a fundamental need for communication to express their thoughts, needs and intentions, and their language has continuously evolved over time. However, since the emergence of machines, whose main role is to interact with humans through their components, the concept of Human-Machine Interaction (HMI) has become more than necessary.

In the field of HMI, the objective is to enable more efficient and facilitates the interaction between the user and the computer. This involves the use of AI techniques to make the behavior of machines as close as possible to human language. These techniques encourage machines to understand and generate human language.

This evolution has witnessed a remarkable acceleration in recent years with the emergence of new communication systems such as virtual assistants and chatbots. The goal is to create a conversational experience that is both effective and enjoyable. These AI chatbots have been developed to become partners and tools that facilitate tasks, provide information and deliver services while considering the needs, preferences and expectations of humans.

1.2 Human Machine Interaction

The communication and interaction between humans and machines, such as computers, robots and other devices, is referred to as human-machine interaction (HMI). It includes the design, development and testing of technologies that enables humans and machines to collaborate efficiently [1].

1.2.1 Definition

Human-machine interaction (HMI) involves exploring and understanding how humans and machines interact with each other. This includes studying how humans use machines, how machines learn from human interactions, and how to design interfaces that facilitate natural communication and collaboration between humans and machines. In fact, HMI focuses on creating efficient and effective interfaces that enable seamless communication and collaboration between humans and machines [2].

HMI is becoming more common in today's society as interactive virtual and robotic technologies improve. The capacity of artificial intelligence to imitate human social solutions is required for HMI to be useful in training people for future social interactions and to ensure the real-time cooperation required for effective HMI [3].

HMI is a research area of how humans interact with machines, robots and other automated systems. It covers an extensive variety of disciplines, including computer science, psychology and engineering, and aims to enhance the usability, accessibility and overall user experience of technology [4].

1.2.2 Types of Human machine interaction

Various forms of HMI have been investigated and developed throughout the years. Some of the most prevalent forms of HMI are:

1. **Graphical User interaction (GUI) :** GUIs are a sort of HMI that allows people to interact with machines through the use of visual components such as icons, menus, and windows. Graphical user interfaces (GUIs) are widely used in computer applica-

tions and mobile devices. GUI is perhaps the most common field of HCI study, which combines a wide range of topics with a wide range of applications. The similarity across these several sectors is that at least one physical sensor is employed between the user and the machine to facilitate interaction [5].

2. **Voice User interaction (VUI) :** VUIs employ voice commands to allow customers to engage with technologies such as virtual assistants or smart speakers [6].

   According to [7] VUIs is concerned with data derived from various audio streams. While the type of audio signals is not as flexible as that of visual signals, the information obtained from them can be more exact, beneficial and in some cases unique. The study areas in this section are classified as follows:

   - **Speech Recognition**
   - **Speakers Recognition**
   - **Musical Interaction**

3. **Touch User interaction (TUI) :** Touch screens are used to allow consumers to interact with gadgets such as smartphones or tablets. TUI is a component of specific devices that can handle human-machine interactions, and it composed of hardware and software that translates user inputs into signals for machines that offer the desired output to the user [8].

4. **Brain-Computer interaction (BCI) :** Brain-computer interfaces (BCIs) employ brain impulses to allow people to communicate with technologies such as prosthetic devices or gaming apps. This HMI enables people to communicate with machines using brain signals such as electroencephalograms (EEGs) or functional magnetic resonance imaging (fMRIs) [9].

5. **Augmented reality (AR) and virtual reality (VR) interaction**: Is an HMI that use digital displays to create immersive environments in which humans interact with machines. For example, both vision and speech recognition fail to recognize tiny details such as finger movements [10].

1.2.3 Case-based reasoning

Case-based reasoning is an artificial intelligence problem-solving approach that requires employing previous expertise to analyze and solve new problems. This approach categorizes expertise into "cases" and compares to the present situation. CBR is utilized in a variety of forms such as Changing current solutions to meet novel requirements, using prior cases to describe novel situations, employing old cases to evaluate novel solutions, reasoning from history to comprehend a new scenario (much like lawyers do) or offer a reasonable answer to a new problem [11].

According to [12] with the exception of expert systems and machine learning, CBR has resulted in the most practical applications of any AI approach family. IBM's Watson system is well-known example for proving the capability of memory-based reasoning.

Richter [13] gives us a clear idea of CBR system which is a sluggish of machine learning method. It is based on an archive that stores 'cases' or previously solved issues and their answers. This is known as the 'case-base' archive. The CBR engine passes through a sequence of processes known as the CBR cycle's (4R) :

- **Retrieve :**A comparable previous example is recalled from memory.
- **Reuse :**The solution to the previous example is modified to match the current issue.
- **Revise :**The updated solution is examined and assessed to ensure that it fulfills the current problem's needs.
- **Retain :**The new solution is saved in memory for future use.

This Figure 1.1 illustrate the CBR cycle's (4R).

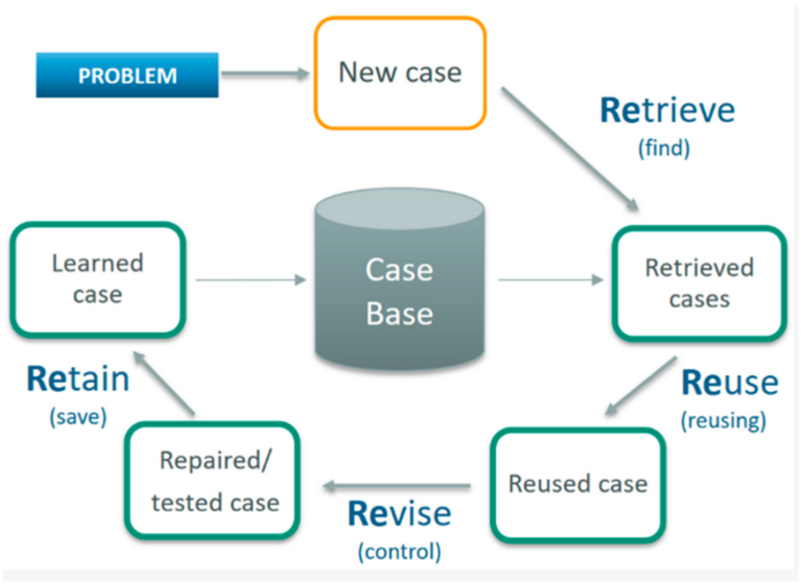


Figure 1.1: Case-Based Reasoning CBR architecture [14]

CBR has been applied in many fields, including engineering, health, law and business. It has been found to be successful in fields where the problem-solving task is complicated and requires a large amount of knowledge and expertise to solve.

1.2.4 Case based reasoning limits

CBR is not appropriate for all sorts of issues and may be ineffective in handling difficult problems requiring a high level of generalization or reasoning [15].

Difficulty in case adaptation: CBR necessitates case adaptation to meet the present problem situation. The adaption process might be difficult, especially when the retrieved instances are not immediately related to the current situation [16].

Difficulty in acquiring knowledge: CBR necessitates the collection of new cases in order to extend the case base and enhance the system's performance. However, acquiring expertise can be difficult, specifically if the domain is complicated or quickly changing [17].

Case-based reasoning has difficulty dealing with natural language, which demands a representation of information that is tightly linked to the syntax and the context [18].

1.3 Revolutionizing of Human-Machine Interaction

The chatbot is a well-known example of an artificial intelligence (AI) system and one of the most fundamental and extensively utilized types of intelligent HMI [19].

1.3.1 Chatbot

Chatbots have become more popular in a wide range of businesses, including marketing, assistance systems, healthcare, educational institutions, cultural heritage and entertainment.

1.3.1.1 Definition

A chatbot is an example of computer software that deals with natural language input from a user and generates fairly complex, wealthy and intelligent replies to the user [20].

It is a form of computer program that acts like intelligent individual when chatted via text or voice and comprehend one or more human languages using Natural Language Processing (NLP) [21].

A chatbot aims to simulate natural conversation by carrying out a dialog and sharing messages. Most domains already use it at the first point of contact with a customer or assistance representative [22].

They are becoming more popular and look to be promising in terms of assisting the business industry. They try to provide a complement to human assistance over the internet. They may offer useful data and in the case of more difficult requests. Actually, they track the achievement stories of virtual assistants like Alexa and Siri, rather than the expected reverse order. Virtual assistants facilitated human-machine interaction and exemplified the future of effective information transmission. Intelligent virtual assistants, chatbots and other language-based human-machine interfaces (HMI) provide information without the need for time-consuming searches. Moreover, they mask the complicated nature and volume of information [23] .

1.3.1.2 Motivation to use Chatbot

Why do people utilize chatbots? Chatbots look to have an enormous amount of opportunities for providing customers with quick and easy service that is corresponding to their specific requirements. The most prevalent motivator for chatbot clients is performance, followed by enjoyment, social objectives and new involvement. To balance the above-mentioned motivations, a chatbot must be developed in such way that it serves as a tool, a toy and a friend all simultaneously [24].

Chatbots are becoming increasingly popular among organizations for many different kinds of reasons, including decreased expenses for customer service and the ability to handle multiple clients independently.

1.3.1.3 Brief History

Alan Turing introduced the Turing exam (”Can computers think?”) in 1950, and it was around this time that the concept of a chatbot gained popularity. Eliza was the initial well-known chatbot founded in 1966. Its goal was to operate as a psychotherapist by returning the user’s utterances in the form of a question. It used primitive pattern matching and a template-based response system. Its conversational ability was limited, yet it was enough to perplex people at this times. As an update to ELIZA, a chatbot with an appearance named PARRY was created in 1972. ALICE, a chatbot established in 1995, won the Loebner Prize, an annual Turing Test, in 2000, 2001 and 2004. It was the initial machine to be known as the ”most human computer”. ALICE depends on a basic pattern-matching algorithms with inherent intelligence relying on AIML (Artificial Intelligence Markup Language). The next step was the creation of AI-powered personal assistants such as Siri from Apple, Microsoft Cortana, Google Assistant, Alexa from Amazon, and IBM Watson [25].

ChatGPT, on the other hand, is a big language model built by OpenAI that generates human-like replies to text-based questions using the most recent deep learning methods. It was trained using a vast dataset of online content and can answer a wide range of queries on a wide range of topics.

1.3.1.4 Chatbot types

According to [26] chatbots are classified based on several factors, including the knowledge domain, the service offered, the goals and the answer generating mechanism. As illustrated in the following figure 1.2.

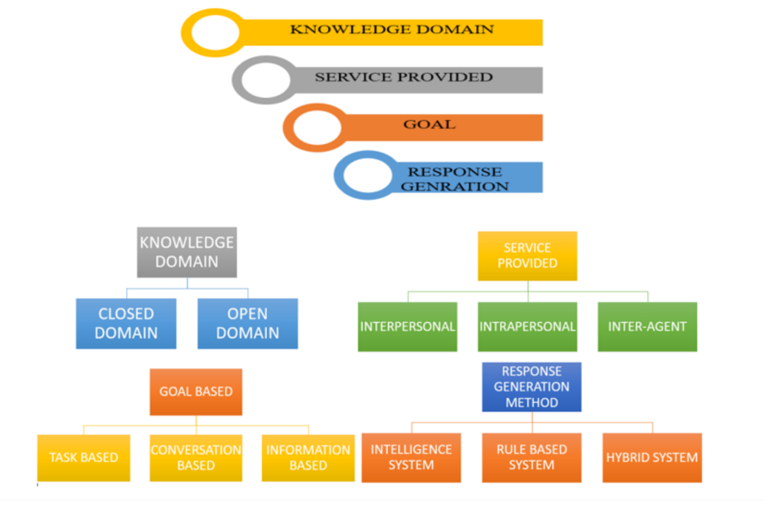


Figure 1.2: Chatbots types [27]

- **Classification based on the knowledge domain**
  - **Open Domain :**Chatbots may converse about general themes and answer properly.
  - **Closed Domain :**Chatbots have limited knowledge subject and might not be able to respond to other queries.
- **Classification based on the goals**
  - **Informative :** Chatbots like those called FAQ bots, are designed to provide users with information that has already been preserved or is accessible from a specific source.
  - **Chat-based/Conversational :**Chatbots speak with users in the same way that humans do, and their purpose is to appropriately answer to the language they are given.

  - **Task-based :**Chatbots provide a specific function, for example reserving a flight or supporting someone. These chatbots are capable of asking information and understanding the consumer's opinions. Restaurant booking bots and FAQ chatbots are examples of task-based chatbots [28] .

- **Classification based on the service provided**

  - **Interpersonal :** Chatbots are conversational bots that offer support such as restaurant bookings, flight reservations and FAQ bots. They are not the user's friends, but they collect information and relay it to the user.
  - **Intrapersonal :**Chatbots like chat systems example Messenger, Slack, and WhatsApp, exist in the user's private domain. They accompany the individual and perceive the client in the same manner as humans do.
  - **Inter-agent :**Chatbots will become ubiquitous, and all chatbots must have inter-chatbot interaction features. Protocols for inter-chatbot interaction are already necessary. Alexa-Cortana connection exemplifies inter-agent communication.

- **Classification based on response generation method**

  - **Rule-based :** Most of the initial chatbots, including numerous internet chatbots, were built using this model of chatbot design. They prefer the system answer based on a predefined set of criteria that detect the lexical form of the input text without producing new text responses. Humans hand-code the chatbot's material, which is arranged and displayed according to chatting patterns [29].
  - **retrieval-based :**Quite distinct from the rule-based model it increases flexibility by searching and analyzing API-accessible resources [30]. A retrieval-based chatbot collects some kind of response choices from an index before utilizing the matching approach for responding choice [31].

– **generative :**This model provides responses that are superior to the other three models based on immediate and past user contacts. These virtual assistants are more human-like in appearance and use artificial intelligence algorithms and deep learning techniques. However, developing and training them presents challenges [30].

1.4 Problem Statement: Customer support

Businesses are increasingly switching from traditional to digital platforms to communicate with clients as a result of the steady advancement of technology.

This section demonstrates how your business may utilize our challenge to operate more effectively while enhancing client loyalty and customer satisfaction.

1.4.1 Definition

Helping customers who have queries, challenges or problems with a product or service is known as customer support. There are several ways to accomplish this operation including phone, email or chat.

One of the most expensive aspects of any business is customer service because of the huge number of clients so it became challenging for businesses to offer expected answers to humans routine questions.

This often requires a large team of support agents to be available around the clock, which can be costly and time-consuming.

As a result of the extraordinary advancements in artificial intelligence, a growing number of companies automate their operations and switch out human specialists for innovative technological solutions that benefit not just customers but also the whole industries.

1.4.2 Benefits of the Chatbot in Customer support system

These are the arguments for implementing customer support chatbots.

Chatbots can respond to humain requirement in only a few seconds, while human operators would find it challenging. Chatbots may learn and get better with every discussion by carefully evaluating the user's input using Natural Language Processing (NLP), Machine Learning (ML), and Artificial Intelligence (AI). Chatbots will be offering accurate replies to improve the client experience as each discussion becomes more human-like [32].

In addition to assisting customers, a chatbot may represent a company and its offerings as a brand ambassador. Potential clients will begin identifying your chatbot with the company if you give it a name or personality, and perhaps even a face. Live examples of this technique include Siri and Alexa [32].

AI chatbots can assist to increase satisfactions and experience by responding to consumer's inquiries regarding their purchases. They may ask the AI chatbot questions about warranties, make requests for information about technical problems, and it will send them to the appropriate resource for an answer by ensuring that customers can always get what they need, this helps to increasing customer satisfaction [33].

With good service, chatbots may reduce customer support charges by up to 30%. It can handle several tickets concurrently without any waiting time while controlling expenses. Chatbots may constantly and dynamically progress. They are trained to respond and interpret consumer inquiries [33].

AI chatbots are being utilized more commonly in customer service because they can respond fast and effectively to simple frequently asked inquiries. As a result, employees have more time to work on improving their abilities and taking on tasks that are more valuable to the business and its consumers [32].

1.5 Releted Works

Sinha et al [34] Create a chatbot that may assist users in generating income by helping them with their educational needs. Users will find this chatbot paradigm to be engaging, scalable, and friendly. Its primary purpose is to assist other course-related activities while automating and streamlining manual and administrative procedures. They applied a machine learning method in their study instead matching keywords or words with similar meanings from a predetermined database. They used K-means, a well-known unsupervised clustering technique, to create this instructive chatbot. K-means assists in grouping data and extracting related information that aids in predicting a certain response to a query submitted by the user.

Dharani et al [35] Discuss the problem with public transportation that passengers confront. People frequently travel, and sometimes they could feel entirely disoriented in a foreign location. An interactive chatbot that provides all of the information about the buses and their times of operation, making it simpler for passengers to utilize or converse at any time or location. The importance of chatbots for customer's daily lives is one of the primary things. This chatbot was created using an RNN (Recurrent Neural Network) algorithm named "LSTM" that uses text classification to identify the category to which the user's message belongs and then selects an arbitrary answer from a database of previous responses.

Ouerhani et al [36] this work presented SMAD which is a smart assistant that provides assistance during and after a medical emergency based on deep learning sentiment analysis. They aim to identify and estimate the user's emotions when interacting with our assistance. The primary goal of SMAD is to aid normal individuals to physically and emotionally confront an emergency scenario. When bad behavior happens and occurs in a continuous discussion, SMAD will send out a questionnaire based on established scales to identify the possibility that the inappropriate behavior may be related to a psychological disorder or not. This chatbot was created using an SVM algorithm for the intent predictions and LSTM neural network was applied for sentiment analysis.

1.5.1 Synthesis

We compare the listed works that are offered in this chapter in this part. This comparison is based on six major criteria, including: domain covred by this work, objective , platform, the chatbot type, method used and language that supported by this chatbot.

	Sinha et al. [34]	Dharani et al. [35]	Ouerhani et al. [36]	Our work
Domain	Education	Transport	Medical	Customer support
Objective	Help student for their educational needs	Provides all the information about transportation	Identify the possibility of the inappropriate behavior	Helping customers with there needs
Platform	Online platform	Online platform	Application	Responsive Web App
Chatbot type	Rule-based chatbot	Conversational chatbot	Rule-based chatbot	Conversational chatbot
Method	K-means	LSTM	SVM and LSTM	Transfer learning BERT for intention and Glove-LSTM for sentiment
Language	English	English	English	English/Arabic French

Table 1.1: Comparison of the related work

1.6 Conclusion

In this chapter, we covered the concepts of human-machine interaction and case-based reasoning. Then we discussed the limitations of CBR, which led us to use the chatbots and we explored the motivations for using this chatbots. Additionally, we looked at previous research and compared it with our own work. In the next chepter We'll talk about how NLP techniques and deep learning algorithms are being used to create our chatbot.

Chapter 2

Natural Language Processing methods

2.1 Introduction

In recent decades the relationship between man and machine has become increasingly close so that the data provided by humans is experiencing a meteoric rise. These data can have different sizes and several formats textual, sounds and images...etc .This phenomenon has prompted scientists to look into the importance of these data and make them more understandable to the machine, hence the need to create natural language processing NLP.

Therefore NLP plays a fundamental role in the interaction between man and machine because this field of artificial intelligence allows the machine to understand and communicate with humans as naturally as possible. However, human language is enigmatic and in constant evolution and because of that NLP is in real challenge to adapt.

2.2 Definition

Natural language processing (NLP) is described in a variety of ways, [37] claims that Natural language processing is defined as a sub-field of artificial intelligence that studies computer manipulation of natural language text or speech. This includes algorithms that create text with a natural look and algorithms that use text written by humans as input.

To put in simple terms, it is frequently defined as software that automatically tries to manipulate natural language, such as speech and text [38].

According to [39] The study of natural language processing aims to understand how computers and human languages interact with each other, with a particular focus on how we can train computers to deal with and evaluate huge quantities of natural language data.

This artificial intelligence discipline employs a variety of computational techniques to enable computers to access human language. It also enables computers to comprehend and produce human language [40].

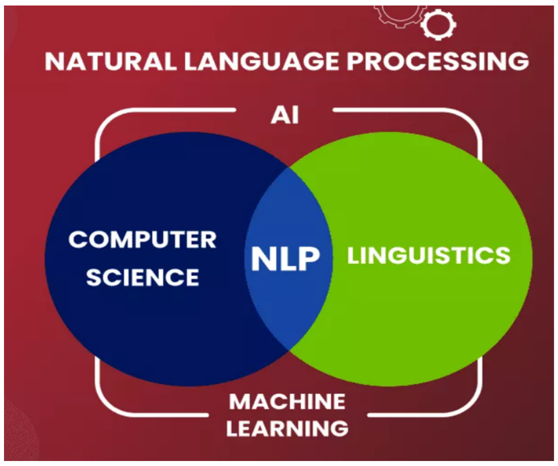


Figure 2.1: Natural language processing [41]

2.2.1 Challenges of Natural Language Processing

Many difficulties confront Natural Language Processing (NLP), such as:

- **Ambiguity:** The issue of ambiguity is one of the main problem in NLP. This is due to the fact that many words have many meanings and contexts, making it challenging for NLP algorithms to comprehend texts correctly. For instance, the term "bank" might be used to describe a business, a riverbank, or a tilt. To overcome this obstacle, more advanced algorithms and models that can comprehend the context in which words are used must be created [42] .

- **Slang and Colloquialisms:** Colloquialisms and other forms of informal speech are abundant in natural language, making it challenging for NLP systems to comprehend.

This is because the grammar and syntax used in these kinds of languages frequently depart from the norm, making them more challenging to interpret. To meet this problem, models that are explicitly trained on informal language must be created. Moreover, NLP algorithms must take into account social and cultural context [43].

- **Data sparsity:** To effectively represent language, NLP algorithms need a lot of high-quality data.However,effective NLP based systems, might be challenging to construct for specific languages or topics due to the scarcity of such data. The creation of strategies for data augmentation as well as the inclusion of other data sources like social media and online forums are necessary to meet this issue [44].

- **Multilingualism:** NLP based systems must be capable of processing and should also comprehend numerous languages, which can be extremely difficult owing to variations in grammar, syntax and structure. The creation of multilingual models that can identify and understand language across several languages is necessary to solve this problem, as is the inclusion of linguistic knowledge in the model-development process [45].

2.2.2 NLP Applications

NLP has grown dramatically in recent years as a result of the rising availability of vast volumes of textual data and the demand for more complex and human-like interactions between technological devices and humans. The objective of NLP is to create algorithms and models that can comprehend, synthesize and customize human speech in a way that is both accurate and natural. The significance of NLP resides in its capacity to revolutionize the way both people and machines communicate, allowing for more natural and human-like communication. This has several practical applications in domains like information retrieval, sentiment analysis, machine translation, question answering and other [46].

2.2.3 NLP with Machine learning and Deep learning

Machine learning is a field of computer algorithms that is still under development and seeks to replicate human intelligence by learning from its natural environment. They are considered the workhorse in today's era of so-called big data. Pattern recognition, computer vision, and natural language processing are just a few of the fields where machine learning techniques have been successfully used [47].

A subset of machine learning is deep learning. It focuses on artificial neural networks, or algorithms, which are modeled after the structure and function of the human brain [38]. It refers to a class of learning techniques known as neural networks, which may be defined as the learning of parameterized differentiable mathematical functions. Neural networks were historically motivated by the way that computing occurs in the brain. The term "deep learning" refers to the chaining together of multiple layers of these differentiable functions [48] .

Deep learning provides an opportunity to replace existing linear models in NLP with more effective models that can recognize and take advantage of non-linear correlations. Deep learning techniques are producing excellent outcomes that are highlighted by [49] in his introduction to neural networks for NLP experts. Recently, neural network models have begun to be used to analyze textual natural language data, and the results are highly encouraging.

Neural networks offer an impressive learning mechanism that is highly appealing for usage in natural language issues.

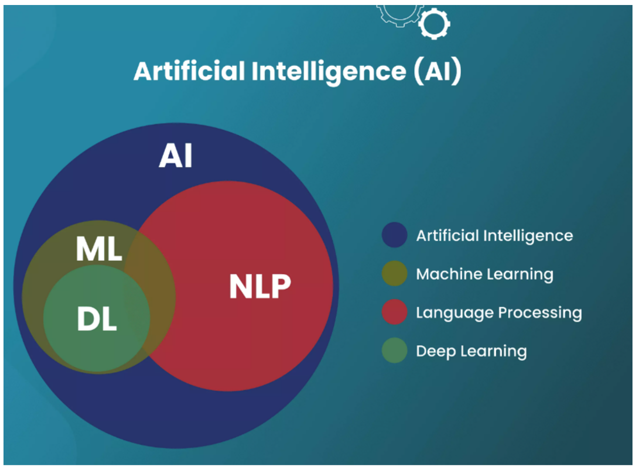


Figure 2.2: NLP with machine learning and deep learning [41]

2.3 Components of NLP

The field of NLP can be split into two subfields: natural language generation and natural language understanding or linguistics.

2.3.1 Natural Language Understanding (NLU)

The field of linguistics known as natural language understanding (NLU) enables computers to interpret natural language and analyze it by separating concepts, entities, emotions, keywords ..etc . Applications for customer assistance use this technique to understand the problems that customers report verbally or in writing. In the study of linguistics, a wide range of linguistic forms including language meaning and context, are all examined [50].

However, to understand natural language, the following techniques are employed:

- **Phonology :** Phonology is a branch of linguistics that deals with the orderly ordering of sound. Phonology involves the meaningful use of sound to encode meaning in any Human language.
- **Morphology :** Refers to the study of term's structure and forms, with a focus on the smallest meaningful elements known as morphemes.
- **Lexical :** Interpret the significance of specific words.
- **Syntax :** By examining the language's grammatical structure, it shows how a sentence should be properly formed.
- **Semantic :** Finding the correct meaning of a sentence is the most crucial task at the semantic level.
- **Pragmatic :** The knowledge or information that comes from sources other than the story is the objective at the pragmatic level. It deals with the meaning of the speaker and the inference of the listener. In fact, it evaluates indirect speech phrases [50] .

2.3.2 Natural Language Generation (NLG)

Natural language generation, or NLG, is the process of creating understandable words, phrases, sentences from an internal representation. It is a part of NLP and has three stages: goal identification, goal planning by assessing the conditions and communicative resources available, the last is goal realization That defies explanation and it is opposite to Understanding [50]. NLG is The process of producing language outputs from non-linguistic inputs. One of the core objectives of NLG is the study of how to make computer systems create high-quality, expressive, clear and natural language output form within complex information representations in computers [51].

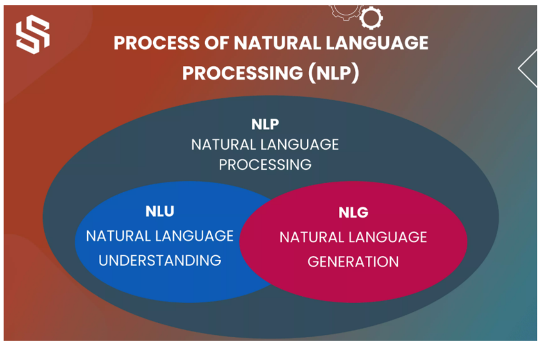

PROCESS OF NATURAL LANGUAGE PROCESSING (NLP)
NLP
NATURAL LANGUAGE PROCESSING
NLU
NATURAL LANGUAGE UNDERSTANDING
NLG
NATURAL LANGUAGE GENERATION

Figure 2.3: Components of NLP :NLU and NLG [41]

2.4 Natural Language Processing: A Brief History

In this part, we present some significant occasions that have influenced modern natural language processing.

Though the term natural language processing (NLP) didn't exist until the late 1940s but Machine translation(MT) research had already begun. In actuality, research at this time wasn't wholly rationalized. The language used for MT was mostly in Russian and English [52].

Since 1960, significant progress has been achieved in both the fabrication of prototype systems and conceptual challenges. This has mostly focused on the challenge of encoding meaning and giving computationally tractable solutions [53].

In addition to theoretical research, numerous prototype systems have been created.

Weizenbaum's ELIZA [54] which was created to simply permute or echo user input in order to recreate the interaction between a psychologist and a patient. Winograd's SHRDLU simulation [55] A robot moving sections on a tabletop demonstrated that computers are able to comprehend simple sentences. PARRY's theory [56] within a system that employed keyword groupings rather than single keywords and used synonyms in the absence of keywords.

However, by the start of the 1980s, there was an increasing demand for applications that worked with language in a broad real-word context, as a result of increasing understanding of the limitations of separated NLP solutions [57].

After then and up to the present, NLP has rapidly expanded. Since different machine learning techniques could produce good results for different NLP tasks, like modeling and parsing, they became prominent approaches to NLP ,The semantic features of words can also be captured using text embedding, machine translation,neural machine translation and deep learning [58].

2.5 NLP Techniques

NLP techniques are addressed to enable computers to comprehend and interpret human language. These techniques reach a large range of methodologies and techniques that facilitate the development of machine learning systems. In this part we present some techniques .

2.5.1 Tokenization

Tokenization is the initial step in every NLP pipeline. Tokenization means converting a character stream into a token stream. To accomplish this we need to remove all capital letters, punctuation, and brackets . Although the definition of a word is not simple, Each token can be considered a single word. A word can be defined as a collection of alphabetical letters separated by white space. Depending on the type of data to be tokenized, several policies might be used for how to split into words [59]. Figure 2.4 shows an example of

Tokenization.

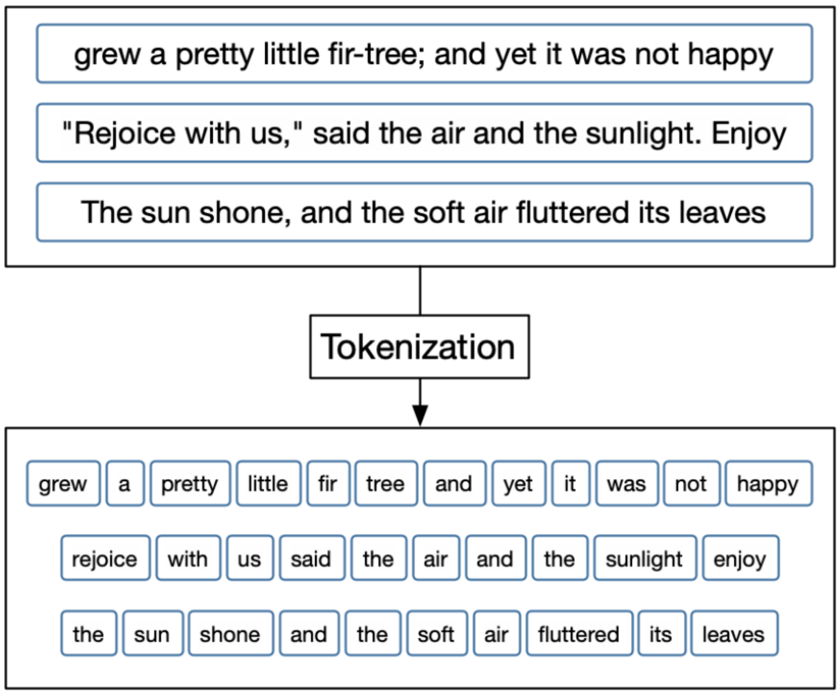

grew a pretty little fir-tree; and yet it was not happy
"Rejoice with us," said the air and the sunlight. Enjoy
The sun shone, and the soft air fluttered its leaves
Tokenization
grew a pretty little fir tree and yet it was not happy
rejoice with us said the air and the sunlight enjoy
the sun shone and the soft air fluttered its leaves

Figure 2.4: Tokenization example [60]

2.5.2 Text Normalisation (Stemming and Lemmatization)

The goal of text normalization is to determine each word's base form. It is NLP technique that aims to create root words from these word variations.

Stemming is the practice of minimizing a word down to its stem, which is also known as its root form. Stemming algorithms construct the word stem by deleting suffixes from words like (ing, ed or s). The effectiveness of search engines and text analysis can be increased by condensing words to a basic form that is prevalent. For instance, the word "jumped" would be reduced to the root "jump" as would the terms "jumping" and "jumps".

On the other hand, lemmatization means compressing a term to its basic form, or lemma, which is how it appears in a dictionary. Lemmatization takes into consideration the word's context and its part of speech. Lemmatization preserves the entire meaning of the word, this can be more accurate than stemming. The WordNet lemmatizer, which is a component of the NLTK toolkit, is one widely used lemmatization method. The WordNet lemmatizer operates by retrieving the relevant lemma for a word from the WordNet lexical database. The word ”is” would have the lemma ”be” and the word ”running” would have the lemma ”run”.

In general, the approach selected relies on the particular application and the needed level of precision. Lemmatization can be more accurate and produce greater outcomes in activities that call for a deeper comprehension of the text, even though stemming can be quicker and easier to use [61].

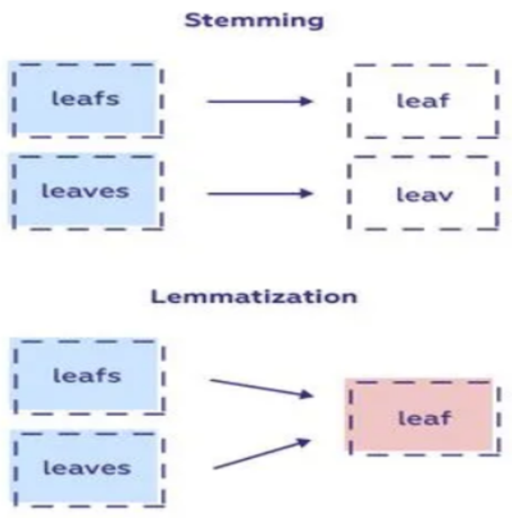


Figure 2.5: Stemming and Lemmatization Example [62]

2.5.3 Stop Word Removal

Common words like (the, that, and when, ..) don’t provide any specific information, So they’re unlikely to be useful in the similarity analysis. These words exist in all texts with almost the same frequency and have no effect on the content. Although their semantic importance, they could interfere with the calculation of similarity if they are not deleted. Prepositions, conjunctions, and pronouns make up the majority of them [59].

The primary goals of stop word removal are to decrease the dimension of the data and increase the effectiveness of text processing algorithms, such as search engines and text classification models. By eliminating stop words, the remaining words can better reflect the text’s meaning and the processing algorithms can concentrate more on the key words.

All in all, stop word removal may be a helpful pre-processing step for many jobs involving natural language processing, but it is crucial to carefully choose which words to delete and to comprehend the potential effects on the precision and interoperability of the findings [63].

2.5.4 Part-Of-Speech (POS)

Part-Of-Speech (POS) is an essential aspect of NLP, which requires identifying the appropriate part of speech to each word in a corpus of text. Nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and interjections are among the components of speech.

Numerous NLP tasks such as text categorization, Named entity identification (NER), sentiment analysis, and others, depend on POS tagging. A machine learning algorithm can better comprehend the grammatical structure of the text and extract helpful information from it by recognizing the part of speech of each word in a phrase.

In general, POS tagging is a key NLP function that enables a machine to comprehend the grammatical structure of text and extract valuable information from it [64].

2.5.5 Named Entity Recognition NER

The problem of locating and identifying important and appropriate nouns in a text is referred to as named entity recognition (NER). For instance, It is typical for names of persons, organizations, and locations to be meaningful in news reports. Named entity recognition has a significant impact on applications like machine translation, question answering and information extraction. The identified named entities in the following example contain important data and are helpful for applications involving language processing [65].

Figure 2.6 shows an example Named Entity Recognition.

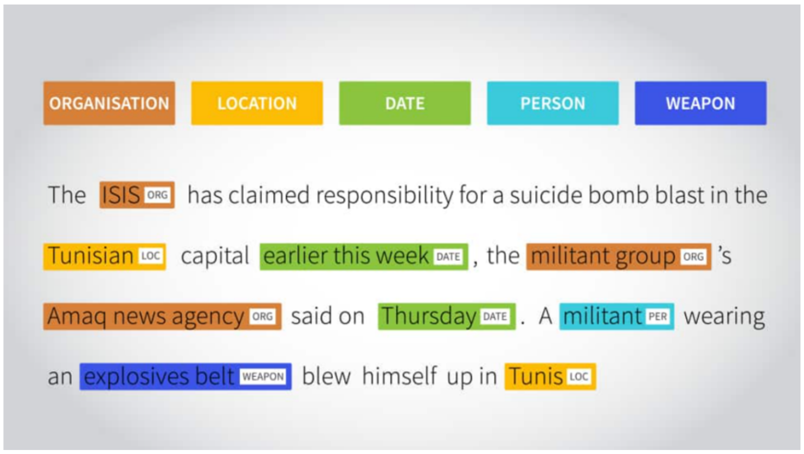
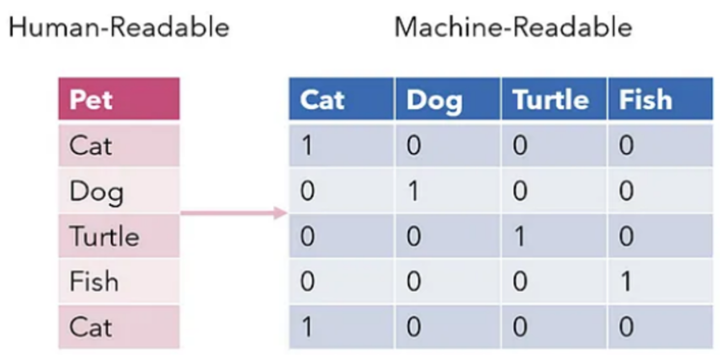

ORGANISATION
LOCATION
DATE
PERSON
WEAPON
The ISIS ORG has claimed responsibility for a suicide bomb blast in the Tunisian LOC capital earlier this week DATE, the militant group ORG 's Amaq news agency ORG said on Thursday DATE. A militant PER wearing an explosives belt WEAPON blew himself up in Tunis LOC

Figure 2.6: Named Entity Recognition Example [66]

2.5.6 One Hot Encoding

The categorization of texts refers to the difficulty of categorizing text material based on its content. Considering the fact that categorical data is regularly encountered in data science and machine learning difficulties, it is generally more difficult to manage than numerical data. Because most machine learning models only consider numerical variables, so data pre-processing with categories becomes crucial. In order to comprehend and return useful data, we have to change these categorical variables to number.One of the most common encoding methods is one-hot encoding. If the features of the data are nominal, which means that they do not have any inherent order, we use this method of categorical data encoding. With one-hot encoding, we create a new variable for each level of a categorical feature, and each category is represented by a binary variable that can contain either 0 or 1. A value of 0 indicates that the category is not present, while a value of 1 indicates that it is present [39]. Figure 2.7 shows an example of One-Hot encoding.


Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

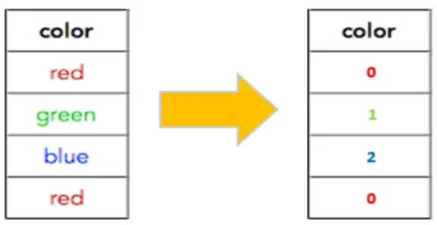
Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

Figure 2.7: Example of One-Hot encoding [67]

2.5.7 Label Encoding

Label Encoding is also commonly used method for encoding categorical variables. This method assigns a unique numerical value to each label based on its alphabetical order. It is used when the categorical feature has an inherent order, also known as an ordinal feature. In such cases, preserving the order of the categories is important, and this encoding method ensures that the sequence is reflected in the numerical values assigned to each label. Each label is converted into an integer value during the label encoding process [39].

Label encoding is an easy and effective method for converting categorical data into numerical values, but it must be handled with care and in accordance with the data's specific characteristics. Figure 2.8 shows an example of Label Encoding

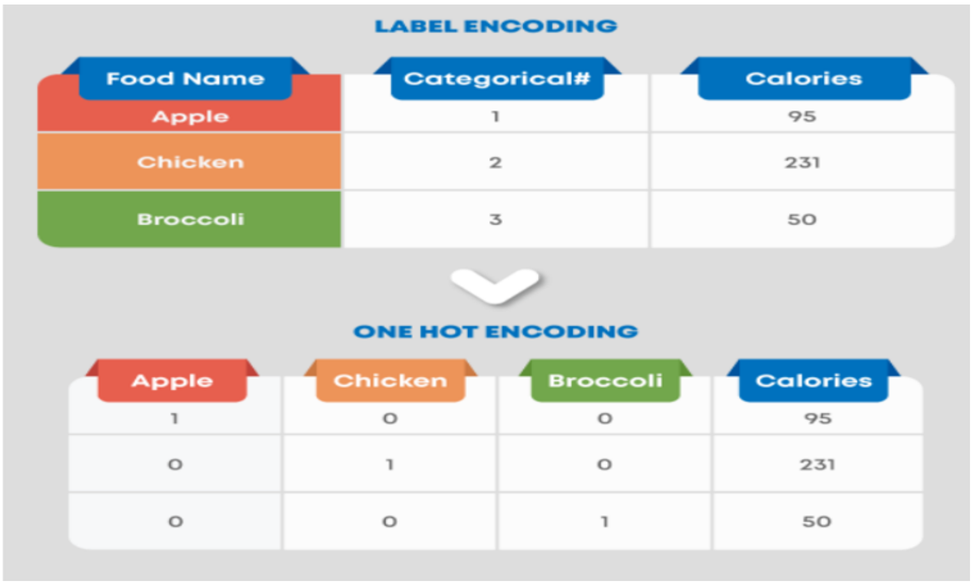


color
red
green
blue
red

color
0
1
2
0

Figure 2.8: Label Encoding example [68]

Figure 2.9 difference between label encoding and one hot encoding.



LABEL ENCODING

Food Name	Categorical#	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

ONE HOT ENCODING

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Figure 2.9: difference between label encoding and one hot encoding [69]

2.5.8 Topic Modeling

In the field of computer science, text mining and information retrieval have seen a huge growth in the use of "topic models" which is a popular class of probabilistic generative models. This models has drawn a lot of interest from academics in a variety of fields since it was first developed.

We first provide a diagrammatic explanation of the fundamental concepts underlying topic modeling in order to better understand how to use it. The topic modeling process essential steps, including the bag of words (BoW), model training and model output, are shown in this Figure 2.10. We begin by assuming that a corpus has N documents, V words, and K subjects [70].

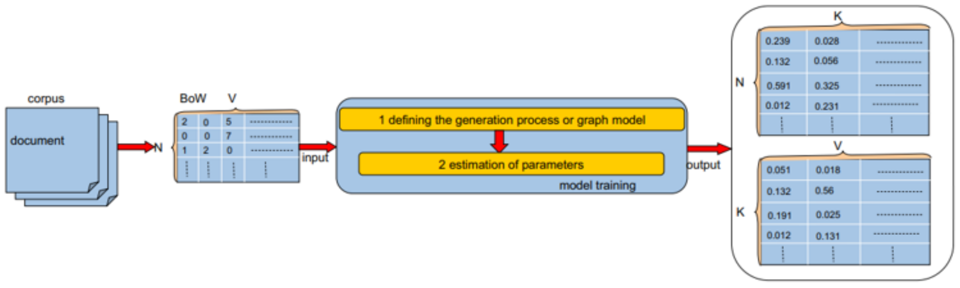
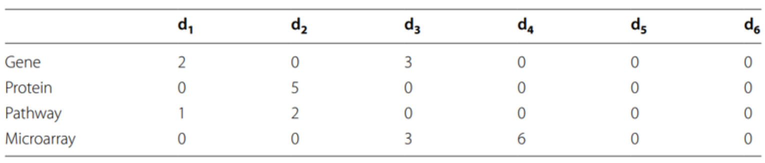


Figure 2.10: The diagram of topic modeling [70]

In natural language processing, a common way to represent a document is through a Bag-of-Words (BoW) model, which is essentially a word-document matrix. The table below provides an example of a BoW model. The matrix contains four words (gene, protein, pathway, and microarray) and six documents (d1 to d6). Each value in the matrix, denoted by w(i,j), represents the frequency of word i in document j. For instance, w(3,1) = 1 indicates that the word "pathway" appears once in document d1.[70]



	d_1	d_2	d_3	d_4	d_5	d_6
Gene	2	0	3	0	0	0
Protein	0	5	0	0	0	0
Pathway	1	2	0	0	0	0
Microarray	0	0	3	6	0	0

Figure 2.11: An example of a BoW [70]

Many techniques, like the Correlated Topic Model, Latent Dirichlet Allocation LDA and Latent Sentiment Analysis, can be used to represent a text's subject. The Latent Dirichlet method is the most used one. Using this method, distinct subjects are extracted from the text after it has been analyzed and broken down into words and sentences. All you have to do is provide the algorithm with a text body, and it will handle the rest.

2.5.9 Word of Cloud

Word of Cloud is used to graphically represent the textual data as a collection of words. The word cloud is very simple to comprehend and interpret. The term "word cloud" refers to a straightforward and efficient visualization and summarizing tool. It has applications in text mining, visualization approaches and contextual data. A word cloud can be effective for focusing on the demands and issues of clients and therefore increasing the profit of business without the need to read the text.Researchers can utilize word clouds to rapidly understand qualitative data. The more often occurring words stand out, both Font size and word count increases as the word count increases. Using word clouds to analyze user comments posted on social media about a particular product, service, political party or any other issue can help to quickly understand the overall sentiment or meaning of the comments without the need to read through each one individually. This approach can save time and effort while still providing valuable insights into the user's opinions and attitudes towards the topic at hand [71] .

Figure 2.12: cloud of words Example

2.6 Transformrs and NLP

With the development of Transformers, the area of Natural Language Processing (NLP) has seen phenomenal transformation. Transformers are a form of neural architecture or a type of deep learning model. They are highly parallelizable and can handle variable-length input sequences, making them ideal for NLP applications where the length of the input text might vary greatly. Transformers do not use sequence-aligned RNNs or convolution to construct representations of their input and output. Instead, they rely exclusively on self-attention [46].

Self-attention enables the model to focus on some elements of the input data while disregarding others in order to accomplish the problem at hand more effectively. It connects distinct points in a single sequence to compute a representation of the sequence. The self-attention mechanism enables the inputs to interact with each other in order to choose who should receive greater attention [72].

Large volumes of data and an approach known as pre-training are used to train transformers for NLP work. The model is pre-trained on a huge corpus of text data utilizing unsupervised learning methods to discover broad language patterns and word associations.

This pre-trained model may then be fine-tuned using algorithms for supervised learning on a smaller specific dataset to learn the particular task at hand. Fine-tuning entails modifying the parameters of the pre-trained model to maximize its accuracy on the task-specific dataset. The fine-tuning method may be done several times with different datasets to enhance the performance of the model on a range of NLP tasks [73].

Since the start of Transfer Learning(Transformer), it has been highly popular in NLP, and yet Google (BERT, Transformer-XL, XLNet), Facebook (RoBERTa, XLM), and even OpenAI (GPT, GPT-2) have begun to pre-train their own model on very big corpora.

In the end, the area of natural language processing (NLP) has made great progress as a result of the development of multiple transformer structures, which have been shown useful in resolving long-term dependencies, scalability and adaptability across varied workloads [46].

2.7 Sentiment Analysis

Sentiment Analysisis is a mental procedure for releasing the user's sentiments and emotions. It is a branch of Natural Language Processing (NLP) that is being explored. It is a procedure of determining the text's context-sensitive polarity. It decides if a particular piece of text is positive, negative or neutral. It is also known as opinion analysis since it gets the speaker's point of view or attitude [74].

A huge amount of study is being conducted in the subject of sentiment analysis because to its importance for business level competitiveness and changing people's requirements. Sentiment analysis needs the use of a training set to work well, and the quality of the training set is critical to obtaining an accurate assessment of the text [74].

By integrating deep learning with sentiment analysis, AI bots may give a deeper understanding of customer sentiment and help companies take intelligent choices about how to enhance the customer interaction. This methods examine text for indications about positive or negative sentiment and connect them with particular features of the text to learn what customers think about a brand or issue.

Sentiment analysis may be utilized to track real-time customer feedback, allowing enterprises to respond swiftly to disagreeable experiences and fixed problem before they get worse.

Another way that the chatbot can use deep learning and sentiment analysis to ensure customer satisfaction is when a client sends a review with a positive sentiment, the chatbot can answer with thankfulness, reinforcing the nice experience. Based on this positive review the system could provide recommendations for items or services that are likely to be of interest to customers. Customers may be more satisfied and loyal because clients believe the AI bot knows their needs and can provide customised recommendations.

If the client's inquiry has a negative emotion, the AI bot should answer in a way that respects the customer's feels while also offering a solution or an opportunity to fix the issue. The bot must avoid acting defensive or disrespectful of the customer's unfavorable feeling and instead it need to try sympathizing with their dissatisfaction.

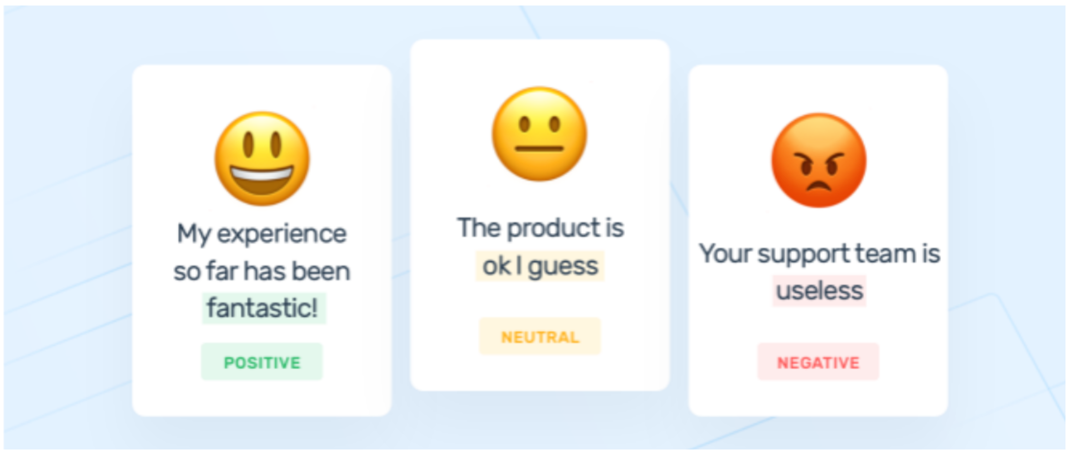

My experience so far has been fantastic!
POSITIVE
The product is ok I guess
NEUTRAL
Your support team is useless
NEGATIVE

Figure 2.13: Sentiment Analysis Example

2.8 Conclusion

In this chapter, we have demonstrated the idea of natural language processing (NLP), its applications and its relationship to machine learning deep learning and its methods for ensuring easier data processing. Additionally, we have observed that the integration of transformers and sentiment analysis in NLP enables the improvement in performance of our chatbot. In the next chapter, The design of our system and the Deep Learning algorithms we utilize and the techniques for NLP are all going to be explored.

Chapter 3

Design and Contribution

3.1 Introduction

Previously we studied the concept of natural language processing NLP and its methods which allowed us to design our system. Our goal now is to make a web chatbot application that can assist the customers using deep and transfer learning and NLP methods to understand the user intention and classify his sentiment in order to give him the appropriate response.

Now we go through the system architecture we've suggested and its detailed description, in addition provide algorithms used and NLP strategies we've employed to accomplish our objective.

3.2 Proposed architecture

Our goal is to create a retrieval-based chatbot employing transformers models, especially BERT for intention classification and Glove-LSTM for sentiment classification. We will now introduce our system architecture, which is composed of three distinct levels for separate functions.

The user interface, which is the initial layer, is where the user input their request. The treatment layer then receives the user's text and converts it to English after identifying the user's language. After that, we use NLP pre-processing and extract the intent and sentiment of our customer from the user text. Moving on to the last layer, known as the data source layer,is where the replies are found. The primary purpose of this layer is to find the appropriate response depending on the intention and emotion of the user. The obtained response appears in the user interface in his native language.

The architecture is depicted in Figure 3.1.

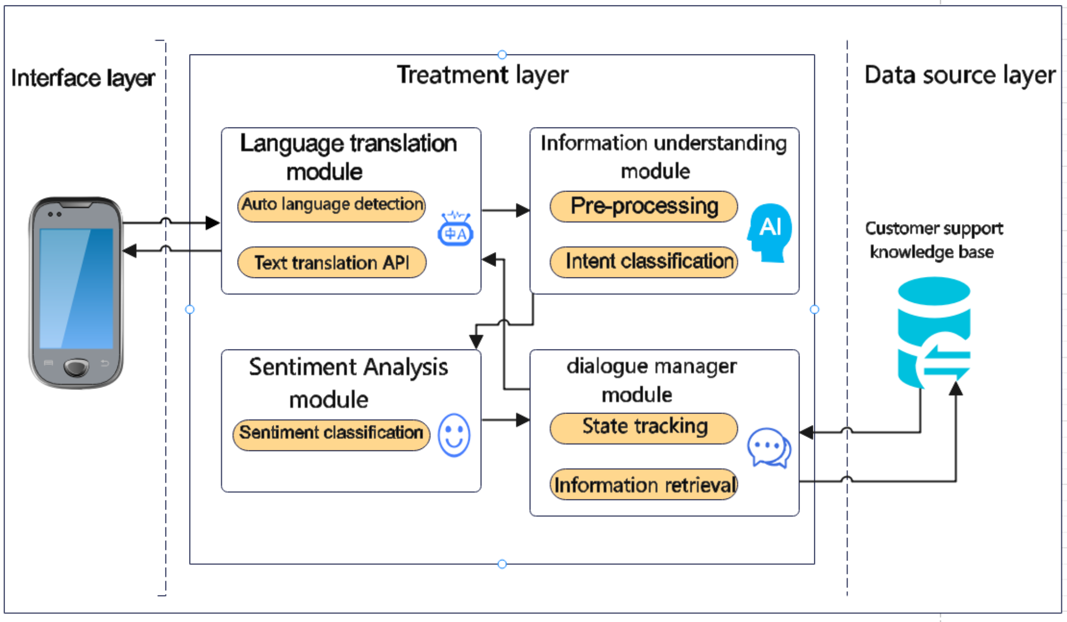


Figure 3.1: Our General architecture

3.2.1 Architecture description

This section provides a detailed explanation of the function and significance of every separate element.

1. **User interface layer**

   This layer is the beginning of interaction between the customer and the chatbot.It acts as a bridge between the user and the chatbot, facilitating users to get replies from the chatbot. It includes an input text bar where users may type their inquiries or messages. This input is then forwarded to the chatbot's processing layer. In addition, the interface layer contains a swiping window that organizes and shows the complete discussion among the client and the chatbot. It guarantees that the chat history is shown in chronological order.

2. **Treatment Layer**

   This layer covers the entirety of our system,it is an essential component of the entire architecture. As it includes a broad range of functionalities and necessary methods. It is divided into four primary module :

   (a) **Language Translation module**

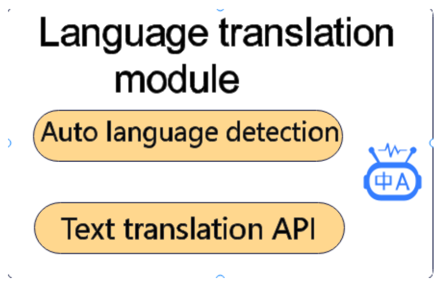


Figure 3.2: Language Translation module

This Module's principal job is to simplify the translation operation of the customer input. It uses an artificial language identification mechanism and supports three languages: English, Arabic and French. To do this, the Module uses the Google API translate server to request translations from the client's identified expression to the chatbot language. Similarly, while translating the chatbot system's responses, the Module uses the same technology to transform the answers from English to the user's identified language.

   (b) **Information understanding module**

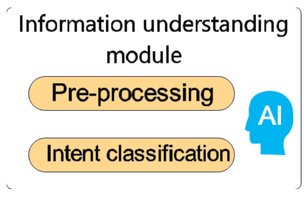


Figure 3.3: Information understanding module

This module is in charge of understanding the user's text in order to identify their purpose. This can be accomplished by the use of the following two tasks: text pre-processing and intention classification.This module's functionality is carried out after the translation process.

i. **Pre-processing**

Pre-processing is an important stage in NLP that covers transforming raw text into a more acceptable format for analysis. In this section, we apply various text pre-processing techniques such as lowercasing, stop-word removal, tokenization, adding special tokens and converting to input IDs.

ii. **Intent classification**

To predict the intent classification,our model receives the numerical representation (input IDs) in which generated using BERT vocabulary. The model then use its prediction algorithm to find the most appropriate intention based on the encoded sequence.

(c) **Sentiment Analysis module**

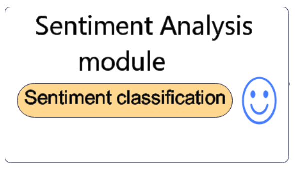


Figure 3.4: Sentiment Analysis module

To perform sentiment classification, our model takes the numerical representation.The encoded sequence is then used by our algorithm to determine the sentiment represented in the text. The model analyzes the input and categorizes the sentiment into 3 classes positive, negative or neutral, providing a full understanding of the sentiment communicated in the text.

(d) **Dialogue manager module**

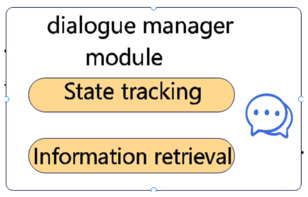


Figure 3.5: Dialogue manager module

i. **State Tracking**

Dialogue state tracking involves continually monitoring and updating the current status of a dialogue between a user and a chatbot. It assists in keeping track of essential details and context, allowing the chatbot to offer more relevant and correct replies.

ii. **Information Retrieval**

After dialogue state tracking, information retrieval comes into play in our chatbot system. Information retrieval includes extracting relevant information from the customer support knowledge base to answer to the user questions. The response will be transmitted to the translation API.

3. **Data source Layer**

This layer includes a repository of data that stores responses and information about services and subjects. The database provides the required data for the chatbot to provide relevant replies for users, which makes it a useful source of information for the chatbot.

3.2.2 Customer Assistance system

In this section, we will provide a step-by-step guide to creating our chatbot system. Once the chatbot is created, we will utilize it to answer user's requests. The suggested chatbot system follows the processes specified in the flowchart below to create relevant and meaningful answers depending on the user's input.

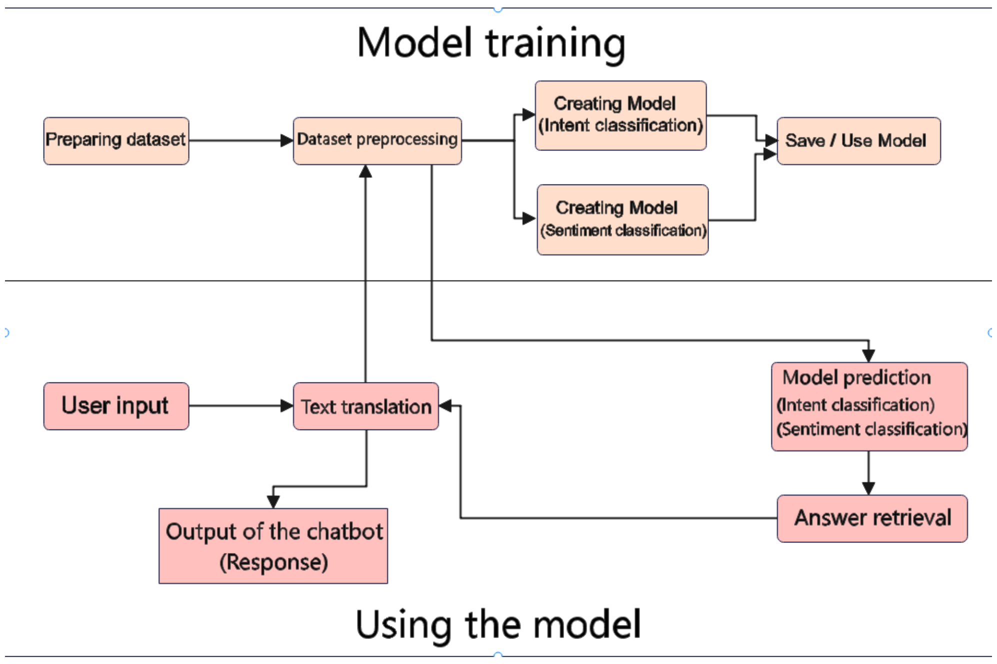

Model training
Preparing dataset
Dataset preprocessing
Creating Model (Intent classification)
Creating Model (Sentiment classification)
Save / Use Model
User input
Text translation
Model prediction (Intent classification) (Sentiment classification)
Answer retrieval
Output of the chatbot (Response)
Using the model

Figure 3.6: Customer Assistance system

1. **Model Training**

   (a) **Preparing Dataset**

   The dataset covers the ”Customer Support” domain .This dataset contains three columns:1-An utterance is a user’s input, such as a question or statement,2-the applicable linguistic flags ,3-the intent corresponding to the user utterance.

   For integrating sentiment analysis into our unlabeled sentiment dataset, we will use VADER (Valence Aware Dictionary and sEntiment Reasoner) which is unsupervised techniques for predicting the sentiment. VADER is a powerful tool that allow us to apply sentiment labels to the dataset by looking at the sentiment presented in the text. This process will involve adding a new column for sentiment classification in our dataset.

   In addition, we will include another dataset with two columns: ”review” and ”sentiment” where the ”review” column represents the input of each review and the ”sentiment” column corresponds to the defined sentiment for each review.

   (b) **Data Pre-processing**

Following the preparation of our dataset, the next critical step is pre-processing, which occurs before training our tranfert learning model. We will use pre-processing approaches designed particularly for the BERT model(intent classification),These include Tokenization,Adding Special Tokens,Padding and Converting to Input IDs.

On the other hand, for the Glove-LSTM model (sentiment classification), we will use LSTM-specific pre-processing approaches. These include, tokenization, sequence padding, and word embedding using GloVe. Later, we shall go through this strategy's specifics in more depth.

These model-specific pre-processing methods guarantee that the input data is correctly organized and optimized for training.

(c) **Creating model for intention classification**

Choosing an appropriate model is critical for accurate and successful learning.The process is separated into two steps:

- The configuration of the BERT model parameters : Number of Layers, Hidden Size, Batch Size, Learning Rate, activation functions...etc.

- Training and evaluating the model (using predictions to put the model to the test).

We will go through the detail of this approach in further detail later.

(d) **Creating model for sentiment classification**

Creating a sentiment analysis model entails two major phases as well.

- The configuration of the Glove-LSTM model parameters: Number of Layers, number of cells, loss, activation functions...etc.

- Training and evaluating the model.

(e) **Save/Use the Model**

Once our models is trained, we will save it for future use in predicting the intents and the sentiment of our customers in the future. Saving the model assures the continues use for accurate and consistent predictions in real-time.

2. **Using the model**

   (a) **User input**

   Once the application interface receives user input, including text data,it will be properly transmitted to backend that handle treatment.

   (b) **Text translation**

   Following the receipt of the text input, the Python langdetect module is employed to identify the language of the input. If the text is not already in English, it is translated via a query to Google API servers. The translated text is then sent to the Data preprocessing module for further preparation.

   (c) **Data pre-processing**

   After translating the user's input, we employ the previously discussed pre-processing approaches. These preprocessing techniques aid to improves our model's performance and prediction.

   (d) **Model Prediction**

   Following pre-processing, the user's input data is transformed into numerical representation (input IDs) and sent to the BERT model to classify the intention. Then the input is turned into a vector representation (word encoding) to categorize our client's emotion. The model prediction result is then given to the conversation state tracker, which retrieves the relevant answer.

   (e) **Answer retrieval**

   Based on the user's intention classified by our intention model and the user's sentiment classified by our sentiment model, the dialogue state tracker will utilize them to obtain the response from the knowledge base.

   (f) **Translate to user language**

   The chatbot's answer from the information retrieval procedure is translated to the user's determined language and provided as textual data to the backend.

(g) **Output of the chatbot**

When the chatbot gets the user's text input, it will show the response in the application's discussion interface.

3.3 Used Training models

In this section we introduce the used models that helps to train our model to construct a system that aim to responds to clients queries. As we mentioned before we have proposed two different models based deep and transfer learning models.

3.3.1 Bidirectional encoder representation from transformers

A pre-trained language model called the Bidirectional encoder representation from transformers (known as BERT) is based on transfer learning approach. This method is characterized as an innovative technique that is quickly imposing itself as the industry norm in NLP. The BERT technique is exceptional Due to it is able to identify and extract contextual meaning from a phrase or paragraph. This contextuality shows that the surrounding words affect how a word or token is represented numerically throughout the idea embedding process. We incorporate it into the processes for dynamic word embeddings. Language models are created using dynamic methods that include a text's syntax and semantics in addition to the environment mentioned above. [75]

The word BERT stands for bidirectional encoder representations from transformers, the concept of Transformers means the network architecture that is built on Transformer units. The Transformer idea depends on switching RNN (recurrent neural network) blocks in neural network design with self-attention components. The self-attention system is a method of focusing the mind that links several topics in a single phrase to calculate a representation of the sequence. Transformer encoders are utilized in the BERT and they represent the only layer of the BERT architecture [76]

The bidirectionality of the BERT is an essential feature which allows it to analyze tokens both right to left and left to right in the studied text segment. This model ensures that the token context is taken into consideration when training the neural network. By focusing on both the left and right contexts concurrently across all layers, the BERT was developed to pre-train deep bidirectional representations from unlabeled text [77].

The BERT is a structure that includes self-attention layers, these levels enable the application of an attention mechanism. This is the benefit of BERT over CNN and RNN approaches, both are not intuitive enough and so have poor comprehensibility [78]

The BERT has been accepted as the best approach for classifying texts in NLP problems.

Various improvements have been made to the original BERT architecture, which was pre-trained on multiple datasets. The initially developed BERT designs are BERT-BASE and BERT-LARGE; BERT-BASE has 12 layers, a hidden size of 768, 12 self-attention heads, and 110M total parameters .[79]

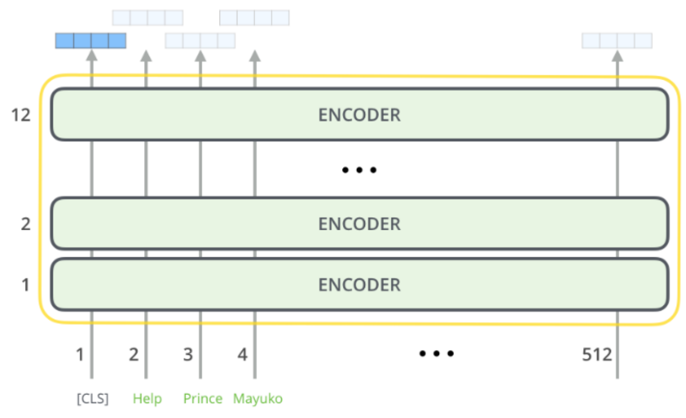


Figure 3.7: The general structure of the BERT-base from [79]

In figure 3.7 BERT receives a string of words that continue to flow up the stack. Each layer applies self-attention and sends its final result to the following encoder using a feed-forward network.

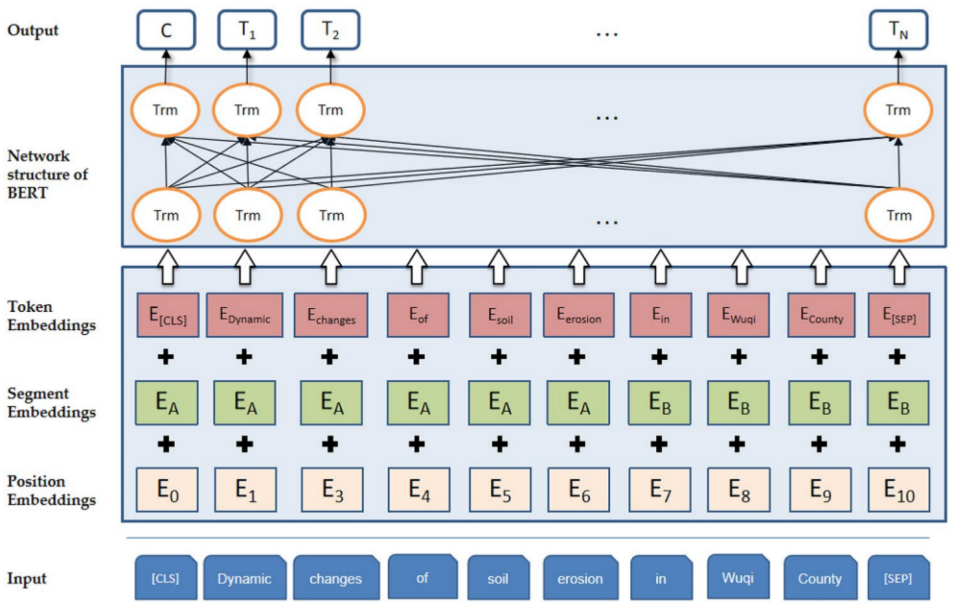


Figure 3.8: BERT training process from [80]

Figure 3.8 illustrates the BERT training process.

In figure 3.8 The encoded representation of words is called [En], the transformer structure is called (Trm) and the trained word vector is called (Tn). The BERT model receives natural input sentences. Token embedding divides the words entered into independent tokens and adds two unique symbols, [CLS] and [SEP], to indicate the beginning and final positions of the text example respectively. For separation of two phrases, segment embedding is employed. Position embedding represents a word's position information. BERT's input word vector is created by combining three vectors: token embedding, segment embedding, and position embedding [80]

One of the key benefits of employing BERT (Bidirectional Encoder Representations from Transformers) is its capacity to detect contextual information and interpret the meaning of words depending on their immediate context. In intent classification the concept of an expression or word can be strongly altered by the words around it. BERT's bidirectional training enables it to examine both the left and right context of a word while producing predictions, allowing for a more in-depth knowledge of the language.

3.3.2 Long-Short Term Memory Network

The long short-term memory (LSTM) is a form of neural network that is used in the fields of AI and deep learning. LSTM is a sort of recurrent neural network (RNN) architecture that is meant to better precisely simulate temporal sequences and their long-term relationships than traditional RNNs. LSTM has found growing popularity in a variety of domains, including voice recognition, time series prediction, and translation by machines. LSTM has a computational difficulty per time step and is spatially and temporally local. LSTM has been demonstrated to be useful in addressing memory issues in traditional RNNs [81]

The LSTM structure allows the cell position values to be forgotten and defines how many current input values will be received. Even if the procedure is repeated, the gradient does not terminate, and learning is no longer feasible. In the same manner that the typical circulatory neural network does, it analyzes the absolute output value over the number of hidden sides. Gateways are used to manage the flow of data in the process to determine the number of variables in the hidden layer. As a consequence, the circulatory neural network based on LSTM cells can accommodate data from a long sequence of procedures without experiencing slope loss [82]

To improve the word representation in our model, we will use the GloVe (Global Vectors for Word Representation) embedding approach. GloVe delivers pre-trained word embeddings that capture semantic and syntactic links from words based on their co-occurrence data in a large corpus. Using GloVe embeddings, our model will be able to encode words into high-dimensional vectors, allowing it to better capture the semantic meaning and contextual information of words in our text data. This will considerably improve the efficacy and accuracy of our model's language understanding and text analysis skills.

Figure 3.9 illustrates Basic LSTM cell structure. In Figure 3.9 it, ft, ct, and ot represent the input gate, forget gate, control gate, and output gate, respectively.

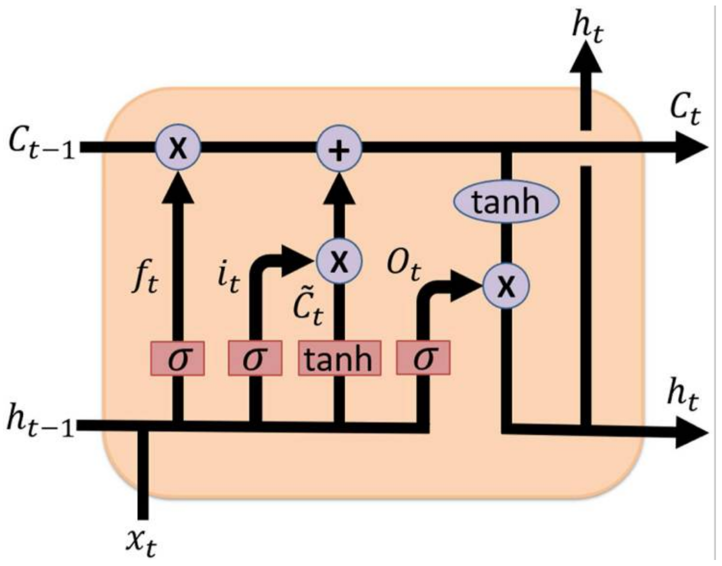


Figure 3.9: Basic LSTM cell structure [82]

3.3.3 BERT Pseudo Code

The next linsting explains the BERT code used to create our model in order respond to clients questions.

```
Split_data(data)        ->Split data 80% for train and data 20% for test
pre-porossing_data(data)->Applying NLP method to convert data into vector
reshape_data(data)    -> Reshape data according to Embedding layer input
model Create (BERT model)  ->  Creating and configuring the model
bert=BertModelLayer.from_params(bert_params)  -> create BERT model layer
input_ids = keras.layers.Input(shape=(max_seq_len)-> add input_ids(
   Input_Layer)
bert_output = bert(input_ids)  ->The input_ids tensor is passed to the
   BERT model layer
cls_out = keras.layers.Lambda(lambda seq: seq[:, 0, :])(bert_output)  ->
   this layer extract the representation of the [CLS] token
cls_out = keras.layers.Dropout(0.5)(cls_out)  ->Dropout is a
   regularization technique
logits = keras.layers.Dense(units=768, activation="tanh")(cls_out)
logits = keras.layers.Dense(units=len(classes), activation="softmax")(
   logits)
model = keras.Model(inputs=input_ids, outputs=logits)
```

```
model.compile(optimizer=Adam(1e-5),
  loss=keras.losses.SparseCategoricalCrossentropy,
  metrics=[keras.metrics.SparseCategoricalAccuracy]) -> Compile model
history = model.fit(x=data.train_x,  y=data.train_y,validation_split=0.2
      batch_size=16,epochs=5)    ->trainning the bert model
test_acc = model.evaluate(data.test_x)  ->Testing model from test data
```

3.4 Architecture of the used models

As we discussed in previous sections, the proposed transfer and deep learning use a different layers. This section present the scheme of each model (BERT and Glove-LSTM).

1. **BERT architecture**

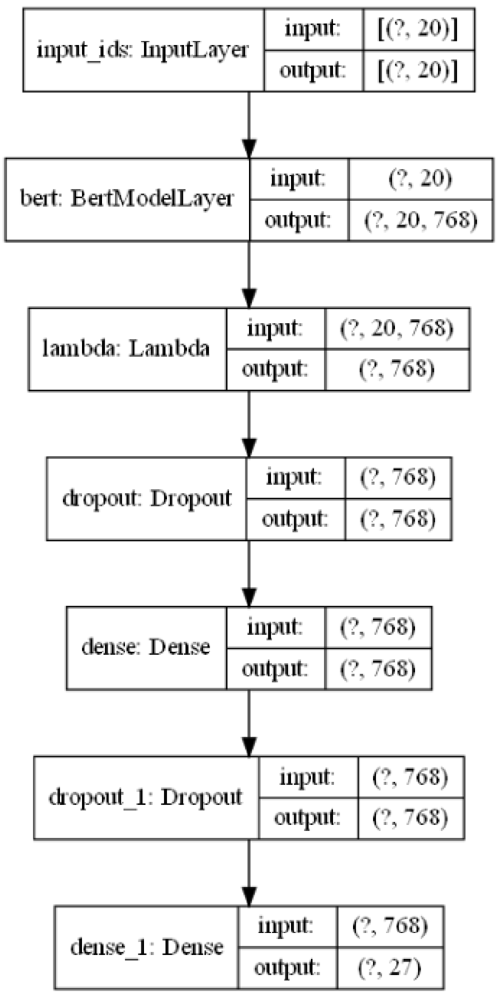


Figure 3.10: BERT model architecture and layers

The structure of the model is composed of seven layers that are implemented in the following order: the layer that receives input contains the tokens vector representation of the input text, the BERT layer for feature extraction and representation, the lambda layer in BERT is in charge of obtaining the initial token representation from the BERT model's output, a dropout layer to prevent over-fitting, a dense layer to transform the extracted features, Finally a softmax layer for the final classification output with 27 different intent.

2. **Glove-LSTM architecture**

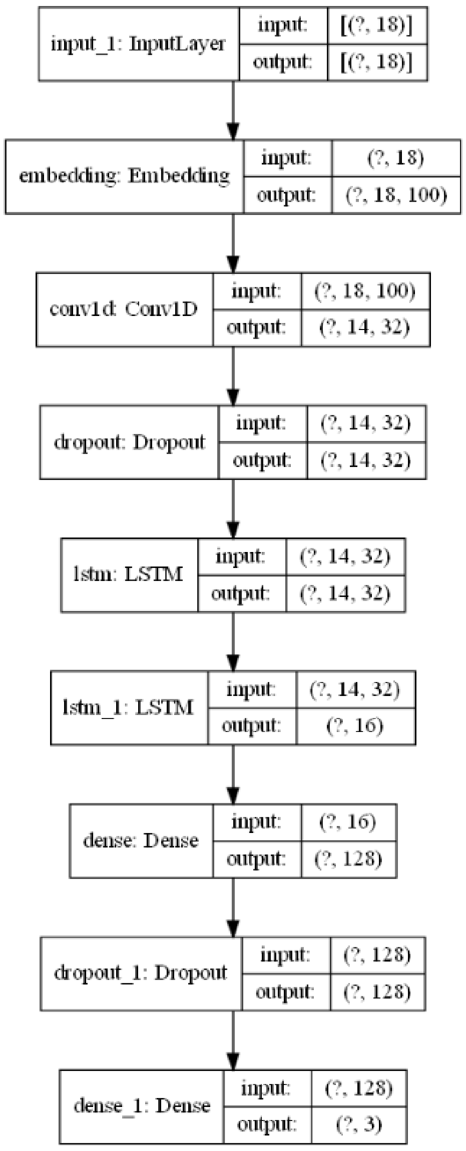


Figure 3.11: Glove-LSTM model architecture and layers

The model's structure is made up of nine layers, which are applied in the following order: the layer that receives input contains the tokens vector representation. Embedding Layer that transforms the input sequences into fixed-size vectors and it uses pre-trained word embeddings Glove. Conv1D Layer detect local patterns in the given input sequence, a dropout layer to prevent over-fitting. Two LSTM layer for feature extraction with 32 units and the second with 16 units, a dense layer to transform the extracted features, at last a softmax layer for the final classification output with 3 different sentiment.

3.5 Conclusion

After providing an overview of the system architecture, including the details of each component and their respective functions, as well as the algorithms used (BERT and Glove-LSTM) with their detailed structures. Next chapter, we will then explore the implementation of these algorithms within our system.

Chapter 4

Implementation and results

4.1 Introduction

The development of application involves a series of steps with the aim of creating a web app that can effectively understand user intentions and consider their sentiments in the business context. After exploring the theoretical aspects and the details of our system architecture, this chapter focuses on the development methodology and presents the results obtained from our models. We discuss the tools and libraries used to create our customer assistant chatbot and conclude with the presentation of the system interface.

4.2 Development environments and tools

4.2.1 Programming languages

- **Python :** Python is the open source programming language used by IT professionals the most. This language was developed with the management of infrastructure, data analysis, or software development in mind. In fact, one of Python's benefits is that it allows developers to focus more on what they are doing than how they are doing it [83].

- **Javascript :**JavaScript is an acronym for an object-oriented programming language used in computer development. The majority of the time, one finds it on Net sites.

It enables, among other things, the introduction of brief animations or effects on a website or HTML page [83].

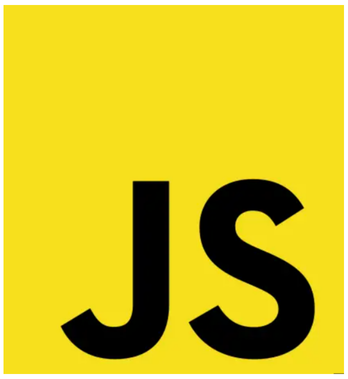


Figure 4.1: java script logo

4.2.2 Machine learning kit

- **Tensorflow :** A well-known machine learning and deep learning library . It was created by the company's Brain Team and made available as a free and open-source library on November 9, 2015. Machine learning is accelerated and made simpler by its complete reliance on the language known as Python for mathematical calculation and data processing [84].

- **Keras :**Google created Keras, an advanced deep learning API, for building neural networks. It is used to simplify the implementation of neural networks and was created in Python. Additionally, it enables various backend neural network data processing [85]

- **NLTK :**The Natural Language Toolkit (NLTK) is a framework for creating Python applications for statistically NLP that operate with data pertaining to human language It includes packages for tokenization,stemming, and other text processing operations. [86].

- **Pandas :**is a Python module designed for a variety of statistical analysis and editing tasks, including the study of tables of information, time series, and different kinds of data sets [87].

- **langdetect :** A language detector is a device which can quickly and accurately determine the spoken language of a input. This is beneficial in a variety of scenarios.
- **Google-trans :**A free and endless Python package called Googletrans uses the Google Translation API. This calls functions like detect and translate using the Google Translate Ajax API [88].

4.2.3 Frameworks and tools

- **Google Colaboratory :** is like a Jupyter notebook workspace on the cloud is known as "Colab" by most users. Everyone with an internet connection may use it when experimenting with machine learning and AI code because it executes in your web browser.Colab has high CPU and GPU workload requirements [89].
- **Jupyter notbook :**is a web-based tool that allows for interactive document creation and distribution.There are several uses for Jupyter notebooks. A notebook is an interacting computational environment where users may run a specific piece of code, view the results, and modify the code to produce the desired results or conduct further exploration [90].
- **Flask :**A well-liked tool for creating Python web apps and APIs. It gives programmers a quick and simple approach for building RESTful APIs that other programs in software may utilize [91].

4.3 Preparing dataset

As already explained in the previous chapter, We need two separate models: one for intention classification ,in which we will employ the BERT model, and another for sentiment classification,in which we will utilize the Glove-LSTM model. For that we have two dataset to prepare.

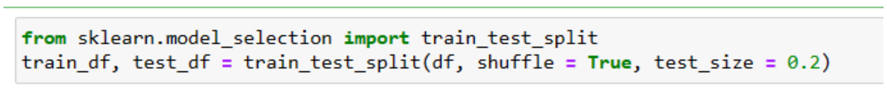
The first dataset, which was obtained from [92], is organized as a CSV file with the columns "Utterance" and "Intent". 27 unique intents are covered by this dataset.

The second dataset has two columns and was collected from [93]. It is in a CSV file structure. Text reviews are in the first column, while labels in the second column indicate the sentiment "positive","negative" and "neutral".

4.3.1 Pre-processing dataset

We will explore the pre-processing methods shared by BERT and Glove-LSTM that we are going to use.

- **Splitting Dataset**



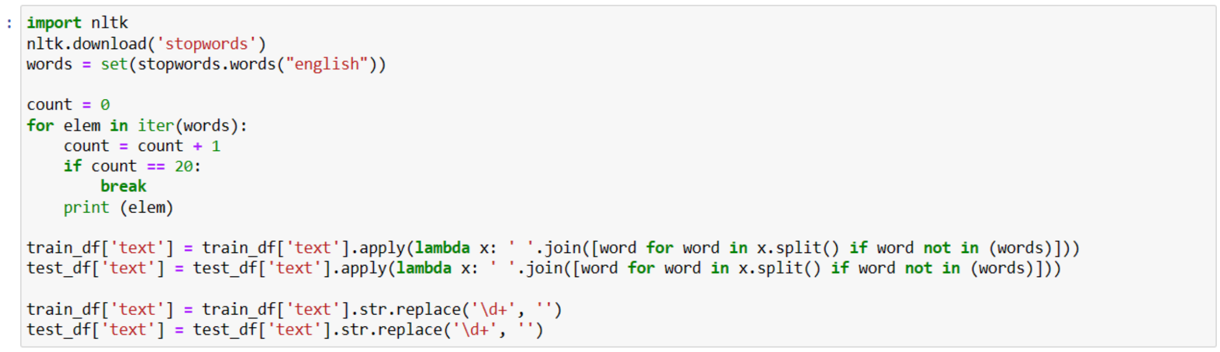
```
from sklearn.model_selection import train_test_split
train_df, test_df = train_test_split(df, shuffle = True, test_size = 0.2)
```

Figure 4.2: Splitting dataset

As demonstrated in the script below, the first step is to divide our dataset into 80% for training data and 20% for testing data.

- **Clean Dataset**

The next step is to do data cleaning by eliminating stop words, useless symbols etc...

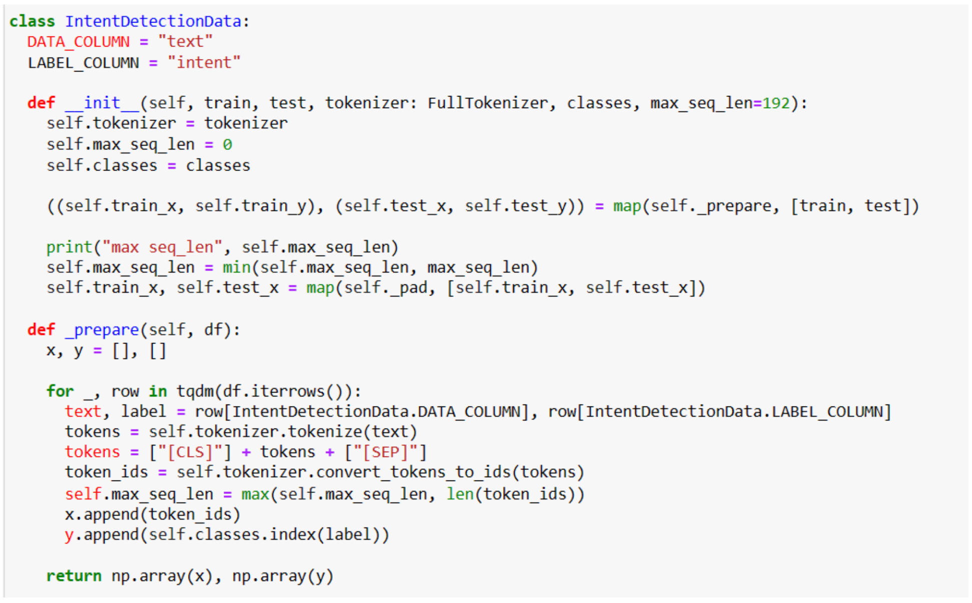


```
import nltk
nltk.download('stopwords')
words = set(stopwords.words("english"))

count = 0
for elem in iter(words):
    count = count + 1
    if count == 20:
        break
    print (elem)

train_df['text'] = train_df['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (words)]))
test_df['text'] = test_df['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (words)]))

train_df['text'] = train_df['text'].str.replace('\d+', '')
test_df['text'] = test_df['text'].str.replace('\d+', '')
```

Figure 4.3: Clean dataset function

- **Pre-processing for BERT**

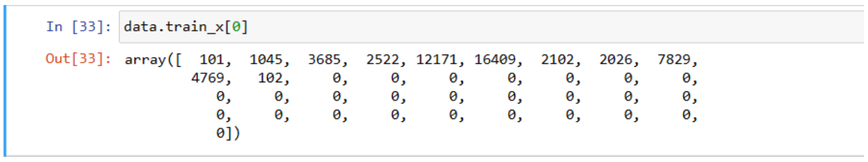
In this part, we'll employ pre-processing methods specifically designed for the BERT model's such as tokenization, adding special tokens, padding and truncating, and converting to input IDs that have the same length. These procedures are necessary to get the text data into a format that the BERT model can correctly understand and deal with. As shown in the pseudo code of the following figure 4.4.



```python
class IntentDetectionData:
  DATA_COLUMN = "text"
  LABEL_COLUMN = "intent"

  def __init__(self, train, test, tokenizer: FullTokenizer, classes, max_seq_len=192):
    self.tokenizer = tokenizer
    self.max_seq_len = 0
    self.classes = classes

    ((self.train_x, self.train_y), (self.test_x, self.test_y)) = map(self._prepare, [train, test])

    print("max seq_len", self.max_seq_len)
    self.max_seq_len = min(self.max_seq_len, max_seq_len)
    self.train_x, self.test_x = map(self._pad, [self.train_x, self.test_x])

  def _prepare(self, df):
    x, y = [], []

    for _, row in tqdm(df.iterrows()):
      text, label = row[IntentDetectionData.DATA_COLUMN], row[IntentDetectionData.LABEL_COLUMN]
      tokens = self.tokenizer.tokenize(text)
      tokens = ["[CLS]"] + tokens + ["[SEP]"]
      token_ids = self.tokenizer.convert_tokens_to_ids(tokens)
      self.max_seq_len = max(self.max_seq_len, len(token_ids))
      x.append(token_ids)
      y.append(self.classes.index(label))

    return np.array(x), np.array(y)
```

Figure 4.4: Pre-processing dataset for BERT

Figure 4.5 displays the end outcome of the pre-processing part.



```
In [33]: data.train_x[0]

Out[33]: array([  101,  1045,  3685,  2522, 12171, 16409,  2102,  2026,  7829,
                 4769,   102,     0,     0,     0,     0,     0,     0,     0,
                    0,     0,     0,     0,     0,     0,     0,     0,     0,
                    0,     0,     0,     0,     0,     0,     0,     0,     0,
                    0])
```

Figure 4.5: Output after pre-processing step

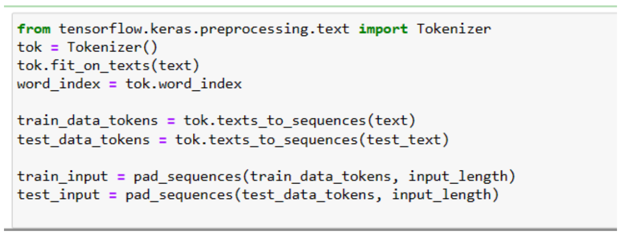
After preprocessing our dataset, now we can pass it to our BERT model for training.

- **Pre-processing for LSTM Sentiment classification**

  In this part, we'll employ pre-processing methods made specifically for the Glove-LSTM model, including tokenization, sequence padding and word embedding using GloVe.

  - **Tokenization and sequence padding**

Tokenization is the process of dividing the text into separate tokens or words. By using sequence padding, all input sequences are guaranteed to be the same length.



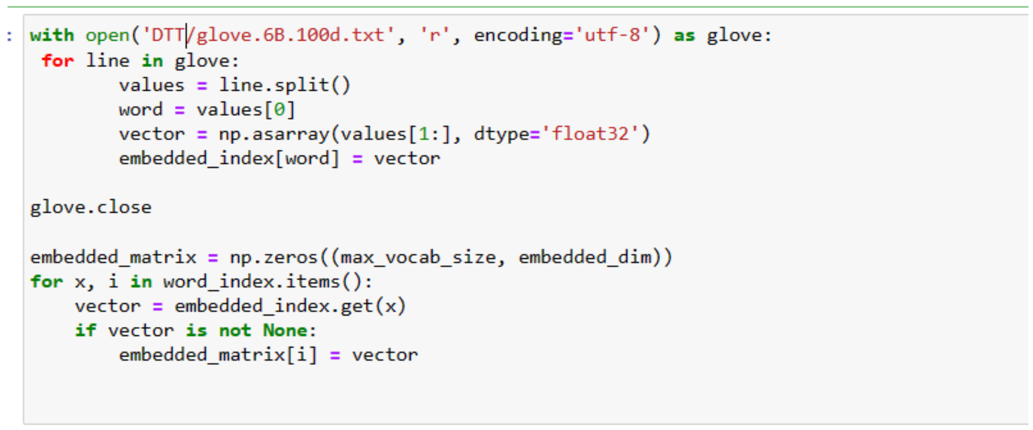
```
from tensorflow.keras.preprocessing.text import Tokenizer
tok = Tokenizer()
tok.fit_on_texts(text)
word_index = tok.word_index

train_data_tokens = tok.texts_to_sequences(text)
test_data_tokens = tok.texts_to_sequences(test_text)

train_input = pad_sequences(train_data_tokens, input_length)
test_input = pad_sequences(test_data_tokens, input_length)
```

Figure 4.6: Tokenization and sequence padding for LSTM

– **Word embedding using GloVe**

GloVe is pre-trained word embeddings, which are vector representations of words. The model can understand the meaning of words in a more complex manner because of these embeddings, which capture semantic links and contextual information.

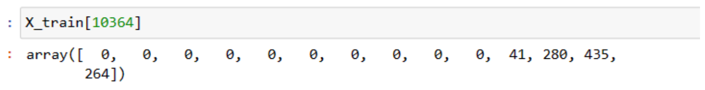
To increase sentiment classification accuracy, we use GloVe embeddings into our LSTM model architecture. This combination improves our model's performance by enabling more precise sentiment categorization.



```
: with open('DTT/glove.6B.100d.txt', 'r', encoding='utf-8') as glove:
   for line in glove:
          values = line.split()
          word = values[0]
          vector = np.asarray(values[1:], dtype='float32')
          embedded_index[word] = vector

  glove.close

  embedded_matrix = np.zeros((max_vocab_size, embedded_dim))
  for x, i in word_index.items():
      vector = embedded_index.get(x)
      if vector is not None:
          embedded_matrix[i] = vector
```

Figure 4.7: word embedding using GloVe

Figure 4.8 displays the result of the preprocessing phase for the model of sentiment analysis Glove-LSTM



```
: X_train[10364]
: array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  41, 280, 435,
        264])
```

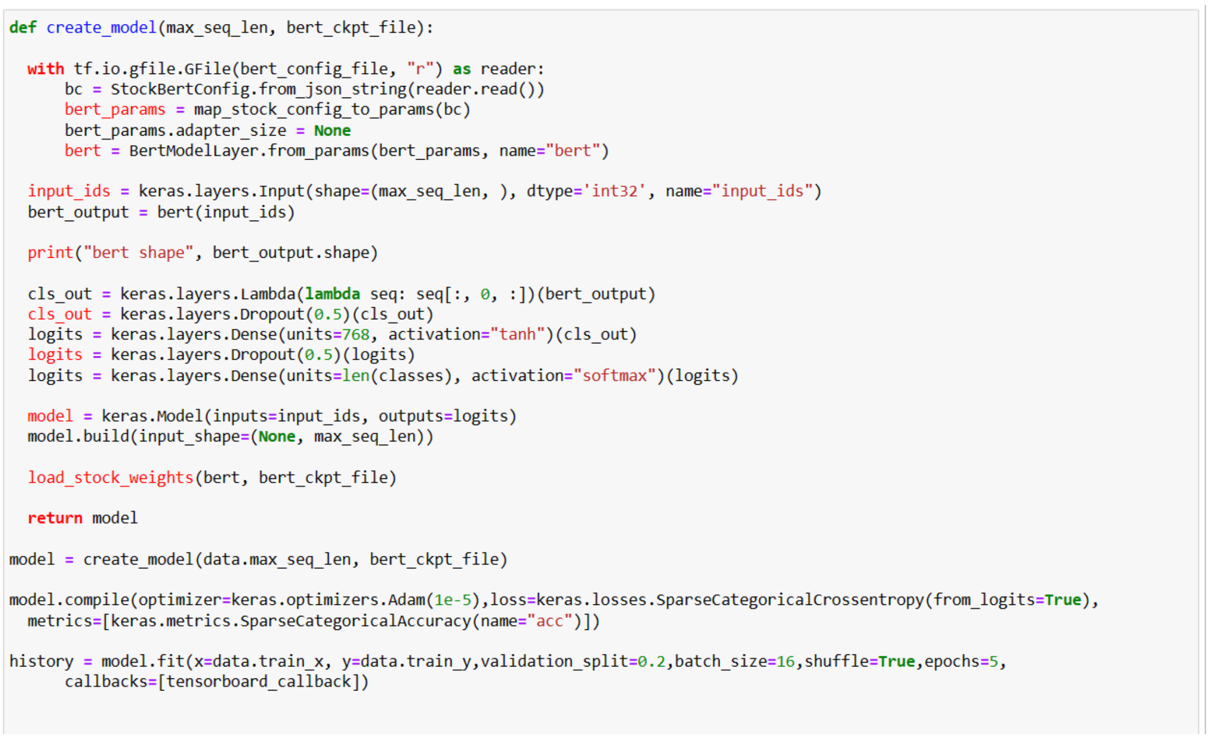
Figure 4.8: Output of pre-processing step Glove-LSTM

These preparation methods should be used to get the dataset ready for the model training procedure.

4.4 Training BERT and obtained results

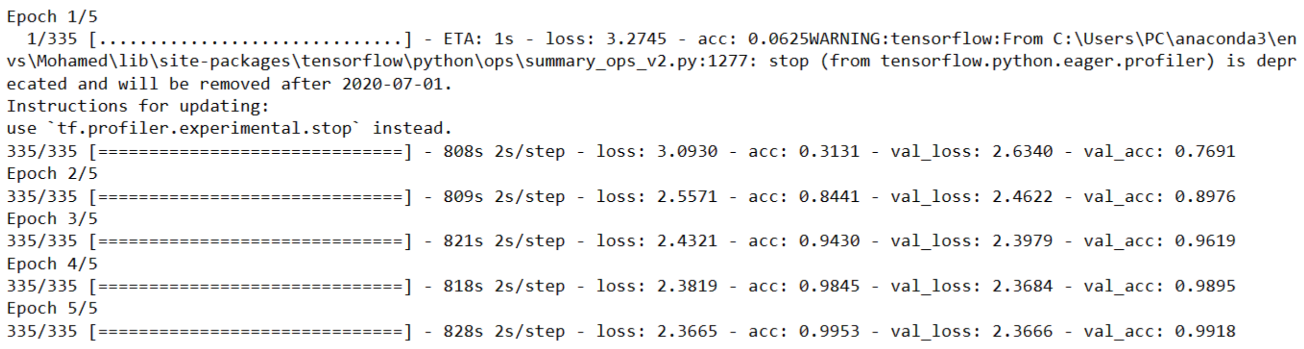
4.4.1 Training model

Our dataset is prepared and preprocessed before we start training our BERT model for intention classification. The model uses the prepared dataset as input and makes an effort to identify patterns between the input data and the associated intent labels in order to predict future intents with high precision. We train the model using a batch size of 16 across 5 epochs. The code snippet 4.9 illustrates the model architecture training process.



```
def create_model(max_seq_len, bert_ckpt_file):

  with tf.io.gfile.GFile(bert_config_file, "r") as reader:
      bc = StockBertConfig.from_json_string(reader.read())
      bert_params = map_stock_config_to_params(bc)
      bert_params.adapter_size = None
      bert = BertModelLayer.from_params(bert_params, name="bert")

  input_ids = keras.layers.Input(shape=(max_seq_len, ), dtype='int32', name="input_ids")
  bert_output = bert(input_ids)

  print("bert shape", bert_output.shape)

  cls_out = keras.layers.Lambda(lambda seq: seq[:, 0, :])(bert_output)
  cls_out = keras.layers.Dropout(0.5)(cls_out)
  logits = keras.layers.Dense(units=768, activation="tanh")(cls_out)
  logits = keras.layers.Dropout(0.5)(logits)
  logits = keras.layers.Dense(units=len(classes), activation="softmax")(logits)

  model = keras.Model(inputs=input_ids, outputs=logits)
  model.build(input_shape=(None, max_seq_len))

  load_stock_weights(bert, bert_ckpt_file)

  return model

model = create_model(data.max_seq_len, bert_ckpt_file)

model.compile(optimizer=keras.optimizers.Adam(1e-5),loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=[keras.metrics.SparseCategoricalAccuracy(name="acc")])

history = model.fit(x=data.train_x, y=data.train_y,validation_split=0.2,batch_size=16,shuffle=True,epochs=5,
      callbacks=[tensorboard_callback])
```

Figure 4.9: BERT Training model architecture



```
Epoch 1/5
  1/335 [..............................] - ETA: 1s - loss: 3.2745 - acc: 0.0625WARNING:tensorflow:From C:\Users\PC\anaconda3\en
vs\Mohamed\lib\site-packages\tensorflow\python\ops\summary_ops_v2.py:1277: stop (from tensorflow.python.eager.profiler) is depr
ecated and will be removed after 2020-07-01.
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
335/335 [==============================] - 808s 2s/step - loss: 3.0930 - acc: 0.3131 - val_loss: 2.6340 - val_acc: 0.7691
Epoch 2/5
335/335 [==============================] - 809s 2s/step - loss: 2.5571 - acc: 0.8441 - val_loss: 2.4622 - val_acc: 0.8976
Epoch 3/5
335/335 [==============================] - 821s 2s/step - loss: 2.4321 - acc: 0.9430 - val_loss: 2.3979 - val_acc: 0.9619
Epoch 4/5
335/335 [==============================] - 818s 2s/step - loss: 2.3819 - acc: 0.9845 - val_loss: 2.3684 - val_acc: 0.9895
Epoch 5/5
335/335 [==============================] - 828s 2s/step - loss: 2.3665 - acc: 0.9953 - val_loss: 2.3666 - val_acc: 0.9918
```

Figure 4.10: BERT training process

As depicted in the figure 4.10, we observe a significant improvement in the model's performance throughout the training process. Starting from an initial accuracy of 0.31, after 5 epochs of training, both the training and validation accuracy reach a high value of 0.99%. Additionally, the loss value exhibits a notable reduction from an initial value of 3.1 to a lower value of 2.3, demonstrating the model's increasing ability to minimize errors and make more accurate predictions.

The duration of each epoch is around 14 minutes, and the entire training time is about 1 hour and 10 minutes.

4.4.2 Model evaluation

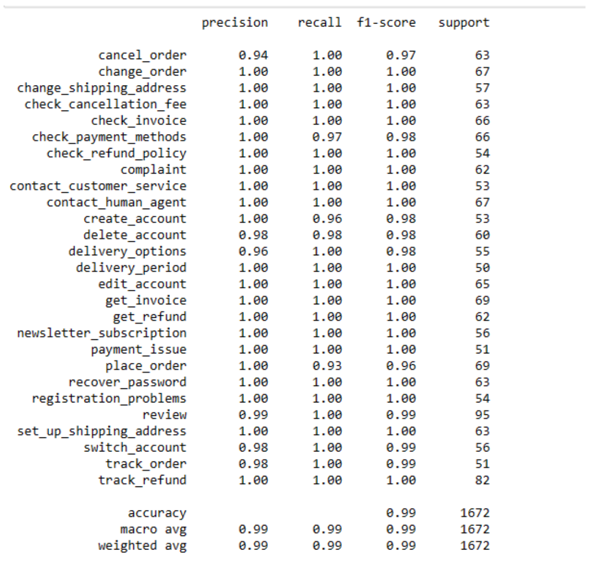
For evaluating and testing our model, we employ well-known metrics such as F1score, precision, recall and the confusion matrix. These metrics let us evaluate the performance of our model and give us useful information about accuracy, precision and ability to correctly classify different intents.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1_score = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \tag{4.3}$$

TP = True positives , FP = False positives, FN = False negative, TN = True negative



```
                          precision    recall  f1-score   support

            cancel_order       0.94      1.00      0.97        63
            change_order       1.00      1.00      1.00        67
 change_shipping_address       1.00      1.00      1.00        57
  check_cancellation_fee       1.00      1.00      1.00        63
           check_invoice       1.00      1.00      1.00        66
   check_payment_methods       1.00      0.97      0.98        66
     check_refund_policy       1.00      1.00      1.00        54
               complaint       1.00      1.00      1.00        62
contact_customer_service       1.00      1.00      1.00        53
     contact_human_agent       1.00      1.00      1.00        67
          create_account       1.00      0.96      0.98        53
          delete_account       0.98      0.98      0.98        60
        delivery_options       0.96      1.00      0.98        55
         delivery_period       1.00      1.00      1.00        50
            edit_account       1.00      1.00      1.00        65
             get_invoice       1.00      1.00      1.00        69
              get_refund       1.00      1.00      1.00        62
 newsletter_subscription       1.00      1.00      1.00        56
           payment_issue       1.00      1.00      1.00        51
             place_order       1.00      0.93      0.96        69
        recover_password       1.00      1.00      1.00        63
   registration_problems       1.00      1.00      1.00        54
                  review       0.99      1.00      0.99        95
    set_up_shipping_address       1.00      1.00      1.00        63
          switch_account       0.98      1.00      0.99        56
             track_order       0.98      1.00      0.99        51
            track_refund       1.00      1.00      1.00        82

                accuracy                           0.99      1672
               macro avg       0.99      0.99      0.99      1672
            weighted avg       0.99      0.99      0.99      1672
```

Figure 4.11: BERT evaluation metrics

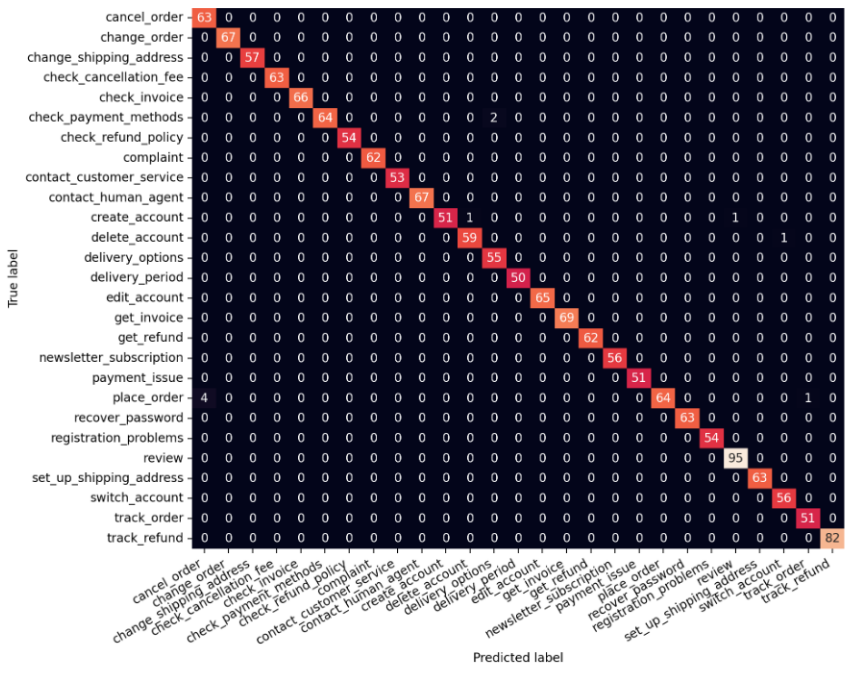


Figure 4.12: BERT confusion matrix

As shown in the confusion matrix and the associated metrics, our model demonstrates excellent performance during the testing phase, achieving an impressive f1score accuracy of around 0.99% for each label or intent prediction. This high accuracy indicates that the model is making accurate predictions and effectively classifying the intents in the test dataset for each specific label.

4.4.3 Result Comparison

For the intent classification phase, we experimented with two additional models, GloVe-LSTM and Bi-LSTM. While both models provided good results, we eventually went with BERT since it performed the best. The decision to select BERT was based on its superior accuracy and its ability to capture contextual information effectively, resulting in more accurate intent classification compared to the other models.

table 4.1 compares the three models in detail.

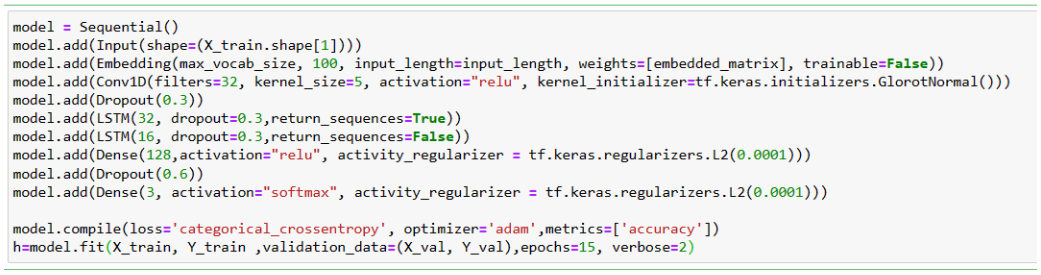
	Accuracy	**Loss**	**Precision**	**Recall**	**f1-score**
Bi-LSTM	0.90	0.29	0.94	0.90	0.90
Glove-LSTM	0.94	0.22	0.97	0.96	0.96
BERT	0.99	2.3	0.99	0.99	0.99

Table 4.1: Comparison of our intention classification models

4.5 Training Glove-LSTM and Obtained results

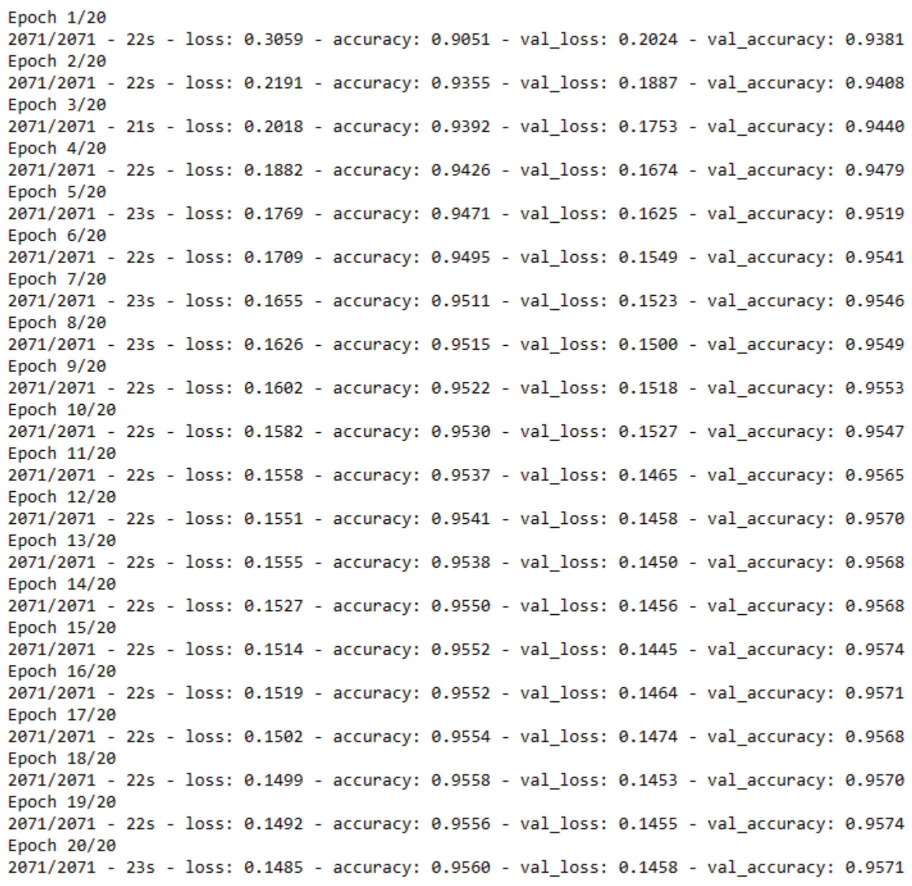
4.5.1 Training model

We've already prepared our dataset, now it's time to train our Glove-LSTM model to classify sentiment. By training our Glove-LSTM model, we aim to achieve accurate sentiment classification results. There will be a total of 20 epochs used to train the network. Code snippet 4.13 illustrates the model architecture.



```
model = Sequential()
model.add(Input(shape=(X_train.shape[1])))
model.add(Embedding(max_vocab_size, 100, input_length=input_length, weights=[embedded_matrix], trainable=False))
model.add(Conv1D(filters=32, kernel_size=5, activation="relu", kernel_initializer=tf.keras.initializers.GlorotNormal()))
model.add(Dropout(0.3))
model.add(LSTM(32, dropout=0.3,return_sequences=True))
model.add(LSTM(16, dropout=0.3,return_sequences=False))
model.add(Dense(128,activation="relu", activity_regularizer = tf.keras.regularizers.L2(0.0001)))
model.add(Dropout(0.6))
model.add(Dense(3, activation="softmax", activity_regularizer = tf.keras.regularizers.L2(0.0001)))

model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
h=model.fit(X_train, Y_train ,validation_data=(X_val, Y_val),epochs=15, verbose=2)
```

Figure 4.13: Glove-LSTM training architecture



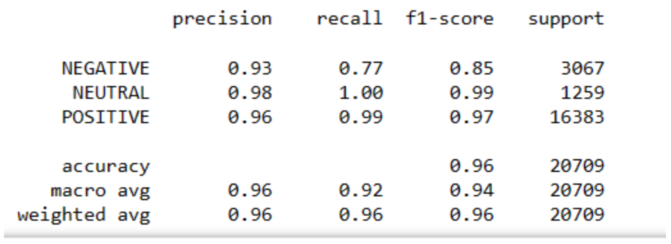
```
Epoch 1/20
2071/2071 - 22s - loss: 0.3059 - accuracy: 0.9051 - val_loss: 0.2024 - val_accuracy: 0.9381
Epoch 2/20
2071/2071 - 22s - loss: 0.2191 - accuracy: 0.9355 - val_loss: 0.1887 - val_accuracy: 0.9408
Epoch 3/20
2071/2071 - 21s - loss: 0.2018 - accuracy: 0.9392 - val_loss: 0.1753 - val_accuracy: 0.9440
Epoch 4/20
2071/2071 - 22s - loss: 0.1882 - accuracy: 0.9426 - val_loss: 0.1674 - val_accuracy: 0.9479
Epoch 5/20
2071/2071 - 23s - loss: 0.1769 - accuracy: 0.9471 - val_loss: 0.1625 - val_accuracy: 0.9519
Epoch 6/20
2071/2071 - 22s - loss: 0.1709 - accuracy: 0.9495 - val_loss: 0.1549 - val_accuracy: 0.9541
Epoch 7/20
2071/2071 - 23s - loss: 0.1655 - accuracy: 0.9511 - val_loss: 0.1523 - val_accuracy: 0.9546
Epoch 8/20
2071/2071 - 23s - loss: 0.1626 - accuracy: 0.9515 - val_loss: 0.1500 - val_accuracy: 0.9549
Epoch 9/20
2071/2071 - 22s - loss: 0.1602 - accuracy: 0.9522 - val_loss: 0.1518 - val_accuracy: 0.9553
Epoch 10/20
2071/2071 - 22s - loss: 0.1582 - accuracy: 0.9530 - val_loss: 0.1527 - val_accuracy: 0.9547
Epoch 11/20
2071/2071 - 22s - loss: 0.1558 - accuracy: 0.9537 - val_loss: 0.1465 - val_accuracy: 0.9565
Epoch 12/20
2071/2071 - 22s - loss: 0.1551 - accuracy: 0.9541 - val_loss: 0.1458 - val_accuracy: 0.9570
Epoch 13/20
2071/2071 - 22s - loss: 0.1555 - accuracy: 0.9538 - val_loss: 0.1450 - val_accuracy: 0.9568
Epoch 14/20
2071/2071 - 22s - loss: 0.1527 - accuracy: 0.9550 - val_loss: 0.1456 - val_accuracy: 0.9568
Epoch 15/20
2071/2071 - 22s - loss: 0.1514 - accuracy: 0.9552 - val_loss: 0.1445 - val_accuracy: 0.9574
Epoch 16/20
2071/2071 - 22s - loss: 0.1519 - accuracy: 0.9552 - val_loss: 0.1464 - val_accuracy: 0.9571
Epoch 17/20
2071/2071 - 22s - loss: 0.1502 - accuracy: 0.9554 - val_loss: 0.1474 - val_accuracy: 0.9568
Epoch 18/20
2071/2071 - 22s - loss: 0.1499 - accuracy: 0.9558 - val_loss: 0.1453 - val_accuracy: 0.9570
Epoch 19/20
2071/2071 - 22s - loss: 0.1492 - accuracy: 0.9556 - val_loss: 0.1455 - val_accuracy: 0.9574
Epoch 20/20
2071/2071 - 23s - loss: 0.1485 - accuracy: 0.9560 - val_loss: 0.1458 - val_accuracy: 0.9571
```

Figure 4.14: Glove-LSTM training process

As can be observed in the figure 4.14 , the performance of the model has shown notable enhancement during the training process. After 20 epochs, the accuracy increases progressively from a starting value of 0.90 to 0.95. The validation accuracy also shows a similar positive trend. Additionally, the loss value decreases from 0.30 to 0.14, demonstrating that the model was successful in reducing error throughout training.

4.5.2 Model evaluation

we employ the previously discussed metric in the evaluation phase to measure the success of the model.



```
              precision    recall  f1-score   support

    NEGATIVE       0.93      0.77      0.85      3067
     NEUTRAL       0.98      1.00      0.99      1259
    POSITIVE       0.96      0.99      0.97     16383

    accuracy                           0.96     20709
   macro avg       0.96      0.92      0.94     20709
weighted avg       0.96      0.96      0.96     20709
```

Figure 4.15: Glove-LSTM evaluation metrics

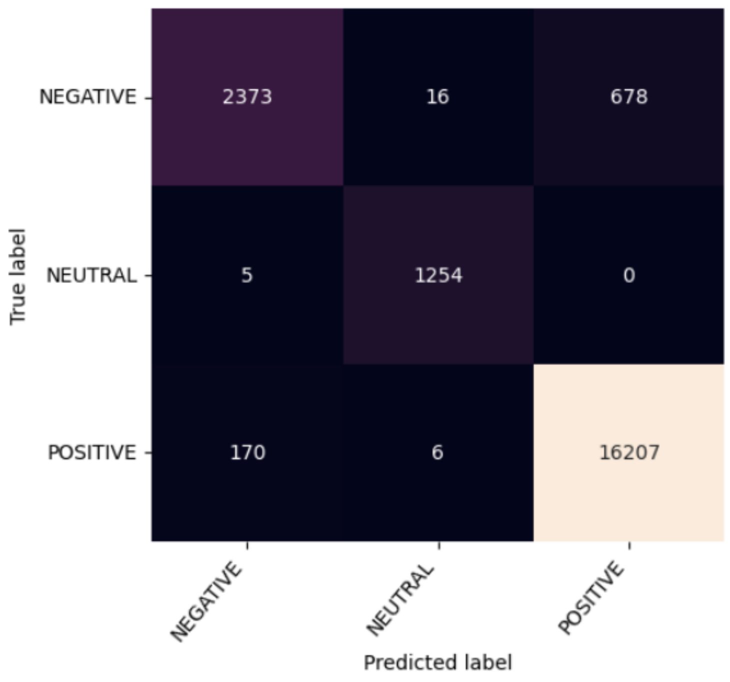


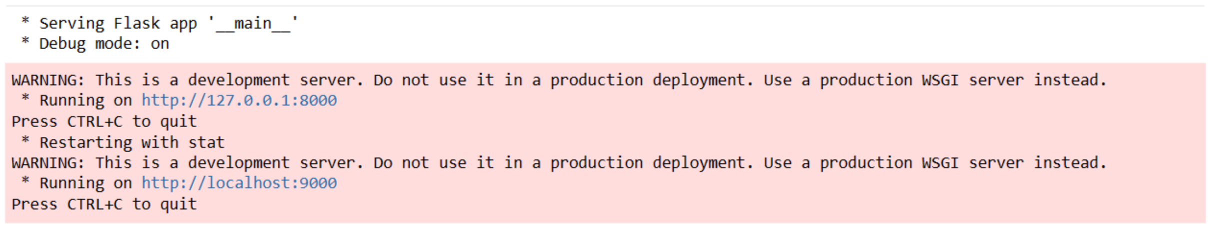
Figure 4.16: Glove-LSTM confusion matrix

Based on the confusion matrix and F1 score analysis, our model demonstrates good performance in sentiment classification during the testing phase. We achieve an accuracy of F1score of 85% in correctly predicting negative sentiments, 97% F1score accuracy in correctly predicting positive sentiments, and 99% accuracy in correctly predicting neutral sentiments. The unbalanced nature of our dataset has a noticeable impact on the accuracy of the sentiment categories in our model.

4.6 Working Process of our chatbot

In this section, we will illustrate the workings of our chatbotand how it functions.

4.6.1 Establishing a connection to the back-end server



```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:8000
Press CTRL+C to quit
 * Restarting with stat
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://localhost:9000
Press CTRL+C to quit
```

Figure 4.17: deployment of the FLASK server

Establishing the FLASK server, which is operating on my own machine, for the back-end chatbot operations which is connected to the same network as the web app.

4.6.2 User request

When a user accesses to the web app, the app instantly connects to the FLASK API. The user then writes his inquiry into a text box and submits it to the customer assistance by clicking the send button.

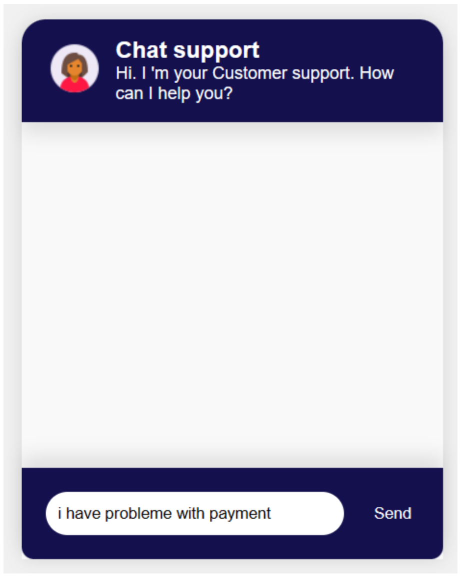
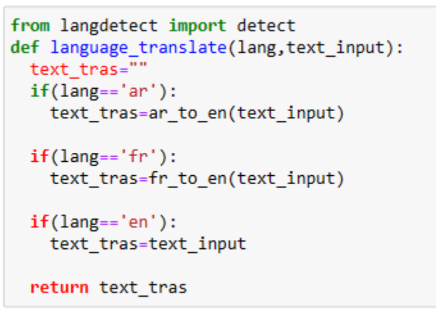

Chat support
Hi. I 'm your Customer support. How can I help you?
i have probleme with payment
Send

Figure 4.18: User input

4.6.3 Text translation

The system of our chatbot generally operates in English, if the user's input is not in English, a translation to English must proceed,the system can translate from three languages (English, french and Arabic).

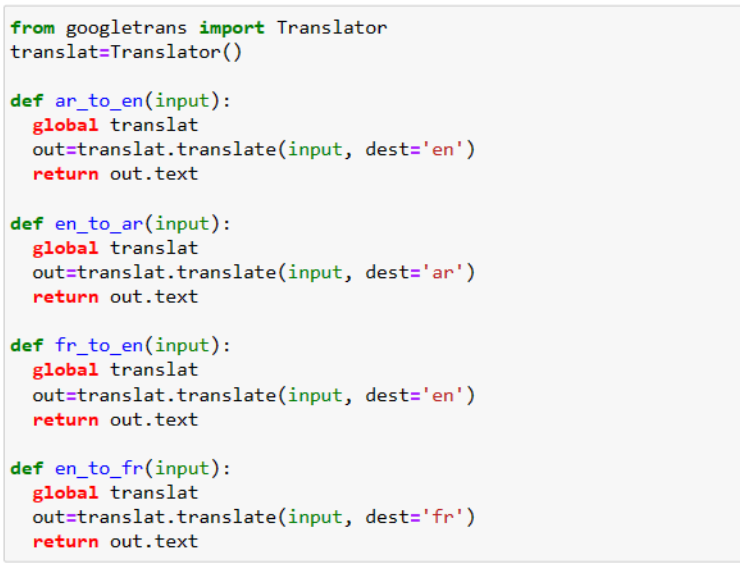


```
from langdetect import detect
def language_translate(lang,text_input):
  text_tras=""
  if(lang=='ar'):
    text_tras=ar_to_en(text_input)

  if(lang=='fr'):
    text_tras=fr_to_en(text_input)

  if(lang=='en'):
    text_tras=text_input

  return text_tras
```

Figure 4.19: User language detection

As demonstrated in the code snippet above, we use a language detection package named "langdetect". After identifying the language, we use the Google Translation API to convert the customer's input into English.

The implementation of the Google Translate API for translating functions is seen in the code snippet below.

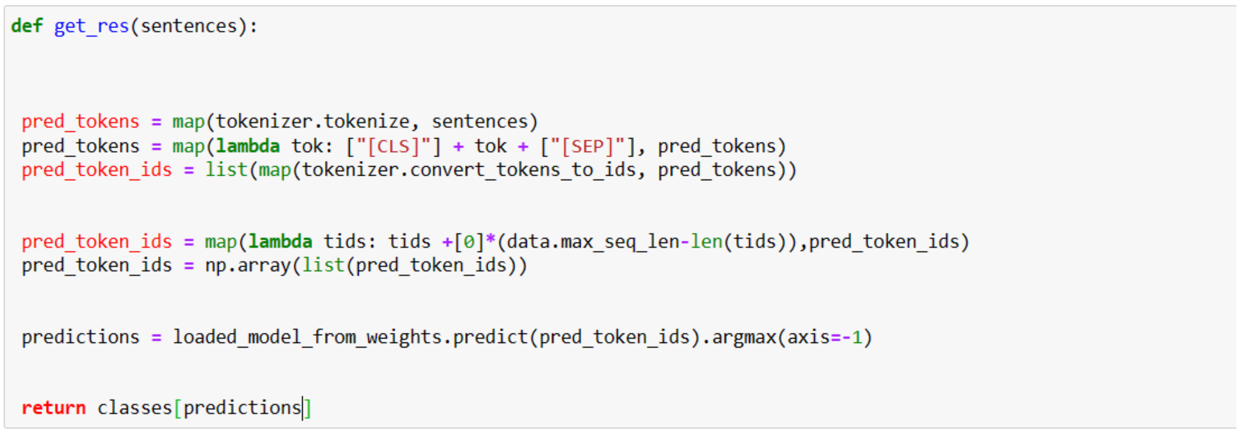


```
from googletrans import Translator
translat=Translator()

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

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

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

def en_to_fr(input):
  global translat
  out=translat.translate(input, dest='fr')
  return out.text
```

Figure 4.20: Google traslaton API

4.6.4 Preprossing the user input

We apply the pre-processing techniques previously covered.

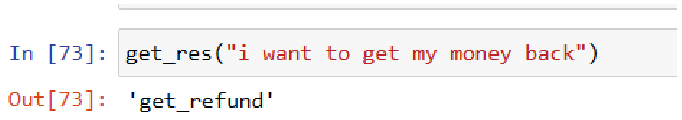


```
def get_res(sentences):


 pred_tokens = map(tokenizer.tokenize, sentences)
 pred_tokens = map(lambda tok: ["[CLS]"] + tok + ["[SEP]"], pred_tokens)
 pred_token_ids = list(map(tokenizer.convert_tokens_to_ids, pred_tokens))


 pred_token_ids = map(lambda tids: tids +[0]*(data.max_seq_len-len(tids)),pred_token_ids)
 pred_token_ids = np.array(list(pred_token_ids))


 predictions = loaded_model_from_weights.predict(pred_token_ids).argmax(axis=-1)


 return classes[predictions]
```

Figure 4.21: Use of the model for predicting

4.6.5 Model prediction

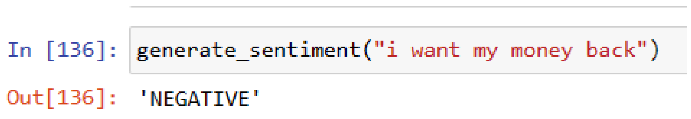
We can guess the intent from the input string ”i want to get my money back” as illustrated in figure 4.22.



```
In [73]: get_res("i want to get my money back")
Out[73]: 'get_refund'
```

Figure 4.22: Intent prediction

Then we guess the sentiment from the input string ”i want to get my money back” as illustrated in figure 4.23.



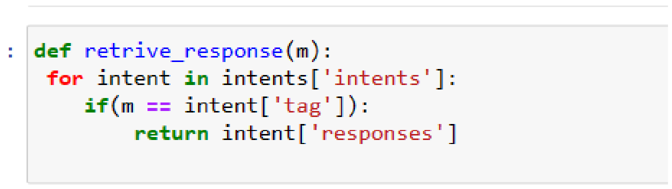
```
In [136]: generate_sentiment("i want my money back")
Out[136]: 'NEGATIVE'
```

Figure 4.23: Sentiment prediction

4.6.6 Answer retrieval

Based on the predicted intent and sentiment, we utilize a function to retrieve the appropriate answer from our data source, which contains the corresponding responses.

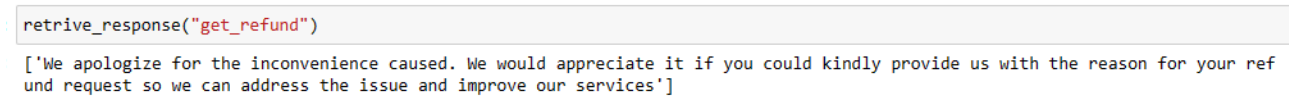
The method responsible for response retrieval is displayed in the given code.



```
def retrive_response(m):
 for intent in intents['intents']:
    if(m == intent['tag']):
        return intent['responses']
```

Figure 4.24: retrieve Response

The chatbot's answer for the "get refund" is displayed in the figure.

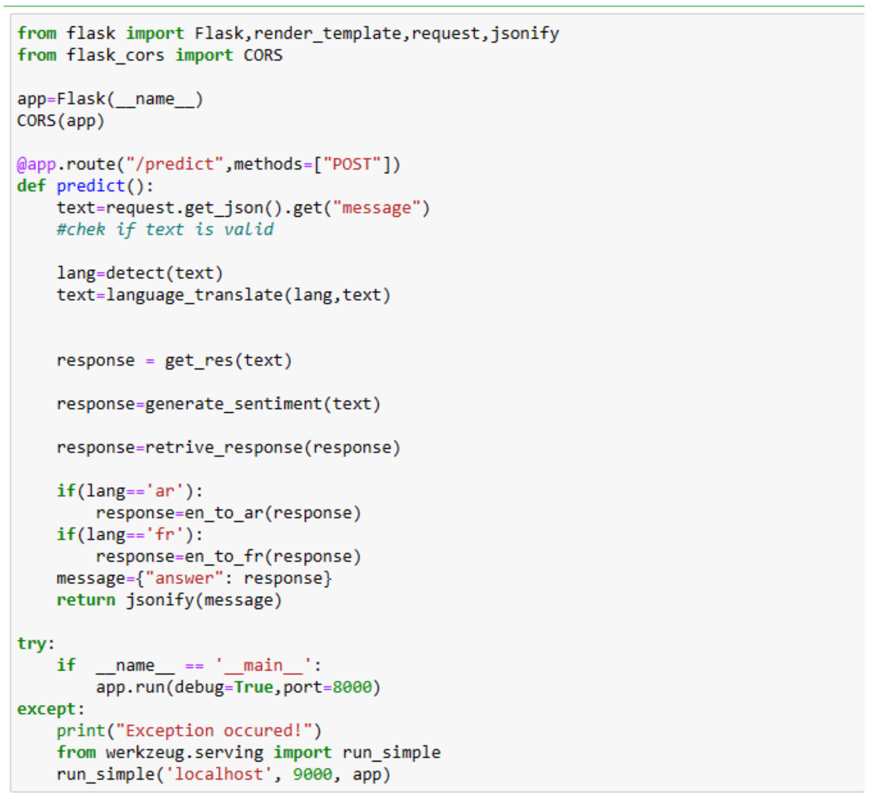


```
retrive_response("get_refund")
['We apologize for the inconvenience caused. We would appreciate it if you could kindly provide us with the reason for your ref
und request so we can address the issue and improve our services']
```

Figure 4.25: Response generation

4.6.7 Output of the chatbot

The acquired answer will be translated into the target language before being sent to the web application using the POST method on the FLASK server, as seen in the code below.



```python
from flask import Flask,render_template,request,jsonify
from flask_cors import CORS

app=Flask(__name__)
CORS(app)

@app.route("/predict",methods=["POST"])
def predict():
    text=request.get_json().get("message")
    #chek if text is valid

    lang=detect(text)
    text=language_translate(lang,text)


    response = get_res(text)

    response=generate_sentiment(text)

    response=retrive_response(response)

    if(lang=='ar'):
        response=en_to_ar(response)
    if(lang=='fr'):
        response=en_to_fr(response)
    message={"answer": response}
    return jsonify(message)

try:
    if  __name__ == '__main__':
        app.run(debug=True,port=8000)
except:
    print("Exception occured!")
    from werkzeug.serving import run_simple
    run_simple('localhost', 9000, app)
```

Figure 4.26: Flask output of the chatbot

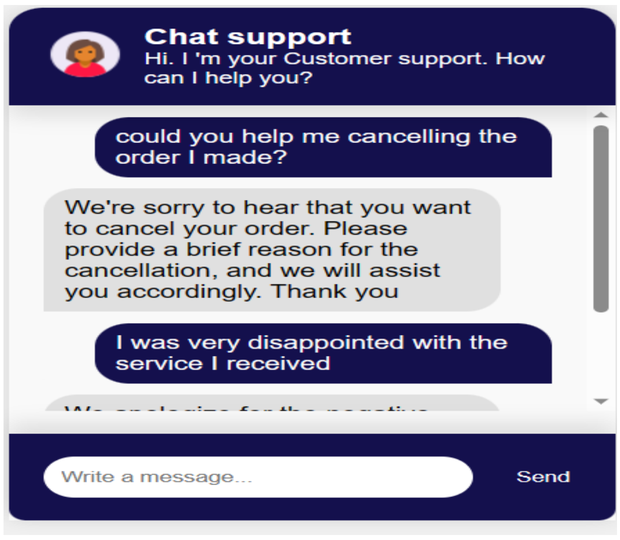
4.7 Chatbot interfaces and examples

This part includes examples of our chabot applications and user interfaces in the three available languages: Arabic, English, and French.

- **English user example**

  As illustrated in the figure 4.27 the chatbot talk to an english user.

  when the chatbot faced with negative sentiment, the chatbot should answer in a way that respects the customer's feels while also offering a solution or an opportunity to fix the issue.



Chat support
Hi. I 'm your Customer support. How can I help you?

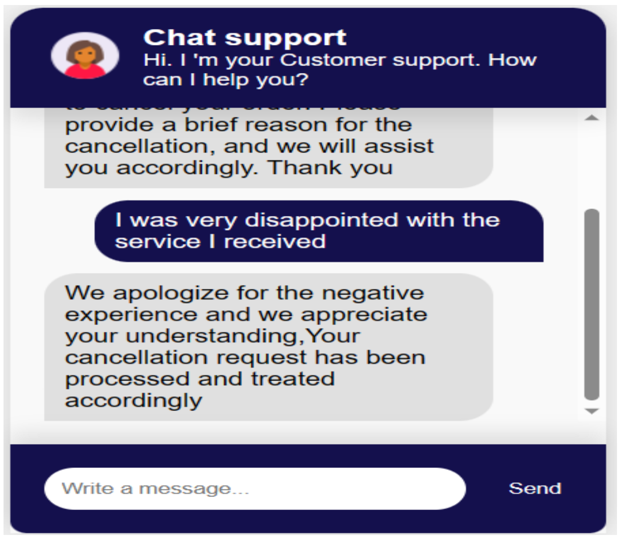
could you help me cancelling the order I made?

We're sorry to hear that you want to cancel your order. Please provide a brief reason for the cancellation, and we will assist you accordingly. Thank you

I was very disappointed with the service I received

Write a message...
Send

1



Chat support
Hi. I 'm your Customer support. How can I help you?

provide a brief reason for the cancellation, and we will assist you accordingly. Thank you

I was very disappointed with the service I received

We apologize for the negative experience and we appreciate your understanding,Your cancellation request has been processed and treated accordingly

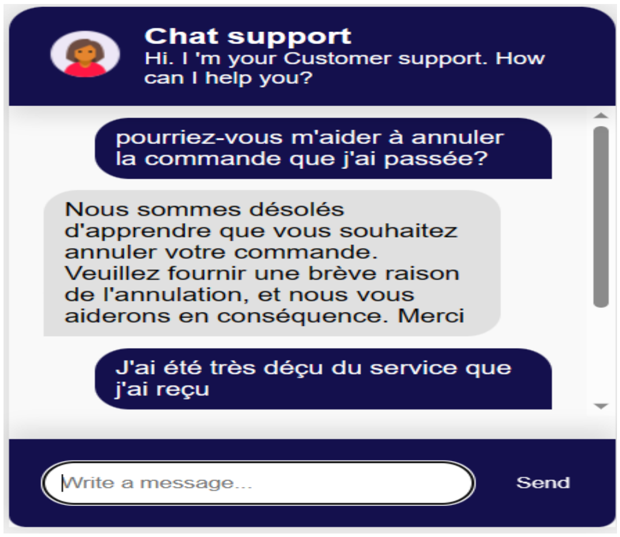
Write a message...
Send

2

Figure 4.27: Interface with English user

- **French user example**

  As shown the figure 4.28 the chatbot talk to an french user



Chat support
Hi. I 'm your Customer support. How can I help you?

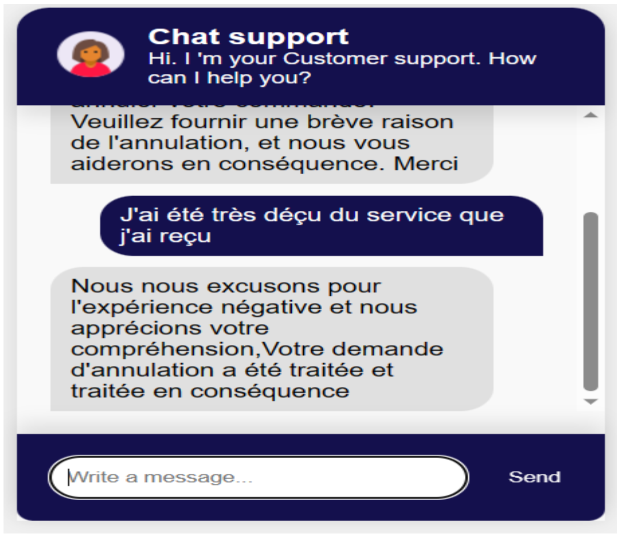
pourriez-vous m'aider à annuler la commande que j'ai passée?

Nous sommes désolés d'apprendre que vous souhaitez annuler votre commande. Veuillez fournir une brève raison de l'annulation, et nous vous aiderons en conséquence. Merci

J'ai été très déçu du service que j'ai reçu

Write a message...
Send

1



Chat support
Hi. I 'm your Customer support. How can I help you?

Veuillez fournir une brève raison de l'annulation, et nous vous aiderons en conséquence. Merci

J'ai été très déçu du service que j'ai reçu

Nous nous excusons pour l'expérience négative et nous apprécions votre compréhension,Votre demande d'annulation a été traitée et traitée en conséquence

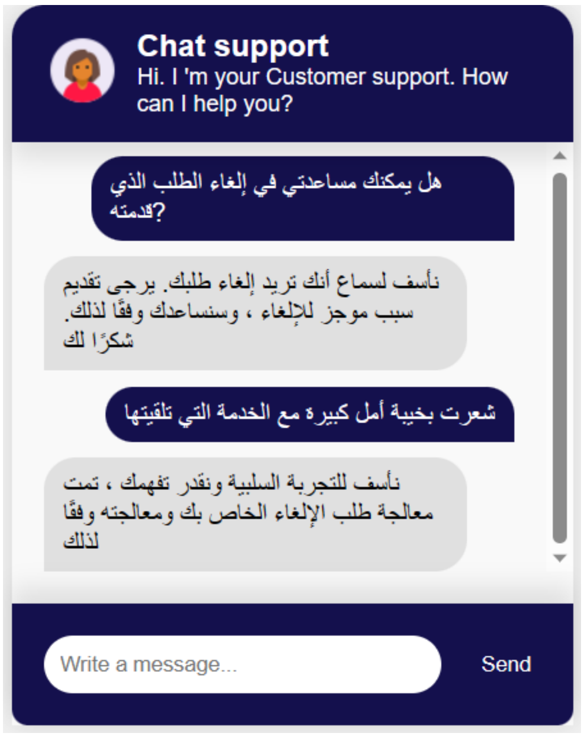
Write a message...
Send

2

Figure 4.28: Interface with French user

- **Arabic user example**

  As shown the figure 4.29 the chatbot talk to an arabic user



Chat support
Hi. I 'm your Customer support. How can I help you?

هل يمكنك مساعدتي في إلغاء الطلب الذي قدمته?

نأسف لسماع أنك تريد إلغاء طلبك. يرجى تقديم سبب موجز للإلغاء ، وسنساعدك وفقًا لذلك. شكرًا لك

شعرت بخيبة أمل كبيرة مع الخدمة التي تلقيتها

نأسف للتجربة السلبية ونقدر تفهمك ، تمت معالجة طلب الإلغاء الخاص بك ومعالجته وفقًا لذلك

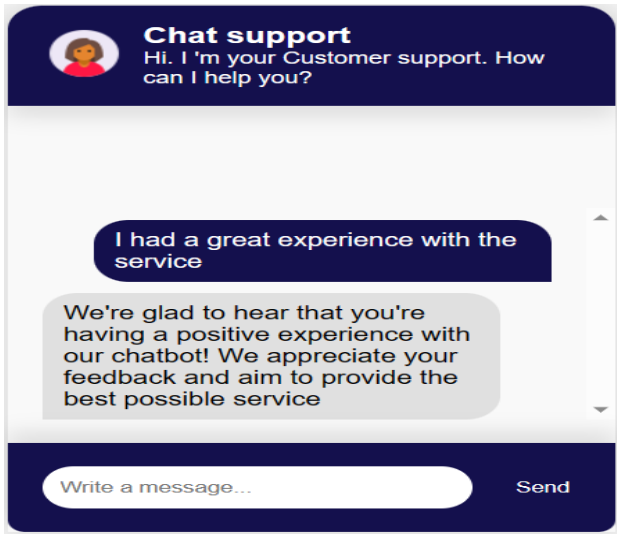
Figure 4.29: Interface test with Arabic user

- **Sentiment review example**

  As illustrated in the figure 4.30, the chatbot effectively handles with user reviews by providing appropriate responses based on the sentiment expressed.

  When the chatbot encountering positive reviews, the chatbot answer with thankfulness, reinforcing the nice experience.

  On the other hand, when faced with negative reviews, the chatbot responds with understanding and compassion.

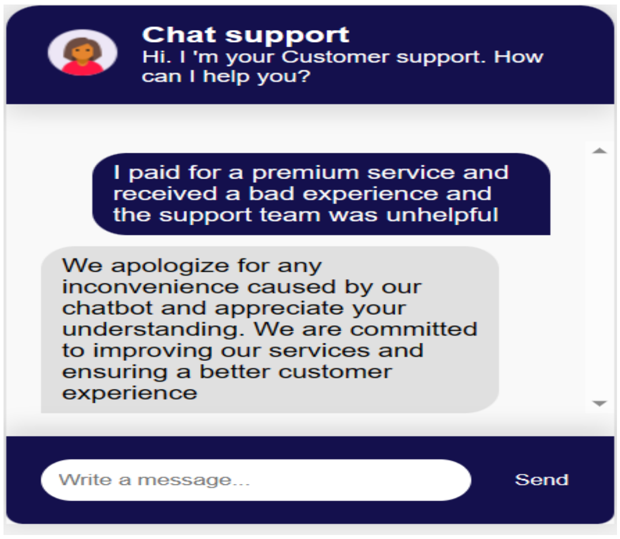


Chat support
Hi. I 'm your Customer support. How can I help you?

I had a great experience with the service

We're glad to hear that you're having a positive experience with our chatbot! We appreciate your feedback and aim to provide the best possible service

Positive review



Chat support
Hi. I 'm your Customer support. How can I help you?

I paid for a premium service and received a bad experience and the support team was unhelpful

We apologize for any inconvenience caused by our chatbot and appreciate your understanding. We are committed to improving our services and ensuring a better customer experience

Negative review

Figure 4.30: Interface test with Sentiment

4.8 Conclusion

In this chapter, we have showcased the implementation of our system and discussed the process of training our models and the analysis of the obtained results. After doing a comprehensive comparison of three different models to determine the most effective one. We carefully designed the architectures of our models to suit our dataset and maximize their performance. The results were highly promising, and our models showed a high level of accuracy.

General Conclusion

Conclusion

In the business world, customer support serves as a vital communication channel between the business and its customers.

In this context, we employed transfer learning and natural language processing (NLP) techniques to build a powerful architecture for intention classifications and deep learning model for sentiment classification.

By incorporating transfer learning techniques, specifically utilizing the pre-trained model BERT, our architecture demonstrated a superior performance in recognizing and understanding customer intentions. In comparison to earlier chatbot systems, this method performed better than both deep learning and machine learning models.

The integration of deep learning LSTM with pre-trained GloVe embeddings, our sentiment analysis architecture achieved remarkable results in understanding and classifying customer sentiments. The GloVe embeddings provided a valuable representation of words and their contextual meaning, enabling our model to capture the nuanced sentiment expressed in customer interactions.

Our system includes a dialogue state monitoring component that serves as the dialog manager in charge of controlling the conversation's flow between the user and the chatbot.

Future work

In order to enhance the customer experience, our work aims to continually improve our services.

We plan to implement a recommendation system that takes into account customer sentiment to offer personalized and relevant suggestions. Based on the data collected by our chatbot regarding customer sentiment, we can use this information to enhance the customer experience furthermore. We also need to Take into consideration negative feedback by offering alternative options to address any dissatisfaction.

By customizing suggestions depending on the sentiment and preferences of the our client, this recommendation system helps improving the whole customer experience, increasing customer satisfaction while driving company growth.

In addition to text-based interactions, we plan to enhance our chatbot system by implementing voice capabilities. This integration of voice technology will enable customers to effortlessly communicate their queries or concerns by speaking, creating a more natural and convenient interaction experience.

Bibliography

1. Rautaray, S. S. & Agrawal, A. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial intelligence review* **43,** 1–54 (2015).
2. Ke, Q. *et al.* in *Computer Vision for Assistive Healthcare* 127–145 (Elsevier, 2018).
3. Nalepka, P. *et al.* Human social motor solutions for human–machine interaction in dynamical task contexts. *Proceedings of the National Academy of Sciences* **116,** 1437–1446 (2019).
4. Assaf, T. A frequency modulation-based taxel array: A bio-inspired architecture for large-scale artificial skin. *Sensors* **21,** 5112 (2021).
5. Reiterer, H. & Dachselt, R. Designing graphical user interfaces: challenges and principles. *Journal of Multimodal User Interfaces* **10,** 75–91 (2016).
6. Stolterman, E., Wiberg, M. & Jakobsen, M. The design of voice user interfaces. *Journal of Design and Science* **4,** e26 (2018).
7. Karray, F., Alemzadeh, M., Abou Saleh, J. & Arab, M. N. Human-computer interaction: Overview on state of the art. *International journal on smart sensing and intelligent systems* **1,** 137–159 (2008).
8. SCETA. *Human-Machine Interfaces* `https://sceta.io/human-machine-interfaces/`. 2021.
9. Bashashati, A., Fatourechi, M., Ward, R. K. & Birch, G. E. A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural engineering* **4,** R32 (2015).

10. Zhu, M. *et al.* Haptic-feedback smart glove as a creative human-machine interface (HMI) for virtual/augmented reality applications. *Science Advances* **6,** eaaz8693 (2020).

11. Gartner. *Case-Based Reasoning (CBR)* `https://www.gartner.com/en/information-technology/glossary/cbr-case-based-reasoning`. Accessed: May 2, 2023. 2022.

12. Goel, A. & Diaz-Agudo, B. *What's hot in case-based reasoning* in *Proceedings of the AAAI Conference on Artificial Intelligence* **31** (2017).

13. Richter, M. M. & Weber, R. O. *Case-based reasoning* (Springer, 2016).

14. Pusztová, L., Babič, F. & Paralič, J. Semi-Automatic Adaptation of Diagnostic Rules in the Case-Based Reasoning Process. *Applied Sciences* **11,** 292 (2020).

15. Reuss, P., Dick, M., Termath, W. & Althoff, K.-D. Case-based reasoning: potential benefits and limitations for documenting of stories in organizations. *Zeitschrift für Arbeitswissenschaft* **71** (Dec. 2017).

16. Sheng, Y. *et al.* Incorporating case-based reasoning for radiation therapy knowledge modeling: a pelvic case study. *Technology in cancer research & treatment* **18,** 1533033819874788 (2019).

17. Riesbeck, C. K. & Schank, R. C. *Inside case-based reasoning* (Psychology Press, 2013).

18. Liu, W., Tan, R., Cao, G., Yu, F. & Li, H. Creative design through knowledge clustering and case-based reasoning. *Engineering with Computers* **36,** 527–541 (2020).

19. Bansal, H. & Khan, R. A review paper on human computer interaction. *Int. J. Adv. Res. Comput. Sci. Softw. Eng* **8,** 53 (2018).

20. Khan, R. & Das, A. Build better chatbots. *A complete guide to getting started with chatbots* (2018).

21. Khanna, A. *et al.* A study of today's AI through chatbots and rediscovery of machine intelligence. *International Journal of u-and e-Service, Science and Technology* **8,** 277–284 (2015).

22. Paz, F. J., Silveira, C., Krassmann, A. & Tarouco, L. M. Perspectivas tecnológicas para o aprimoramento de chatbots educacionais em AIML. *Revista Iberoamericana de Tecnologıa en Educación y Educación en Tecnologıa,* 7–15 (2017).

23. Ondáš, S., Pleva, M. & Hládek, D. *How chatbots can be involved in the education process* in *2019 17th international conference on emerging elearning technologies and applications (ICETA)* (2019), 575–580.

24. Brandtzaeg, P. B. & Følstad, A. *Why people use chatbots* in *Internet Science: 4th International Conference, INSCI 2017, Thessaloniki, Greece, November 22-24, 2017, Proceedings 4* (2017), 377–392.

25. Adamopoulou, E. & Moussiades, L. *An overview of chatbot technology* in *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16* (2020), 373–383.

26. Nimavat, K. & Champaneria, T. Chatbots: An overview types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev* **5,** 1019–1024 (2017).

27. Tamrakar, R. & Wani, N. *Design and development of CHATBOT: A review* in *Proceedings of International Conference On "Latest Trends in Civil, Mechanical and Electrical Engineering". https://www. researchgate. net/publication/351228837* (2021).

28. Kucherbaev, P., Bozzon, A. & Houben, G.-J. Human-aided bots. *IEEE Internet Computing* **22,** 36–43 (2018).

29. Ramesh, K., Ravishankaran, S., Joshi, A. & Chandrasekaran, K. *A survey of design techniques for conversational agents* in *Information, Communication and Computing Technology: Second International Conference, ICICCT 2017, New Delhi, India, May 13, 2017, Revised Selected Papers* (2017), 336–350.

30. Hien, H. T., Cuong, P.-N., Nam, L. N. H., Nhung, H. L. T. K. & Thang, L. D. *Intelligent assistants in higher-education environments: the FIT-EBot, a chatbot for administrative and learning support* in *Proceedings of the 9th International Symposium on Information and Communication Technology* (2018), 69–76.

31. Wu, Y., Wu, W., Xing, C., Zhou, M. & Li, Z. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627* (2016).

32. Nguyen, T. *et al.* Potential effects of chatbot technology on customer support: A case study (2019).

33. Peters, F. *et al.* Master thesis: Design and implementation of a chatbot in the context of customer support (2018).

34. Sinha, S., Basak, S., Dey, Y. & Mondal, A. *An educational Chatbot for answering queries* in *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018* (2020), 55–60.

35. Dharani, M., Jyostna, J., Sucharitha, E., Likitha, R. & Manne, S. *Interactive transport enquiry with AI chatbot* in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)* (2020), 1271–1276.

36. Ouerhani, N., Maalel, A. & Ben Ghézala, H. SMAD: SMart assistant during and after a medical emergency case based on deep learning sentiment analysis: The pandemic COVID-19 case. *Cluster computing* **25,** 3671–3681 (2022).

37. Jung, S. Semantic vector learning for natural language understanding. *Computer Speech & Language* **56,** 130–145 (2019).

38. Brownlee, J. *Deep learning for natural language processing: develop deep learning models for your natural language problems* (Machine Learning Mastery, 2017).

39. Dahouda, M. K. & Joe, I. A deep-learned embedding technique for categorical features encoding. *IEEE Access* **9,** 114381–114391 (2021).

40. Eisenstein, J. *Introduction to natural language processing* (MIT press, 2019).

41. SyndellTech. Artificial Intelligence and Natural Language Processing. *SyndellTech Blog.* `https://syndelltech.com/artificial-intelligence-and-natural-language-processing/` (2021).

42. Jusoh, S. A STUDY ON NLP APPLICATIONS AND AMBIGUITY PROBLEMS. *Journal of Theoretical & Applied Information Technology* **96** (2018).

43. Van der Aa, H., Carmona Vargas, J., Leopold, H., Mendling, J. & Padró, L. *Challenges and opportunities of applying natural language processing in business process management* in *COLING 2018: The 27th International Conference on Computational Linguistics: Proceedings of the Conference: August 20-26, 2018 Santa Fe, New Mexico, USA* (2018), 2791–2801.

44. Hirst, G. & Søgaard, A. *Natural Language Processing: Synthesis Lectures on Human Language Technologies* (Morgan Claypool Publishers, 2012).

45. Jurafsky, D. & Martin, J. H. *Speech and Language Processing* (Pearson, 2019).

46. Patwardhan, N., Marrone, S. & Sansone, C. Transformers in the Real World: A Survey on NLP Applications. *Information* **14,** 242 (2023).

47. El Naqa, I. & Murphy, M. J. *What is machine learning?* (Springer, 2015).

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

49. Goldberg, Y. A primer on neural network models for natural language processing. CoRR abs/1510.00726 (2015). *arXiv preprint arXiv:1510.00726* (2015).

50. Khurana, D., Koli, A., Khatter, K. & Singh, S. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications* **82,** 3713–3744 (2023).

51. Hämäläinen, M. *Harnessing nlg to create finnish poetry automatically* in *Proceedings of the ninth international conference on computational creativity* (2018).

52. Léon, J. *Early Machine Translation: Integration and Transfers between Computing and the Language Sciences* in *Language, Life, Limits: 10th Conference on Computability in Europe, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings 10* (2014), 275–282.

53. Liddy, E. D. Natural language processing (2001).

54. Weizenbaum, J. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM* **26,** 23–28 (1983).

55. Winograd, T. *Procedures as a representation for data in a computer program for understanding natural language* tech. rep. (MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971).

56. Cerf, V. *RFC0439: PARRY encounters the DOCTOR* 1973.

57. Joseph, S. R., Hlomani, H., Letsholo, K., Kaniwa, F. & Sedimo, K. Natural language processing: A review. *International Journal of Research in Engineering and Applied Sciences* **6,** 207–210 (2016).

58. Bahja, M. Natural language processing applications in business. *E-Business-Higher Education and Intelligence Applications* (2020).

59. Runeson, P., Alexandersson, M. & Nyholm, O. *Detection of duplicate defect reports using natural language processing* in *29th International Conference on Software Engineering (ICSE'07)* (2007), 499–510.

60. SMLTAR. *Tokenization* SMLTAR. `https://smltar.com/tokenization.html`.

61. Schütze, H., Manning, C. D. & Raghavan, P. *Introduction to information retrieval* (Cambridge University Press Cambridge, 2008).

62. ithome. *ChatbotChatbot* ithome. `https://ithelp.ithome.com.tw/articles/10296189`.

63. Bird, S., Klein, E. & Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit* (" O'Reilly Media, Inc.", 2009).

64. Jurafsky, D. & Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* 3rd ed. (Pearson, 2020).

65. Mohit, B. Named entity recognition. *Natural language processing of semitic languages,* 221–245 (2014).

66. Shaip. *Named Entity Recognition and Its Types* Shaip. `https://www.shaip.com/named-entity-recognition-and-its-types/`.

67. AlmaBetter. *Data Preprocessing with scikit-learn: A Tutorial* AlmaBetter. `https://www.almabetter.com/blogs/data-preprocessing-with-scikit-learn-a-tutorial`.

68. Jawale, C. *Using Label Encoder on Unbalanced Categorical Data in Machine Learning using Python* Medium. `https://medium.com/@chexki_/using-label-encoder-on-unbalanced-categorical-data-in-machine-learning-using-python-435f521323b1`.

69. ALI, A. *Getting Started - Titanic: Machine Learning from Disaster* Kaggle. `https://www.kaggle.com/getting-started/187540`.

70. Liu, L., Tang, L., Dong, W., Yao, S. & Zhou, W. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus* **5,** 1–22 (2016).

71. Asghar, M. Z., Khan, A., Ahmad, S. & Kundi, F. M. A review of feature extraction in sentiment analysis. *Journal of Basic and Applied Scientific Research* **4,** 181–186 (2014).

72. Greco, C. M., Tagarelli, A. & Zumpano, E. *A Comparison of Transformer-Based Language Models on NLP Benchmarks* in *Natural Language Processing and Information Systems: 27th International Conference on Applications of Natural Language to Information Systems, NLDB 2022, Valencia, Spain, June 15–17, 2022, Proceedings* (2022), 490–501.

73. Chernyavskiy, A., Ilvovsky, D. & Nakov, P. *Transformers: "the end of history" for natural language processing?* in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21* (2021), 677–693.

74. Devika, M., Sunitha, C. & Ganesh, A. Sentiment analysis: a comparative study on different approaches. *Procedia Computer Science* **87,** 44–49 (2016).

75. Kula, S., Kozik, R. & Choraś, M. Implementation of the BERT-derived architectures to tackle disinformation challenges. *Neural Computing and Applications,* 1–13 (2021).

76. Vaswani, A. *et al.* Attention is all you need. *Advances in neural information processing systems* **30** (2017).

77. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

78. Li, Q. *et al.* A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364* (2020).

79. Alammar, J. The Illustrated BERT, ELMo, and co. *Jay Alammar's Blog.* `http://jalammar.github.io/illustrated-bert/` (2018).

80. Sun, J., Liu, Y., Cui, J. & He, H. Deep learning-based methods for natural hazard named entity recognition. *Scientific reports* **12,** 4598 (2022).

81. Lindemann, B., Müller, T., Vietz, H., Jazdi, N. & Weyrich, M. A survey on long short-term memory networks for time series prediction. *Procedia CIRP* **99,** 650–655 (2021).

82. Absar, N., Uddin, N., Khandaker, M. U. & Ullah, H. The efficacy of deep learning based LSTM model in forecasting the outbreak of contagious diseases. *Infectious Disease Modelling* **7,** 170–183 (2022).

83. Journal du Net. *Python: définition et utilisation de ce langage informatique* `https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/`. accessed 2023.

84. JavaTpoint. *Introduction to TensorFlow* `https://www.javatpoint.com/tensorflow-introduction`. 2021.

85. Simplilearn. *What is Keras? A Comprehensive Guide to Deep Learning using Keras* `https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras`. 2021.

86. Techopedia. *Natural Language Toolkit (NLTK) - Definition* `https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk`. 2022.

87. Built In. *Pandas: A Comprehensive Guide* `https://builtin.com/data-science/pandas`. 2019.

88. Read the Docs. *Py-googletrans Documentation* `https://py-googletrans.readthedocs.io/en/latest/`. accessed 2023.

89. Android Police. *What is Google Colab? An Explainer* `https://www.androidpolice.com/google-colab-explainer/`. accessed 2023.

90. Kluyver, T. *et al. Jupyter Notebooks: A publishing format for reproducible computational workflows* `https://jupyter.org`. Accessed: May 30, 2023. 2016.

91. Moesif. *Building RESTful API with Flask* Moesif Blog. 2018. `https://www.moesif.com/blog/technical/api-development/Building-RESTful-API-with-Flask/`.

92. Bitext. *Training Dataset for Chatbots/Virtual Assistants* `https://www.kaggle.com/datasets/bitext/training-dataset-for-chatbotsvirtual-assistants`. 2022.

93. Mansithummar67. *Flipkart Product Review Dataset* Kaggle. Accessed: Month Day, Year. Month of dataset publication Year of dataset publication. `https://www.kaggle.com/datasets/mansithummar67/flipkart-product-review-dataset`.