Democratic and Popular Algerian Republic

Ministry of Higher Education and Scientific Research

University 20 Août 1955 Skikda

Faculty of Sciences

Department of Computer Science

Thesis

In order to obtain a Master's degree

Major: Computer Science

Specialization: Advanced Software Engineering and Applications

Theme

Transforming Sentiments:

Unleashing the Power of BERT for Arabic Tweet Sentiment Analysis

Presented by: **OUDJHANI Radia**

CHOUIT Ikdam

Publicly defended on: …/07/2023

In front of the jury composed of:

Chairman:	**Dr. BELAID Hassina**	University 20 Août 1955 - Skikda
Supervisor:	**Dr. BOUGAMOUZA Fateh**	University 20 Août 1955 - Skikda
Reviewer:	**Dr. RAMDANE Chafika**	University 20 Août 1955 – Skikda

Promotion 2022 – 2023

Special thanks

We would like to express our sincere gratitude to our supervisor, Dr. BOUGAMOUZA FATEH, for his patience, assistance, and guidance. We extend our thanks to all the professors of the Department of Computer Science at the University of SKIKDA. We also appreciate the contributions of our colleagues and everyone who has supported us throughout this work.

Dedication

First and foremost, we express our gratitude to God for granting us the determination and strength to successfully complete this work.

We dedicate this humble endeavor to our beloved parents, who have been our guiding light and unwavering support throughout our journey.

To our cherished families, whose love and encouragement have been the driving force behind our accomplishments, this work is dedicated to you.

We would also like to express our heartfelt appreciation to our dear friends, who have stood by us through thick and thin, offering their encouragement and valuable insights.

To all those who have played a significant role in shaping our path, including our parents, families, and friends, this dedication is a testament to your unwavering belief in us.

Abstract

Sentiment analysis is currently gaining significant attention in domains like politics, social sciences, marketing, and economics due to the influential role of opinions in decision-making processes. This study focuses on developing a sentiment analysis system specifically for Arabic texts. Unlike traditional approaches, we explore the potential of deep learning techniques, particularly BERT, in this field of automatic language processing. By leveraging BERT's advanced capabilities, we aim to enhance the accuracy and effectiveness of sentiment analysis in Arabic texts.

Keywords: sentiment analysis, Arabic text analysis, automatic language processing, deep learning, Bert

Résumé

L'analyse des sentiments suscite actuellement un intérêt croissant dans des domaines tels que la politique, les sciences sociales, le marketing et l'économie en raison du rôle influent des opinions dans les processus de prise de décision. Cette étude se concentre sur le développement d'un système d'analyse des sentiments spécifiquement pour les textes arabes. Contrairement aux approches traditionnelles, nous explorons le potentiel des techniques d'apprentissage profond, en particulier BERT, dans ce domaine du traitement automatique du langage. En exploitant les capacités avancées de BERT, nous visons à améliorer la précision et l'efficacité de l'analyse des sentiments dans les textes arabes.

Mots clés : analyse des sentiments, analyse des textes arabe, traitement automatique du langage, l’apprentissage en profondeur, Bert

ملخص

يحظى تحليل المشاعر حاليًا باهتمام كبير في مجالات مثل السياسة والعلوم الاجتماعية والتسويق والاقتصاد نظرًا للدور البارز للآراء في عمليات اتخاذ القرار. تركز هذه الدراسة على تطوير نظام لتحليل المشاعر خصيصًا للنصوص العربية. على عكس النهج التقليدي، نستكشف إمكانيات تقنيات التعلم العميق، بالتحديد Bert في هذا المجال من معالجة اللغة الآلية. من خلال استغلال قدرات Bert نهدف إلى تعزيز دقة وفعالية تحليل المشاعر في النصوص العربية

الكلمات الرئيسية: تحليل المشاعر، تحليل النصوص العربية، معالجة اللغة الآلية، التعلم العميق، Bert

CHAPTER 2: DEEP LEARNING

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 1

CHAPTER 2

CHAPTER 3

Abbreviation list

DL: Deep Learning

ML: Machine Learning

LSTM: Long Short-Term Memory

GRU: Gated Recurrent Unit

NLP: Natural Language Processing

MLM: Masked Language Modeling

AI : Artificial Intelligence

NN: Neural Network

ANN: Artificial Neural Network

SVM: Support Vector Machine

General Introduction

With the rapid growth of web applications and social media, there has been an abundance of user-generated reviews, comments, evaluations, and feedback. These opinions cover a wide range of topics, including products, politics, news, individuals, services, and events. It is crucial to process and analyze this wealth of information to gain a comprehensive understanding of user thoughts and sentiments. Prior to the availability of automated sentiment analysis tools, gathering customer feedback was an arduous and time-consuming task. This is likely why there is significant interest in this research field.

Sentiment analysis can contribute to creating a more comprehensive profile and inform future content and social media strategies. It can be regarded as a collection of algorithms implemented in software that detects and leverages opinions and emotions present in online social media resources. This interdisciplinary field draws upon natural language processing, text analysis, and computational linguistics techniques to extract subjective information.

While numerous sentiment analysis tools have been developed for English, we are venturing into new territory by proposing a sentiment analysis tool specifically designed for the Arabic language. This tool empowers Arabic-speaking users to analyze social media, providing them with the ability to gauge the overall sentiment surrounding current topics of discussion.

Purpose

In this project, our aim is to explore the field of sentiment analysis in Arabic texts, particularly by utilizing deep learning techniques. Deep learning has shown remarkable results in the English language, and therefore, we seek to develop a deep learning model and apply it to a corpus of Arabic data collected from social media. In our approach, we specifically employ the BERT model to enhance the accuracy and effectiveness of sentiment analysis in Arabic texts.

Work plan

This thesis is structured in three main chapters, in addition to the introduction, the general conclusion and the annex:

Chapter 1: The first chapter provides an overview of Natural Language Processing, including its classification, tasks, and fields. Additionally, it offers an overview of sentiment analysis, covering its categories, levels, and various approaches.

Chapter 2: The second chapter will provide an overview of deep learning, its applications, and the underlying principles. It will also include a discussion on transformers and their significance in the field.

Chapter 3: The third chapter will delve into the system architecture employed for the experiments, outlining the different components and their functionalities. Additionally, it will present the results obtained from the experiments and provide a comprehensive discussion on the findings.

Conclusion: Our work ends with a general conclusion.

Annex: This annex presents the tools we used and the implementation of our code.

Chapter 01

Natural Language Processing and Sentiment Analysis

1 Introduction

Natural Language Processing (NLP) is a field of study that enables computers to interact with humans using natural language. One of the most popular applications of NLP is sentiment analysis, which involves determining the emotional tone of a piece of text, such as a tweet or a product review.

Sentiment analysis has become increasingly important in recent years, particularly with the rise of social media and online reviews. Companies can use sentiment analysis to gain insights into how their customers feel about their products or services, and to identify areas for improvement. Researchers can also use sentiment analysis to study public opinion on various issues, such as political candidates or social movements.

In this chapter, we will provide an overview of NLP and sentiment analysis

2 Natural Language Processing

2.1 Definition

Natural Language Processing (NLP) is a field of study and application that focuses on analysing and representing text in natural language at one or more levels of linguistic analysis. The ultimate objective is to understand and manipulate text and words in natural language to perform useful tasks using computers, and to strive towards achieving a level of comprehension and analysis of natural language that is comparable to that of humans [1][2]**.**

Natural Language Processing (NLP) is an interdisciplinary field of computer science that encompasses a variety of areas of study. Its applications span multiple domains and rely on expertise from fields such as linguistics, artificial intelligence, machine learning, mathematics, and robotics, among others [3][2][4].

NLP is a subfield of computer science that aims to enable computers to understand language in a "natural" way, much like humans do. Typically, this refers to tasks such as sentiment analysis, speech recognition, and question answering. NLP has its roots in the development of computer science in the 1950s [5].

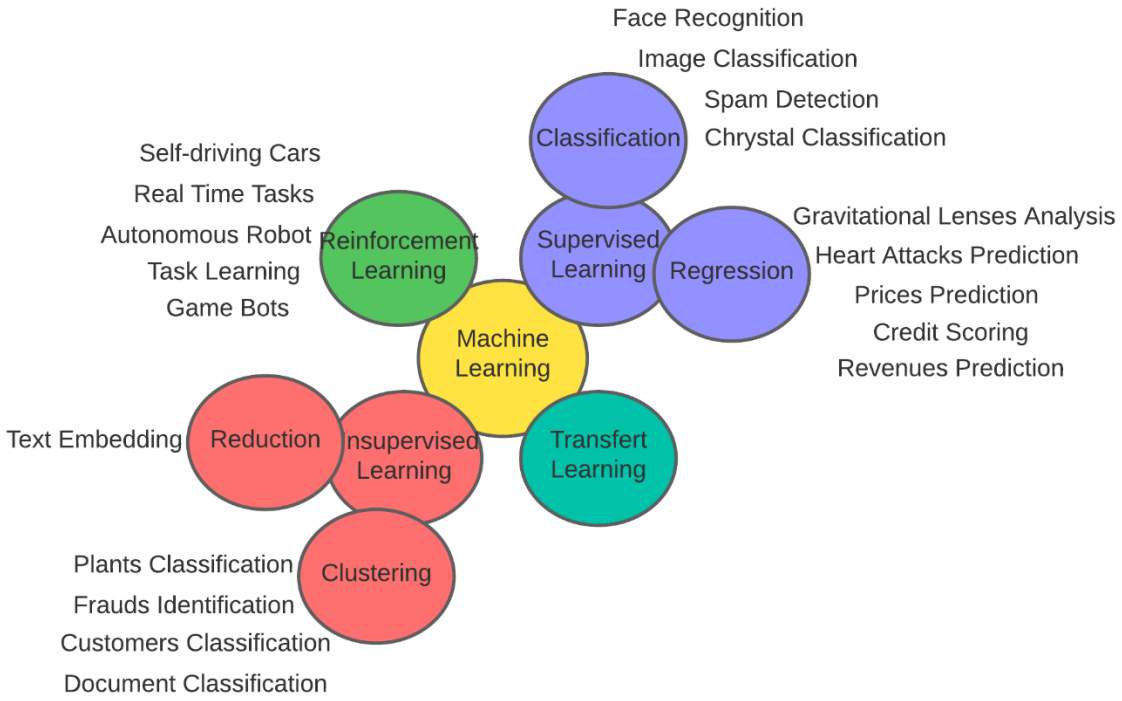

Figure 1.1 fields of Machine Learning

2.2 History

Natural Language Processing (NLP) has a history that began in the 1950s and 1960s. Significant milestones include Alan Turing's work on the "imitation game" and IBM's development of the first machine translation system. Over the years, NLP evolved with statistical approaches in the 1970s and 1980s, followed by machine learning techniques in the 1990s and 2000s. More recently, advancements like Recurrent Neural Networks (RNNs) and Transformer models have had a profound impact on NLP. Noam Chomsky, a prominent linguist, has contributed to the theoretical understanding of language structure and syntax, influencing NLP research.

2.3 Natural Language Processing classification

Natural Language Processing (NLP) encompasses two specific domains

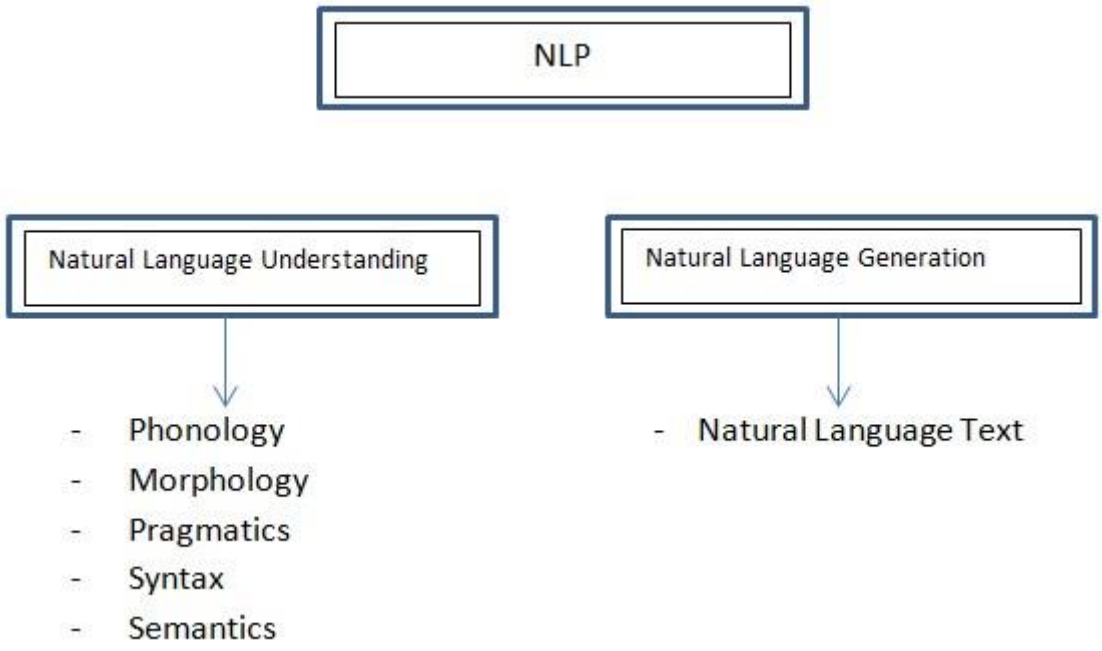

Figure 1.2 general classification of NLP [28].

2.3.1 Natural Language Understanding (NLU)

This involves analysing and interpreting human language input, such as text or speech, to derive meaning and context. In order to understand structure, Natural Language Understanding (NLU) attempts to solve the following ambiguity in natural language [6]:

- **Lexical ambiguity:** Words can have multiple meanings.
- **Syntactic ambiguity:** A sentence can have multiple parse trees.
- **Semantic ambiguity:** A sentence can have multiple meanings.

2.3.2 Natural Language Generation (NLG)

This involves using computer algorithms to generate human-like language output, such as text or speech. The process of Natural Language Generation (NLG) can be divided into three steps [6]:

- **Content determination:** This involves selecting the relevant information to be included in the generated text.

- **Discourse planning:** This involves organizing the selected information in a coherent and structured way, taking into account factors such as the intended audience and purpose of the text.
- **Surface realization:** This involves transforming the structured information into actual natural language text, including aspects such as grammar, syntax, and style.

2.4 Natural Language Processing tasks

The following section provides a brief overview of the four standard tasks in Natural Language Processing (NLP).

2.4.1 Part-Of-Speech Tagging

The process of Part-of-Speech (POS) tagging is essential to Natural Language Processing (NLP), as it assigns a unique label to each word in a text indicating its syntactic function, such as a noun, verb, or adjective. This information is the basis for other NLP tasks, including Named Entity Recognition, Sentiment Analysis, Question Answering, and word sense disambiguation [8]; [7].

Example:

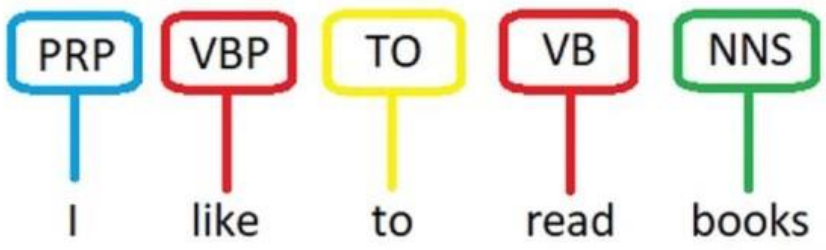

Figure 1.3 Example of Part-Of-Speech Tagging [29].

2.4.2 List of Tags and Descriptions

Tag	Description
PRP	pronoun
VBP	verb
TO	to
VB	verb
NNS	noun

Table 1.1 Tags and descriptions.

2.4.3 Named Entity Recognition (NER)

Identifying and categorizing named entities in text, such as names of people, organizations, locations, and so on [5]; [7].

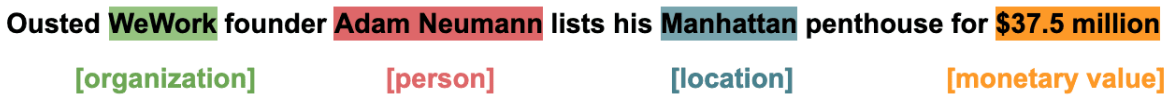

Figure 1.4 Example of NER [30].

2.4.4 Parsing or Chunking

Analysing the grammatical structure of a sentence by identifying the relationships between words and their dependents.

Example

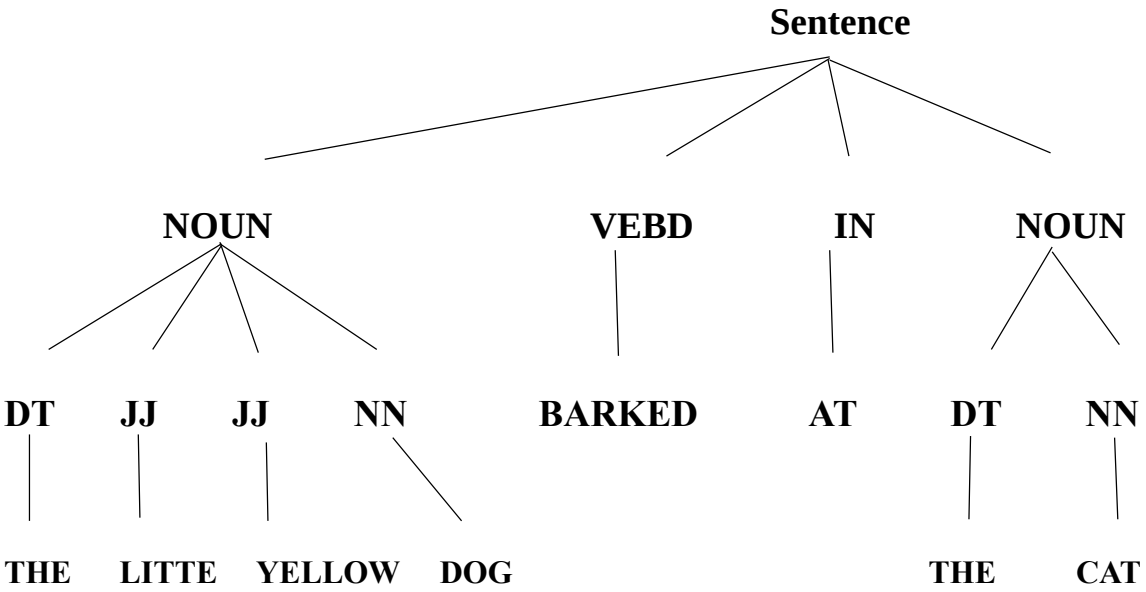

Figure 1.5 Example of Parsing.

2.4.5 Semantic Role Labelling (SRL)

SRL aims to assign a semantic role to a syntactic constituent of a sentence [7].

2.5 NLP Purpose

Natural Language Processing (NLP) is a field that focuses on transforming human language into a formal representation that computers can easily handle. Its main objective is to study fundamental problems related to natural language processing. NLP is particularly useful for modelling textual data and extracting information from it, which can then be represented differently [9] [10] [11].

2.6 Fields of application of Natural Language Processing

Natural Language Processing (NLP) has gained significant attention lately for its ability to analyse human language across a range of fields, including but not limited to:

- **Automatic translation:** the process of translating text from one language to another automatically.
- **Speech recognition:** the process of recognizing and transcribing spoken language.
- **Information extraction (retrieval):** the process of identifying and extracting useful information from text data.

- **Question answering:** the process of answering questions posed in natural language.
- **Text summarization:** the process of creating a summary of a longer piece of text.
- **Text classification:** the process of assigning text to predefined categories.
- **Named entity recognition:** the process of identifying and classifying named entities in text, such as people, organizations, and locations.
- **Spam detection:** the process of identifying and filtering unwanted messages from emails or other forms of communication.
- **Medical query processing:** the process of interpreting and responding to medical queries posed in natural language.
- **Sentiment analysis:** the process of determining the emotional tone of a piece of text.

These are just a few examples of the many fields of application of NLP.

2.7 Challenges

Natural Language Processing (NLP) faces several challenges, including:

- **Ambiguity:** Natural language can be highly ambiguous, which can lead to difficulties in accurately interpreting and processing text.
- **Context:** The meaning of a word or phrase can be highly dependent on the context in which it is used, which can be challenging to capture in an automated system.
- **Data quality:** NLP models rely heavily on large datasets for training, but these datasets can be noisy or biased, which can negatively impact model accuracy.
- **Multilingualism:** Processing multiple languages presents additional challenges, including differences in grammar, syntax, and vocabulary.
- **Privacy:** NLP models trained on large datasets of human language can sometimes pose privacy concerns if sensitive personal information is inadvertently captured or revealed.
- **Generalization:** NLP models trained on one type of data may struggle to generalize to new, unseen types of data.
- **Interpretability:** NLP models can be highly complex and difficult to interpret, which can make it challenging to diagnose and fix errors or biases.

These are just a few examples of the challenges faced by NLP. The field is constantly evolving, and new challenges are likely to arise as NLP systems become more advanced and widely used.

3 Sentiment analysis

3.1 Definition

Sentiment analysis, also known as opinion mining, is a subfield of computer science that is considered a part of natural language processing. Its purpose is to classify the sentiments expressed in texts. There are many other related names that refer to slightly different tasks, such as opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, and so on.

Sentiment analysis is generally performed using one of two basic approaches: rule-based classifiers, in which rules derived from the linguistic study of a language are applied to sentiment analysis [13]; [14]; [15], and machine learning classifiers, in which statistical machine learning algorithms are used to automatically learn sentiment signals [16]; [17].

Therefore, we must define the two terms sentiment and opinion because there is confusion between these two words in the literature and in the active research domain. In the Oxford dictionary [18], sentiment is defined as a viewpoint that is held or expressed and as an emotion. For the word opinion, it is a belief or judgment formed about something, which is not necessarily based on facts or knowledge but on the beliefs or opinions of a group or the majority of people. One could say that the term sentiment is more about a person's emotion about something, whereas the term opinion represents or shows a person's viewpoint.

Sentiment analysis is a process mainly based on text mining, which can be applied to verbatims from social media, reviews, forums, etc. Sentiments are generally classified into three types: negative, neutral, and positive. It can also be a multi-class classification, where the sentiment can have a neutral label or even a different label such as very positive, positive, neutral, negative, very negative. Labels can also be associated with emotions such as sadness, anger, happiness, etc.

3.2 Categorisation of sentiments

Sentences can be either objective or subjective. When a sentence is objective, no further fundamental task is required. As for a subjective sentence, its polarity (positive, negative, or neutral) can be studied, as shown in the figure.

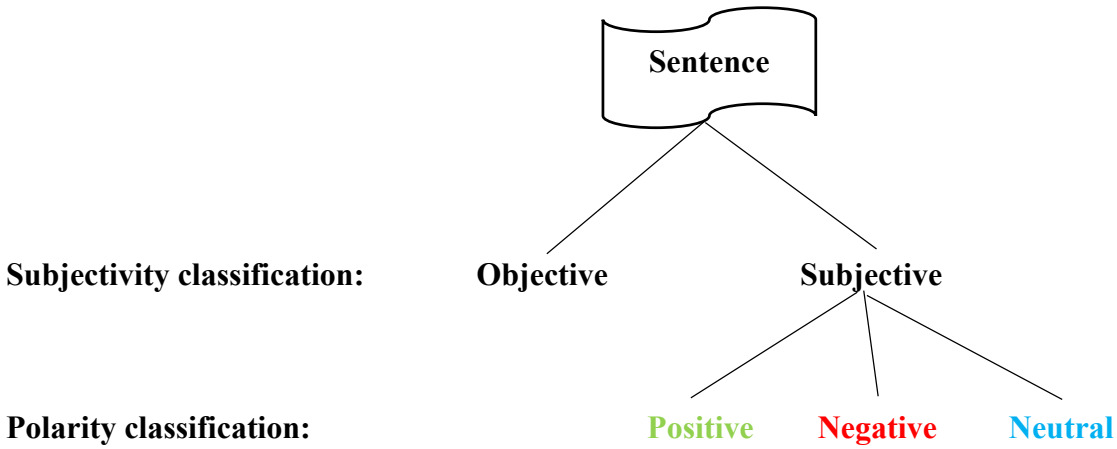

Figure 1.6 Polarity classification [31].

- **Subjectivity classification** is the task of distinguishing sentences expressing objective information from those expressing subjective views and opinions.
- **Polarity classification** is the task of distinguishing sentences that express positive, negative, or neutral polarities.

3.3 Levels of sentiment analysis

The first step when applying sentiment analysis is to define the text that will be analysed in a study. Generally, there are three levels of analysis: document-level, sentence-level, and aspect-level [19]:

3.3.1 Level of document

At the document-level, the goal is to determine the polarity of an entire text. The assumption is that the text expresses only one opinion about a single entity (for example, a single product). This level is the simplest form of classification [20].

3.3.2 Level of sentence

At this level, the unit of information being processed is a sentence, with the assumption that each sentence expresses a single opinion. This level of analysis is considered as a subjectivity classification.

3.3.3 Level of aspects

At this level, a more detailed and in-depth analysis is performed compared to the other levels. The basic step at this level is to identify and extract target entities. For example, the sentence "The iPhone is great, but battery life and security issues still need to be worked on" evaluates three aspects: the iPhone (positive), battery life (negative), and security (negative) - discovering exactly what people like and dislike [19].

3.4 Principal applications of sentiment analysis

As highlighted by Pang and Lee in "Opinion Mining and Sentiment Analysis" [21], "what others think" is regularly invoked in any decision-making process. It can find numerous applications in the field of prediction and monitoring. We briefly mention some applications below:

3.4.1 As a subset of systems

sentiment analysis systems can serve as an enhancement for other systems such as recommendation systems. Similarly, with regards to information extraction systems, it has been demonstrated that we can improve and make them more effective by parsing information in subjective sentences [22]. Question answering systems are another area where sentiment analysis can be beneficial [23].

3.4.2 The use by industries and governments and in politics

Today, political actors have followed the trend of sentiment analysis, as it is highly strategic to also know the opinion of internet users on a politician during a presidential election, for example, or by governments on decisions or something specific like elections. In the 2016 United States presidential election, Donald Trump said [24]: "I doubt I'd be here if it weren't for social media, to be honest with you." A study shows a strong correlation between estimates based on data from Google Trends and the outcome of several elections [25]. Similarly, for industries where sentiment analysis is part of their planning strategies for decision making, companies like Google, Microsoft, Hewlett-Packard, Amazon, Oracle, and Adobe have built or are building their own sentiment analysis systems.

3.4.3 Other fields of application

Sentiment analysis applications have expanded to almost all possible domains, such as the economy, with a focus on improving products and increasing sales and revenue for

businesses. Education is another area where sentiment analysis can be applied, as the results can help teachers and institutions take corrective measures regarding teaching methodology and course curriculum. Indeed, sentiment analysis can be applied to many areas beyond those mentioned. Here are some examples:

- **Marketing:** Companies can use sentiment analysis to understand how consumers perceive their products or services, which can help them adjust their marketing strategy accordingly.
- **Customer service:** Companies can also use sentiment analysis to understand how customers perceive their customer service, in order to improve their customer experience.
- **Politics:** Sentiment analysis can be used to track public opinion on government policies or election campaigns.
- **Social media:** Companies can use sentiment analysis to monitor discussions about their brand or product on social media, in order to understand customer opinions and preferences.
- **Recruitment:** Companies can use sentiment analysis to evaluate job candidates during the hiring process by analysing their responses and behaviours.
- **Healthcare:** Sentiment analysis can be used to track the mood and sentiments of patients as part of the treatment of mental health disorders.
- **Tourism:** Sentiment analysis can be used to track tourist opinions on tourist attractions, hotels, and restaurants, which can help businesses in the tourism industry improve their offerings.

These areas are just a few examples of the possibilities offered by sentiment analysis, which can be applied to almost any domain where people's opinions and emotions are important.

3.5 Sentiment analysis approach

It is crucial to start by clarifying the distinction between supervised and unsupervised learning techniques. In supervised learning, two distinct sets of data are used: a training set and a test set. The name "supervised" comes from the fact that the system is trained on a pre-labelled subset of data that has been already processed. Conversely, unsupervised learning utilizes a

single set of data, and the system autonomously reorganizes the information to group similar data points together.

There are three main approaches to sentiment analysis: lexicon-based approaches, corpus-based approaches, and hybrid approaches (mixing both) [26].

3.5.1 Lexicon-based approach

This approach involves identifying the polarity of a text by using two sets of words: those that express a positive sentiment and those that express a negative sentiment. The model counts the number of positive and negative words in the text, and the sum provides an overall evaluation of the text's sentiment. If the number of positive words exceeds the number of negative words, the text is considered positive; conversely, if the number of negative words outweighs the positive words, the text is considered negative. If the numbers are equal, the text is considered neutral. [26] An example algorithm is Support Vector Machine (SVM).

3.5.2 Automated sentiment analysis based on corpus

Requires the development of two manually annotated corpora. The first corpus serves as the training corpus, which is used to train an automated system. It contains notes added by human annotators. From these notes, the system should be able to perform an analysis autonomously. The second corpus is used as the test corpus. It is formed to verify the performance of the automated system. In an ideal scenario, the results of the analysis performed by the automated system would correspond 100% with those of the training corpus. In order for the performance of the automated system to be maximal, it is important that the training corpus is representative for the test corpus. [26] Example algorithm: Neural network.

3.5.3 Hybrid approach

This approach takes advantage of the two previous methods, and there are three ways to do it. The first is to use linguistic tools to create the corpus and then classify texts using a supervised learning tool. The second way is to use machine learning to establish the opinion corpus needed for the lexicon-based approach. The third way is to combine the two previous approaches and jointly use their results [27][28]. Example algorithm: Semi-Supervised Support Vector Machine (SVM).

4 Deep learning in sentiment analysis

Deep learning has revolutionized sentiment analysis, allowing models to learn hierarchical representations and capture contextual information. Prominent models include LSTMs and CNNs, which excel at capturing sequential dependencies and local patterns, respectively.

Pre-trained models like BERT have achieved state-of-the-art results by leveraging large-scale language understanding. Transfer learning and ongoing research in attention mechanisms and hybrid models further enhance sentiment analysis performance. Deep learning continues to push the boundaries of sentiment analysis, enabling the extraction of nuanced sentiment from diverse text sources. [30]

5 Conclusion

To summarize, this chapter showcased the significance of NLP and sentiment analysis. We examined various techniques, fields, and approaches. This can help for better understanding.

Chapter 02

Deep Learning

1.Introduction

In recent years, the field of artificial intelligence has witnessed a remarkable transformation. The introduction of machine learning and, more significantly, deep learning has had a profound impact on various domains, including natural language processing. A pivotal development within deep learning is the emergence of transformers. These innovative architectures have revolutionized the way we approach complex tasks by leveraging the power of attention mechanisms.

2. Deep learning

2.1 Definition

Deep learning is a type of machine learning that uses multiple layers of non-linear transformations to extract increasingly abstract representations of the input data. It achieves this by composing simple but effective modules that operate at different levels of representation-learning, each of which transforms the input representation into a higher-level, slightly more abstract representation. This hierarchical approach enables deep learning models to automatically learn features and patterns that are relevant for a given task, making it a powerful technique for a wide range of applications such as computer vision, natural language processing, and speech recognition[32].

2.2 Deep Learning Vs Machine Learning

2.2.1 Machine learning

Machine learning is a paradigm of artificial intelligence that enables computers to automatically learn from data and improve their performance at a task, without being explicitly programmed for it. Machine learning involves collecting and preprocessing data, training a model using a set of algorithms and statistical techniques, and using the trained model to make predictions or decisions on new, unseen data[33][34].

The two main types of machine learning are supervised learning and unsupervised learning.

- In supervised learning, the algorithm is trained on a labeled dataset where the input data and corresponding output labels are provided. The algorithm learns to map the

input data to the correct output labels and can make predictions on new, unseen data. Examples of supervised learning tasks include image classification, speech recognition, and regression analysis.

- In unsupervised learning, the algorithm is trained on an unlabeled dataset where the input data is provided but the output labels are not. The algorithm learns to find patterns and structure in the data, such as clusters or latent variables, and can be used for tasks such as anomaly detection, data compression, and exploratory data analysis. Examples of unsupervised learning algorithms include clustering, principal component analysis, and generative models.

2.2.2 The difference between deep learning and machine learning

In machine learning, algorithms need to be explicitly instructed on how to make predictions by providing them with relevant information, such as manually extracted features. However, in deep learning, the algorithm can automatically learn to make accurate predictions by processing data through an artificial neural network without the need for manual feature extraction. The structure of the neural network allows the algorithm to learn and improve its predictions over time.

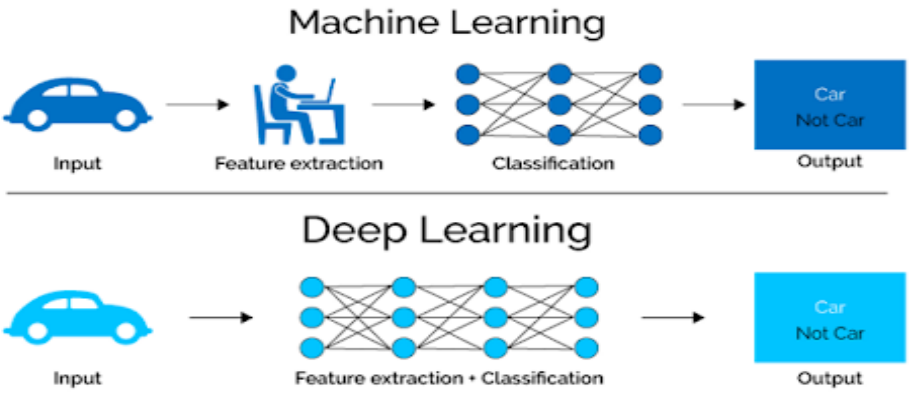

Figure 2.1 The difference between ML and DL from an architectural perspective[35].

In terms of results, the main difference between ML and DL is their performance on complex tasks. While ML can perform relatively well on simpler tasks with less data, DL can handle complex tasks with large amounts of data and produce significantly better results. This is because DL models have more layers of interconnected nodes that allow them to recognize patterns and make more accurate predictions.

However, this increased performance comes at a cost in terms of computational resources. DL models require more powerful hardware, such as Graphics Processing Units (GPUs), to process the large amounts of data needed for training. GPUs can significantly speed up the training process by parallelizing computations across multiple cores.

Another difference between ML and DL in terms of results is their interpretability. ML models are often easier to interpret and understand, as they rely on human-engineered features that can be easily interpreted. DL models, on the other hand, learn their own features through the training process, which can make them more difficult to interpret [32].

2.3 Important of Deep learning

The ability to accurately extract features from data is a crucial element of many machine learning models, and has traditionally required significant manual effort. However, deep learning models have automated this process, resulting in high accuracy rates for computer vision tasks. This power comes from the ability of deep learning algorithms to process large amounts of unstructured data, which is increasingly becoming available through the growth of the internet and the digitization of many aspects of our lives. However, it is worth noting that deep learning models can be computationally expensive and require access to large volumes of data in order to be effective. Nonetheless, there are already several companies gafam (Google, Apple, Facebook, Amazon and Microsoft) that have begun using deep learning to analyze their extensive amounts of data, and it is likely that the use of deep learning will continue to grow in importance as the world generates more and more data.

2.4 Aims of Deep Learning

Deep learning aims to explore how computers can exploit data to develop functionalities, use available data to learn how to make good decisions, acquire good knowledge to be able to give good answers on new, unknown examples by generalizing known examples to learn better. To achieve this, preprocessed data is necessary, for example, images are decomposed into simple shapes, which are then given to the artificial neural network to try to recognize them. Automatic feature extraction is another major advantage of deep learning. The ultimate goal of deep learning is to enable computers to learn and solve complex problems in a way that is similar to humans, by processing and analyzing vast amounts of data. Deep learning is particularly useful in fields such as image and speech recognition, natural language processing, and self-driving cars. In essence, deep learning enables machines to learn and

improve on their own without the need for explicit programming, leading to more intelligent and efficient decision-making.

2.5 Deep learning and sentiment analysis

Deep learning has demonstrated its effectiveness in tackling a wide range of complex problems[38]. Thanks to the use of artificial neural networks to learn and extract significant patterns and information from data. As a result, we can find numerous studies attempting to apply this approach as a solution to sentiment analysis challenges. In this context, we will now outline the essential tasks required to carry out this work.

2.5.1 Data preparation

The first step in sentiment analysis is data preparation, which involves loading a dataset. However, it is not enough to have a large amount of data. The quality of the data is also critical, and most issues arise from poor quality data. To ensure that models are trained correctly and provide accurate results, the data used must be representative, clean, precise, complete, and well-labeled. For example, if we want to predict sentiment from social media comments, the corpus must contain the same type of documents. Therefore, preparing the data is a crucial step in the sentiment analysis process.

2.5.2 Pre-processing of the corpus

Performing essential tasks like pre-processing and data cleaning is crucial before using a dataset to train models. During this phase, techniques may be applied to alter or remove words, for instance, eliminating non-Language target characters. Afterward, every word in the prepared corpus needs to be transformed into information about its features that can be utilized for learning. Presently, the most advanced text models use word embeddings or word vectors that are trained from text data.

2.5.3 Word Embedding

Word embedding are vector representations of words that capture semantic relationships between words and their contexts. They are used to improve the performance of natural language processing tasks like sentiment analysis and text classification. Traditional methods like one-hot encoding and bag-of-words models have limitations, as they don't capture any semantic information about words. Word embedding, on the other hand, encode the semantic and syntactic properties of words.

- **Techniques for Generating Word Embeddings**

Word embeddings are a way to represent words in a vector space, where words with similar meanings are located close to each other. In natural language processing, word embeddings are commonly used to improve the performance of various tasks, such as sentiment analysis, text classification, and machine translation. The word vectors capture not only the meaning of a word, but also its context. This means that words with similar meanings or related concepts will have similar vector representations. Moreover, the vectors should reflect the relationships between words. For instance, the vector for "**man**" should be close to that of "**king**", just as the vector for "**woman**" is close to that of "**queen**".

There are different methods for generating word embeddings, including count-based methods like TF-IDF and prediction-based methods like Word2Vec and Glove. Count-based methods rely on the frequency of each word in the text corpus to determine its importance, while prediction-based methods predict a word based on its context. Prediction-based methods have shown to be more effective in capturing semantic relationships between words and their contexts. However, both methods have their advantages and disadvantages, and the choice of method depends on the specific task and the available data.

Using word embeddings over traditional bag-of-words approaches has several benefits. Bag-of-words models represent each document as a collection of words, ignoring the order in which the words appear. This approach does not capture any semantic information about the words and treats all words as equally important. On the other hand, word embeddings can capture semantic relationships between words and their contexts, which can lead to more accurate and nuanced analysis of text[36].

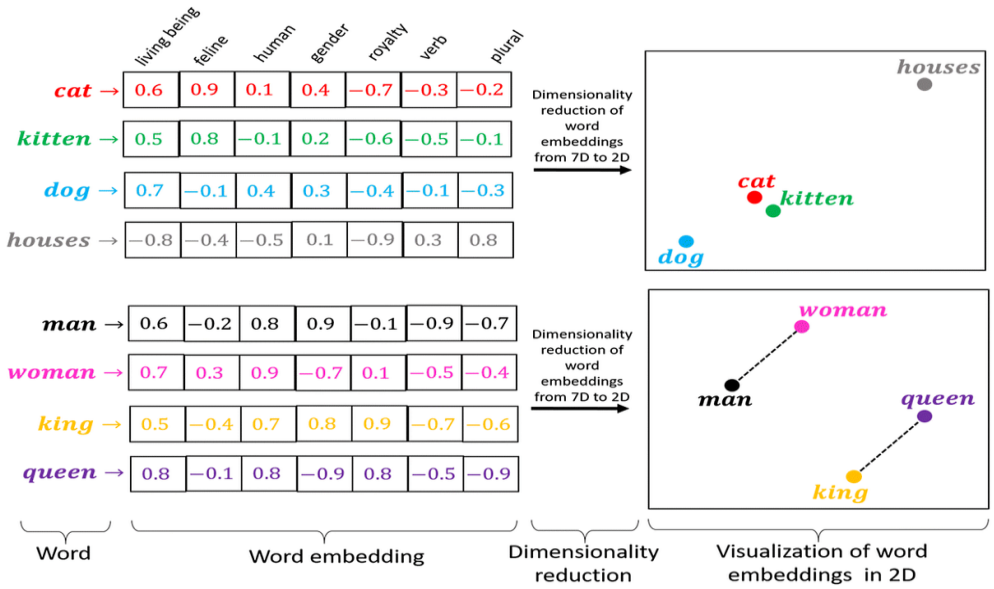

Figure 2.2 word embeddings map words in corpus of text[37].

3. Neural networks

3.1 Definition

All deep learning algorithms are based on neural networks, also known as ANNs (Artificial Neural Networks) [38]. ANN models simulate the functioning of a biological nervous system to process information, similar to how the brain processes information. Neural networks consist of interconnected neurons organized in layers.

3.2 Neurons

Neural networks are composed of artificial neurons that take inspiration from real neurons found in the human brain. A comparison between a real neuron and an artificial neuron is depicted in Figure 2.3 [38].

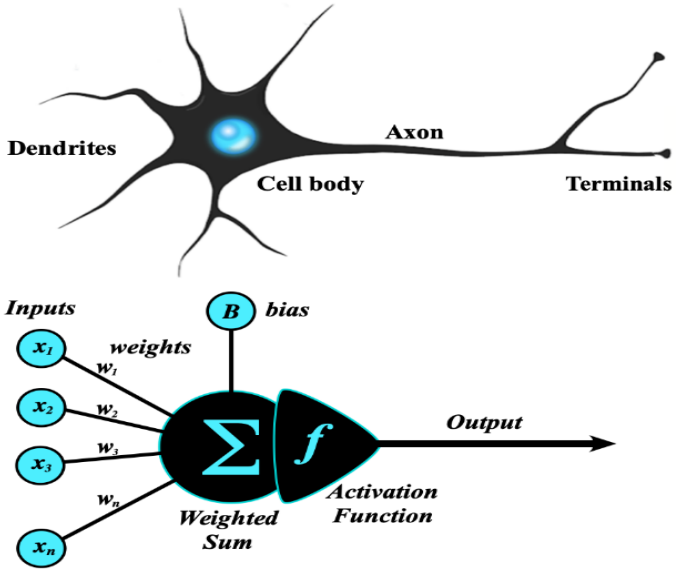

Figure 2.3 artificial neural vs biological neurons[39].

Just as we use our brains to identify patterns and classify different types of information, we can teach neural networks to perform the same tasks on data[34].

3.3 Development of neural networks

Neural networks have come a long way since their inception. We will explore the evolution of neural networks from their earliest forms to the modern deep learning architectures.

- **McCulloch and Pitts model**

Neural networks were first introduced in 1943 by Warren McCulloch and Walter Pitts in their paper "A Logical Calculus of Ideas Immanent in Nervous Activity"[40]. Their model showed how neurons could work together to perform complex computations, and this concept has since led to the development of modern neural networks.

- **The perceptron**

Frank Rosenblatt introduced the perceptron in 1957, which is a type of layered neural network. In this network architecture, neurons are organized into layers, with the basic structure comprising an input layer that connects to an output layer of computation nodes. This particular network model is considered strictly feedforward and is often referred to as a single-layer network. It's worth noting that the input layer is not included in the computation process [42].

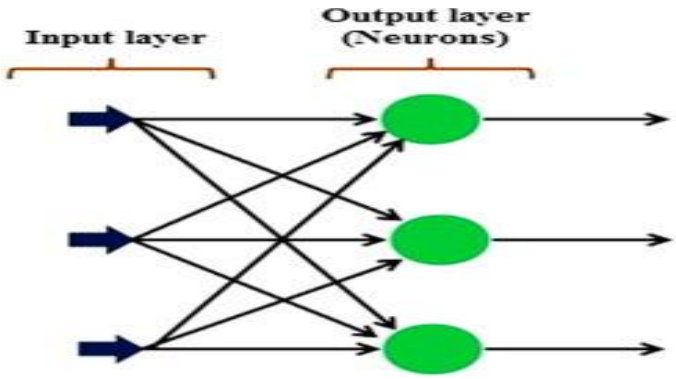

Figure 2.4 Feedforward network with a single layer of neurons[43].

- **Multilayer Feedforward Networks**

The concept of Multilayer Feedforward Networks, characterized by the presence of hidden layers, was introduced in the 1980s. Hidden layers consist of hidden neurons or hidden units that intervene between the input and output of the network, enabling the extraction of higher-order statistics from the input data. By incorporating one or more hidden layers, these networks can capture a global perspective despite their local connectivity. In such networks, the source nodes in the input layer provide input signals to the computation nodes in the second layer, which in turn serve as inputs to the subsequent layers. The output signals of the final layer of neurons represent the overall response of the network to the activation pattern supplied by the source nodes. Figure 2.5 illustrates a multilayer feedforward neural network with a single hidden layer, commonly referred to as a 4-5-3 network. This particular network configuration comprises 4 source nodes, 5 hidden neurons, and 3 output neurons [42].

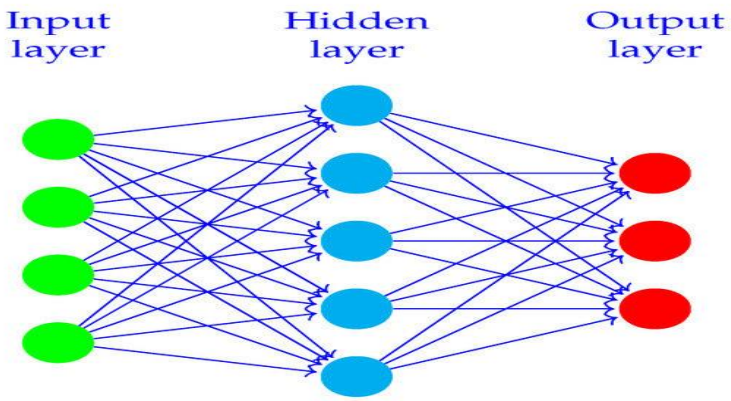

Figure 2.5 Fully connected feedforward network[44].

- **Convolutional Neural Networks (CNNs)**

Convolutional Neural Networks (CNNs) were first introduced in the 1980s and gained significant attention in the field of computer vision during the 1990s. (CNNs) are a type of neural network that excel at image and video processing tasks, they use convolutional layers to extract spatial features from the input data. The output of a convolutional layer is calculated using the following formula:

$$y(x) = \sigma\left(\sum_{ij} (f(i, j) * x(i, j)) + b\right)$$

where *f* is the filter, *x* is the input, and *b* is the bias. CNNs also typically include pooling layers to downsample the feature maps, reducing the number of parameters in the network. The architecture of a CNN can be deep, with many convolutional and pooling layers, and can include other types of layers such as fully connected layers and recurrent layers[42].

- **Recurrent Networks**

A recurrent neural network, which was first introduced in the 1980s, is different from a feedforward neural network because it has at least one feedback loop. In other words, the output signal of each neuron in a recurrent network is fed back to the inputs of all the other neurons. This feedback can either come from the hidden neurons or the output neurons. This type of architecture allows the network to have a nonlinear dynamic behavior, which can impact its learning capability and performance. The feedback loops are made up of unit-time delay elements, denoted by z-1, which can be nonlinear if the neural network contains nonlinear units[42].

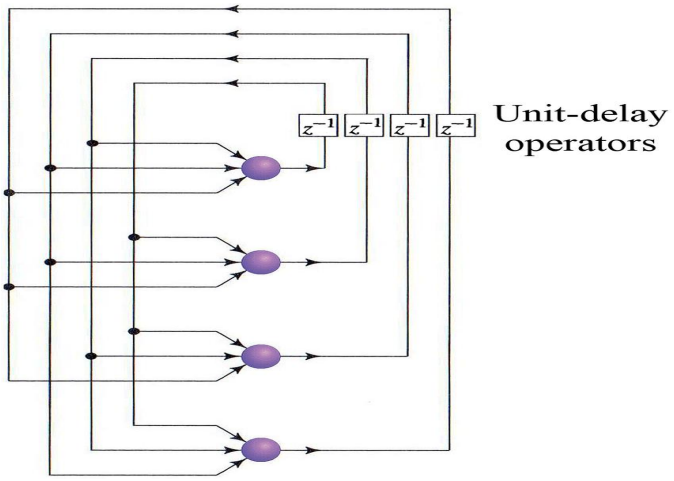

Figure 2.6 Recurrent network with no self-feedback loops and no hidden neurons[45].

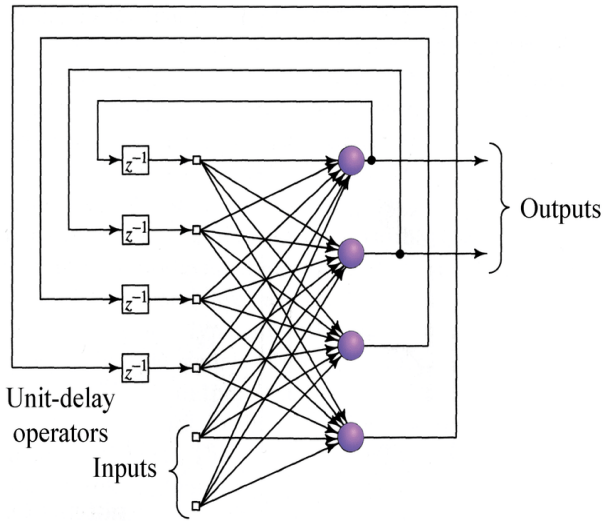

Figure 2.7 Recurrent network with hidden neurons[46].

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two popular types of recurrent neural networks (RNNs) that have overcome some of the limitations of traditional RNN architectures. LSTM and GRU were specifically designed to address the vanishing gradient problem, which arises when training deep neural networks with long-term dependencies. In the context of NLP, these RNN architectures have been commonly employed within the Encoder-Decoder framework. The Encoder part of the architecture creates a vector representation of a word sequence, whilst the Decoder component generates a sequence of words from a vector representation. LSTM and GRU have been utilized as key components in both the Encoder and Decoder.

LSTM introduces a memory cell, which allows the network to retain and propagate information over long sequences. The key idea behind LSTM is the use of gating mechanisms, including the input gate, forget gate, and output gate. These gates control the flow of information through the memory cell, enabling the network to selectively store or forget information as needed. The input gate determines how much new information should be stored, the forget gate controls what information to discard, and the output gate regulates the information to be passed on to the next time step. This gating mechanism allows LSTMs to effectively capture and utilize long-term dependencies in sequential data [47].

Similarly, GRU also addresses the vanishing gradient problem by employing gating mechanisms. It consists of a reset gate and an update gate. The reset gate determines how much of the past information should be forgotten, whilst the update gate controls how much of the new information should be incorporated into the memory. The reset gate and update gate in GRU work together to capture and propagate relevant information whilst discarding unnecessary information.

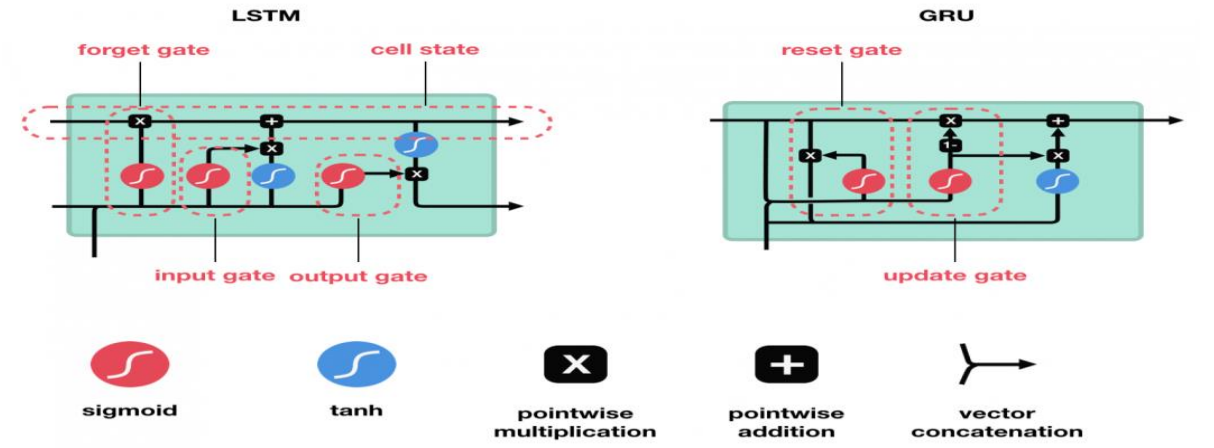

Figure 2.8 Gate Mechanisms in GRU and LSTM Models[48].

Both LSTM and GRU have shown remarkable performance in various NLP tasks, including sentiment analysis, machine translation, text generation, named entity recognition, and sentiment classification. These models excel in capturing contextual information, understanding the semantic meaning of words and phrases, and modelling temporal dependencies in sequential data—making them particularly well-suited for NLP tasks involving

text data with inherent sequential structures. Whilst LSTM and GRU have demonstrated significant advancements in NLP, they still have limitations. One major drawback is their computational complexity, which makes training and inference time-consuming, especially for large-scale applications. Additionally, these models can struggle with capturing long-term dependencies in very long sequences and may not always generalize well to unseen data. To overcome these challenges, the field of NLP witnessed a significant breakthrough with the introduction of the Transformer model. The Transformer model revolutionized NLP by departing from the traditional recurrent architectures like LSTM and GRU. Instead, it leveraged a self-attention mechanism to capture interdependencies between words in a sequence. This innovative approach significantly improved training speed and parallelizability compared to LSTM-based models [49].

Whilst LSTM and GRU played a crucial role in advancing NLP tasks within the Encoder-Decoder framework, the advent of the Transformer model marked a significant turning point. Its ability to efficiently process sequential data and capture interdependencies has revolutionized the field of natural language processing, offering more accurate and efficient methods for understanding and generating human language.

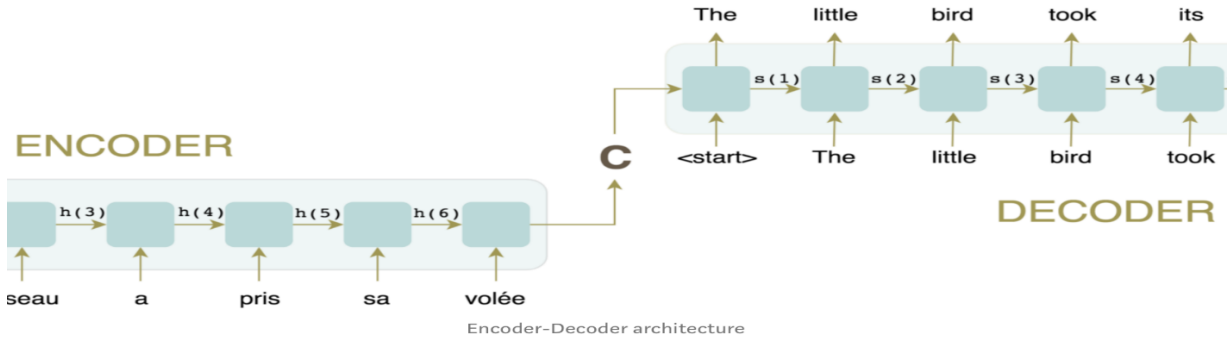

Figure 2.9 Encoder-Decoder mechanism[50].

4. Transformers

Recent advancements in natural language processing (NLP) have led to the development of transformers, which are a type of neural network architecture that has revolutionized the field of NLP. Transformers were first introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017, and have since become the dominant architecture for NLP tasks.

Transformers use self-attention mechanisms to process sequences of data, such as sentences or documents. The key innovation of transformers is their ability to process sequences in parallel, without the need for recurrent connections like RNNs, which can be slow and computationally expensive.

Self-attention allows the model to weight the importance of different words or tokens in a sequence, based on their context within the sequence. This allows the model to capture long-range dependencies and understand the relationships between different parts of the text. The self-attention mechanism also allows the model to adapt to different input lengths, making it more flexible than traditional RNNs.

Transformers have become the dominant architecture for many natural language processing (NLP) tasks, such as machine translation, language modeling, and question answering. This is because they are better suited for processing large amounts of text data, where long-range dependencies are common and the context of each word is important. [49]

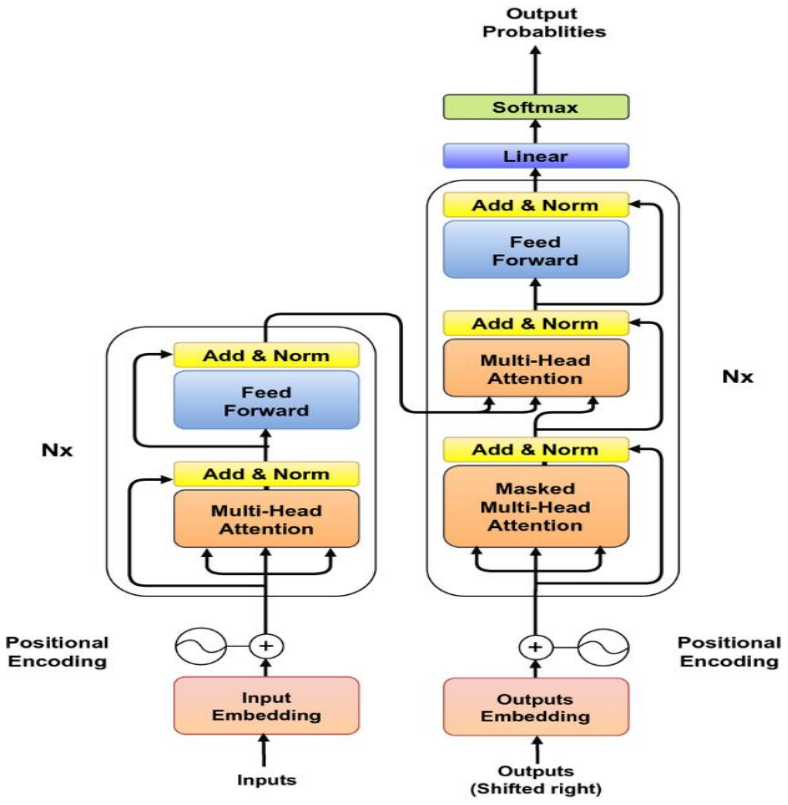

Figure 2.10 The Transformer architecture[51].

4.1 Transformer Architecture

The Transformer architecture has an encoder/decoder structure that processes input data. The model uses a self-attention mechanism instead of recurrent neural networks for faster processing.

4.1.1 Encoder

The encoder consists of N = 6 identical layers, each containing two sub-layers: a multi-head self-attention mechanism, and a fully connected feed-forward network. Residual connections are used around each sub-layer, followed by layer normalization. This means that the output of each sub-layer is LayerNorm (x+Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the integration layers, produce model outputs of dimension D=512 [49][52].

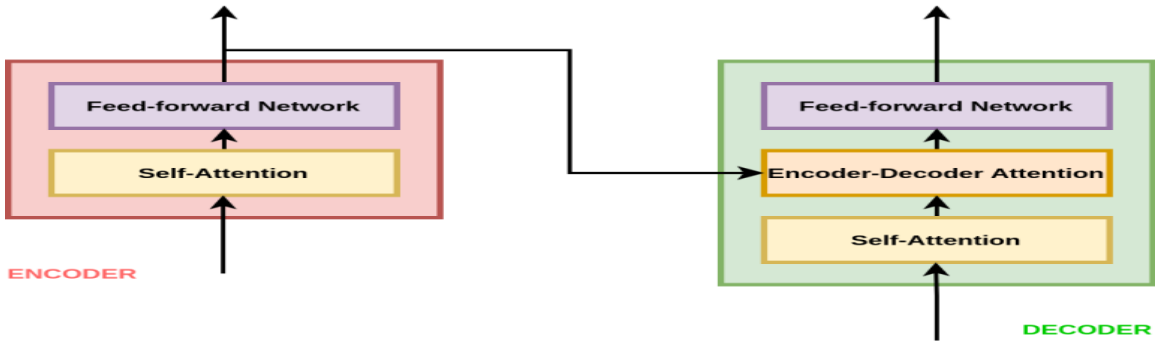

Figure 2.11 Bloc of encoder and decoder[53].

4.1.2 Decoder

The Transformer decoder is composed of a stack of N = 6 identical layers, each containing three sub-layers: a multi-head attention sub-layer, a feed-forward sub-layer, and a modified self-attention sub-layer. Residual connections and layer normalization are applied around each sub-layer. The decoder also uses masking and offset output embeddings to ensure that predictions for each position can depend only on known outputs at positions less than that position. The final decoder layer is connected to a linear neural network with softmax activation to identify corresponding vocabulary words for the decoder outputs.[49] Regardless of the number of encoders-decoders, the principle remains the same.

4.1.3 Attention

The attention mechanism in NLP measures the relationship between elements of two sequences. In sequence-to-sequence tasks, the attention mechanism helps the model focus on relevant words in one sequence when processing a word from another sequence. For instance, in the visualization of attention, the brighter the cell, the stronger the connection between the corresponding words. As a general rule, a word has a strong connection with its literal translation.[49].

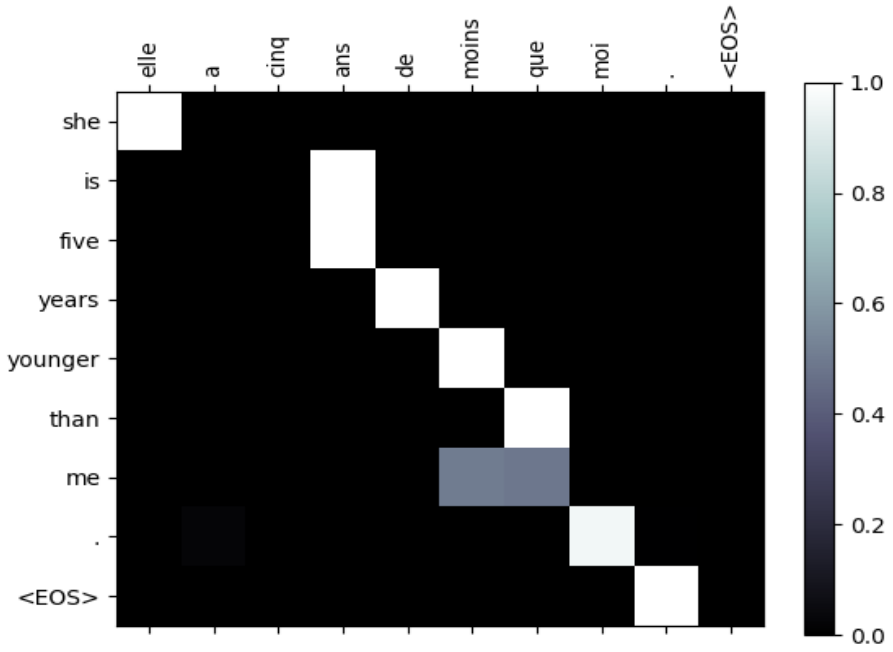

Figure 2.12 Distribution of attention between two sequences[54].

- **Self-attention**

Self-attention is a mechanism of attention applied to a single sequence, which determines the interdependence between different words within the sequence and associates it with a relevant encoding representation. For instance, given the sentence "The animal didn't cross the street because it was too tired," the process of self-attention helps to identify the relationship between "animal" and "it," recognizing that the pronoun refers to the former and not the latter. By visualizing a layer of attention over the sequence, we can observe that the attention coefficient for "animal" and "cross the street" are high for the word "it," indicating that the model needs to "focus" on these words when encoding "it."[49].

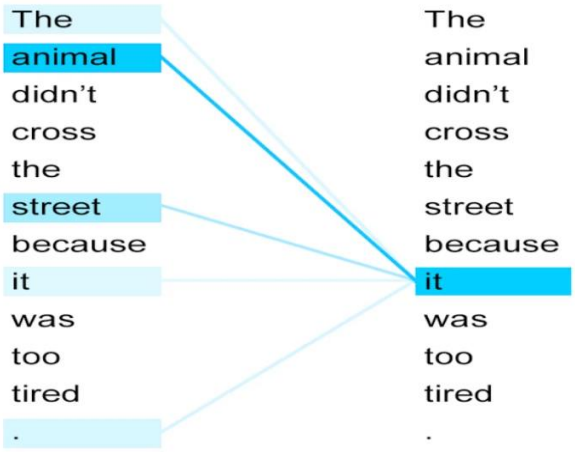

Figure 2.13 Visualization of a self-attention layer[55].

- **Calculation of (self) attention :**

The calculation of attention within the Transformer involves computing an attention vector for each word in the input sequence. This is done by considering three vectors for each encoder input: the query vector (q), the key vector (k), and the value vector (v). These vectors are obtained by multiplying the input sequence embedding (x) by three matrices (Q, K, and V) that are trained during the training process of the Transformer. The vectors are indexed by the position of the word in the sequence (e.g. q1, k1, and v1 for the first word in the sequence).

- **Calculation of self-attention vectors :**

The self-attention process starts by calculating the dot product of q1 and the different ki, divided by the square root of the dimension of k to stabilize the gradient.

Words	**q**	**k**	**v**	**Score/8**
I	q1	k1	v1	q1.k1/8
Am	k2	v2	q1.k2/8	q1.k2/8
your	k3	v3	q1.k3/8	q1.k3/8
father	k4	v4	q1.k4/8	q1.k4/8

Table 2.1 Claculation of Score/8.

Next, we apply the Softmax function to the vector "score/8". We denote the probability of the i-th word as s1i , as given by the Softmax. We then multiply each element of the vector s (the result of the Softmax) by the corresponding vector v.

Words	q	k	v	Softmax	Softmax*v
I	q1	k1	v1	s11	s11*v1
Am	q1	K2	V2	s12	s12*v2
Your	q1	K3	V3	s13	s13*v3
father	q1	K4	V4	s14	s14*v4

Table 2.2 Applying of Softmax function.

To conclude, we sum up the vectors obtained from the multiplication of Softmax and v, resulting in the self-attention vector of the word "I".

Words	q	k	v	Softmax	Summation
I	q1	k1	v1	s11	z1 = s11*v1 + s12*v2 + s13*v3 + s14*v4
Am	q1	K2	V2	s12	z2 = s12*v1 + s22*v2 + s23*v3 + s24*v4
Your	q1	K3	V3	s13	z3 = s13*v1 + s23*v2 + s33*v3 + s34*v4
father	q1	K4	V4	s14	z4 = s14*v1 + s24*v2 + s34*v3 + s44*v4

Table 2.3 Summation of vectors.

- **Multi-head Attention**

in addition, the computation is performed in parallel by multiple blocks of different attention. This is called "Multi-head Attention" and aims to have multiple "subspaces of representation" that prevent the representation from being completely biased if a layer (head) of attention is biased. The multi-head self-attention vector is simply the concatenation of the output vectors of each head.

In conclusion, transformers and their various components, such as self-attention and multi-head attention, have revolutionized the field of natural language processing by enabling models to learn contextual representations of text. BERT, a state-of-the-art language model, is an example of a transformer-based model that has achieved remarkable results on a wide range of NLP tasks. With the continued development of transformer-based models, we can expect to see even more impressive advancements in the field of NLP in the years to come.

5. Conclusion

In conclusion, this chapter has provided a comprehensive overview of deep learning and its significance in the field of artificial intelligence. We have highlighted the power and effectiveness of deep learning, particularly in sentiment analysis, where neural networks have demonstrated the ability to extract meaningful patterns and information from data. The exploration of the transformative Transformer architecture, with a focus on the concept of attention, has shed light on its potential for advancing natural language processing tasks. As a result, deep learning has emerged as a promising solution for sentiment analysis, and numerous contributions have been made to adapt this approach to address the challenges in this domain. This chapter serves as a solid foundation for further exploration and understanding of deep learning's implications in sentiment analysis and its potential for extracting valuable insights from textual data.

Chapter 03

Architecture And Experimental Analysis

1. Introduction

In this chapter, we present our conceptual framework for sentiment analysis in Arabic text. We propose utilizing the Bert model and Transformers to transform textual sentiments into numerical representations. We discuss the implementation of the different phases and highlight the achieved results.

2. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a revolutionary model in natural language processing (NLP), known for its exceptional performance and versatility in understanding and generating human language. Developed using the transformer architecture, BERT has achieved remarkable success in various language tasks, including sentiment analysis. What sets BERT apart is its ability to capture contextual information by considering both left and right contexts, enabling it to understand the intricate relationships between words and their broader context. Through pretraining on extensive unlabeled text data, BERT acquires deep language understanding, making it highly effective for fine-tuning on specific tasks and delivering impressive results across diverse applications [62].

2.1 BERT Architecture and Model Components

Our Arabic sentiment analysis system leverages the powerful architecture of BERT, specifically designed to handle the intricacies of Arabic text. BERT, based on a transformer-based model, incorporates multiple layers of transformers that allow us to capture the unique characteristics of Arabic language and effectively analyze sentiment.

One key feature of BERT is its self-attention mechanisms, which enable the model to assign weights to different words in a sentence based on their contextual relevance. By doing so, BERT gains a comprehensive understanding of Arabic sentence structures and can accurately capture long-range dependencies. This capability allows our system to grasp the intricate relationships between words, leading to a more nuanced analysis of sentiment in Arabic text.

To facilitate efficient training, BERT utilizes residual connections, which enable the smooth flow of gradients and prevent the issue of vanishing gradients. This feature allows our system to handle deeper architectures effectively and capture complex patterns and dependencies in Arabic language. The inclusion of residual connections enhances the overall performance of our sentiment analysis system.

Another important component of BERT is layer normalization, specifically tailored to Arabic text. This normalization process stabilizes the training process and promotes faster convergence. By normalizing the inputs within each transformer layer, BERT effectively captures the contextual representations necessary to understand the subtle meanings of words in their Arabic context.

The utilization of BERT's architecture and model components in our Arabic sentiment analysis system provides several advantages. It enables us to model long-range dependencies and capture the intricate relationships between words in Arabic text. By considering both left and right contexts, our system leverages a broader context to make more accurate sentiment predictions. The incorporation of self-attention mechanisms, residual connections, and Arabic-specific layer normalization enhances the overall performance of our system, enabling us to achieve a deeper understanding of sentiment in Arabic text.

2.2 Pretraining and Transfer Learning in BERT

In our system, BERT undergoes a crucial pretraining phase using a vast corpus of unlabeled text. It focuses on two key objectives: masked language modeling (MLM) and next sentence prediction (NSP). Through MLM, BERT predicts masked words, grasping contextual relationships and meaningful representations. In NSP, it captures sentence-level context by determining consecutiveness or swapping of sentences. The pretraining phase grants BERT deep understanding of language patterns and semantic representations.

2.3 Fine-Tuning BERT (Adapting General Language Representations to Sentiment-Specific Tasks)

The goal is to adapt BERT's general language representations to the requirements of sentiment analysis. We explore the process of fine-tuning and discuss strategies for effectively handling sentiment-specific data.

2.3.1 Fine-Tuning Steps

- Incorporating a Classification Layer: In our system, we enhance BERT for sentiment analysis by introducing a classification layer on top of the pre-trained BERT layers. This additional layer empowers the model to classify input text into sentiment categories, such as positive, negative, or neutral, based on the learned representations.

- Fine-tuning Model Parameters: During the fine-tuning process, we selectively update the parameters of both the added classification layer and the pre-trained BERT layers. While the majority of pre-trained weights are typically frozen to retain the general language understanding acquired during pretraining, we carefully adjust the model parameters to specialize in sentiment analysis. This fine-tuning allows the model to adapt and improve its performance on our specific sentiment classification task.

2.3.2 Handling Sentiment-Specific Data

To optimize fine-tuning, several techniques are employed for sentiment-specific data:

- Sentiment-Specific Labels: Using sentiment-specific labeled datasets during fine-tuning facilitates the model in capturing sentiment-related patterns. Training BERT with examples that convey positive and negative sentiments enhances its ability to discern sentiment nuances.
- Data Augmentation: Generating additional training examples by applying transformations or modifications to existing sentiment-labeled data helps to increase the diversity and coverage of the training set. This aids the model in better generalizing to different sentiment expressions.
- Domain Adaptation: Fine-tuning BERT on sentiment-specific data from the target domain or industry enables the model to adapt and specialize in analyzing sentiments within a specific context. This improves its performance and relevance to the target application.
- Transfer Learning: Leveraging pre-trained sentiment analysis models or sentiment lexicons supplements the fine-tuning process. By incorporating prior knowledge, the model can benefit from existing sentiment analysis expertise, capturing sentiment nuances more effectively.

3. System architecture

In this section, we provide an overview of our system, which focuses on sentiment analysis using BERT model.

3.1 Overall system architecture

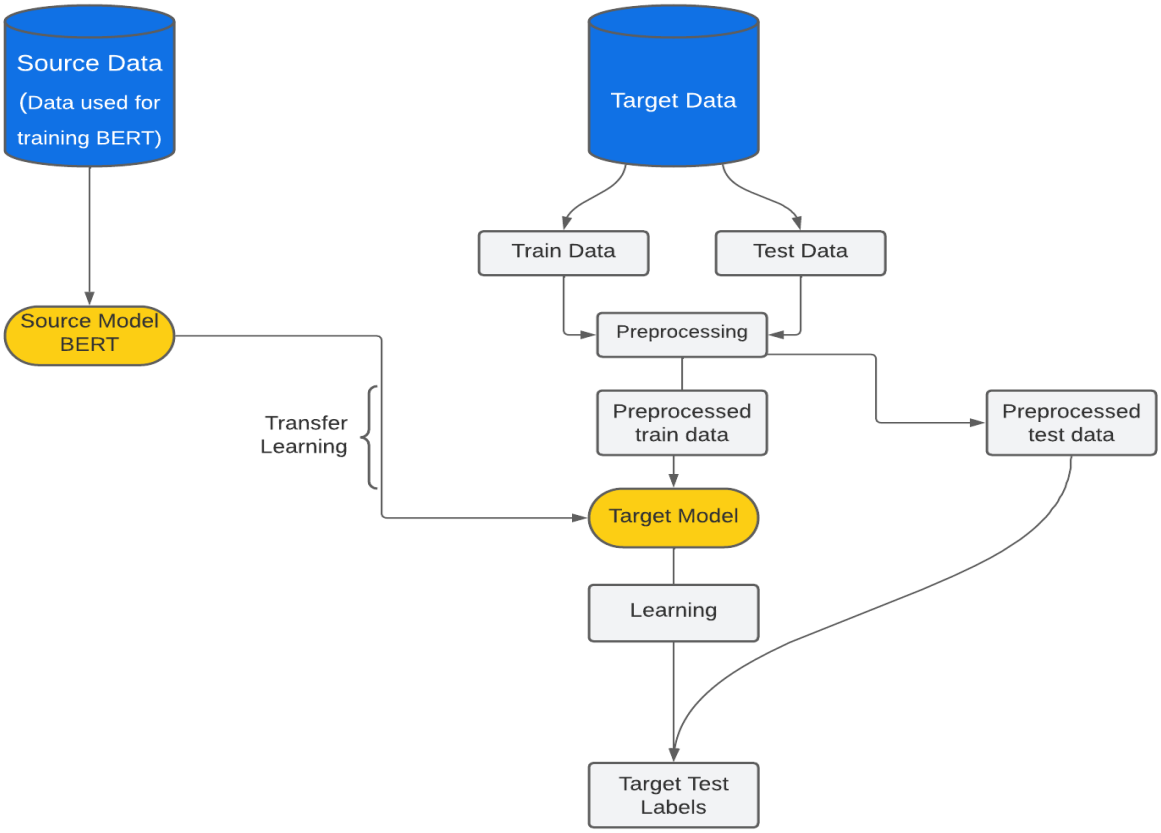

Figure 3.1 The Overall system architecture.

3.2 Detailed system architecture

3.2.1 Dataset

Social media platforms play a pivotal role in providing valuable data, especially regarding opinions and sentiments. With a vast user base, these platforms serve as a rich source for individuals to express their thoughts and feelings [41].

- **Importation**

In the realm of sentiment analysis, having a high-quality dataset is crucial for training accurate models. In our project, we obtained a substantial corpus of Arabic tweets by importing it from Kaggle, a renowned platform for sharing datasets. The dataset that named "***Arabic Sentiment Twitter Corpus"*** [63] consists of 58,000 tweets, which will serve as the foundation for our sentiment analysis task.

The dataset is divided into two main subsets: a training set and a test set. The training set comprises a total of 47,000 tweets (80%), the test set, which consists of 11,000 tweets (20%). This dataset contains two essential columns: "sentiment" and "content." The "sentiment" column indicates whether a tweet expresses a positive or negative sentiment, serving as the ground truth for our model training. The "content" column contains the actual text content of the tweets, providing the textual data on which our sentiment analysis will be performed.

By leveraging this existing dataset, we aim to train a sentiment analysis model using transformers. The availability of this pre-existing dataset of Arabic tweets allows us to streamline the development and evaluation process of our sentiment analysis model.

- **Preparation**

In the preparation of our dataset for sentiment analysis, we initially had distinct train and test parts, each containing positive and negative tweets. However, to streamline our analysis and simplify the process, we decided to perform a concatenation of the corresponding positive and negative tweet subsets within each part.

For the train part, we merged the train subset of positive tweets with the train subset of negative tweets. Similarly, for the test part, we combined the test subset of positive tweets with the test subset of negative tweets. This consolidation allowed us to create unified training and test datasets that encompass a comprehensive range of sentiments and provide a balanced representation. By combining both positive and negative tweets within each dataset, we ensure that our sentiment analysis model is exposed to diverse sentiments during training and evaluation. This approach enhances the model's ability to learn and generalize effectively, leading to accurate sentiment classification in Arabic tweets.

To illustrate these consolidation efforts, the subsequent table presents a clear overview of the merged train and test subsets:

Dataset Part	Original Positive Subset	Original Negative Subset	Merged Subset
Train	Train Positive	Train Negative	Combined Train (47k)
Test	Test Positive	Test Negative	Combined Test (11k)

Table 3.1 Splitting Dataset Overview.

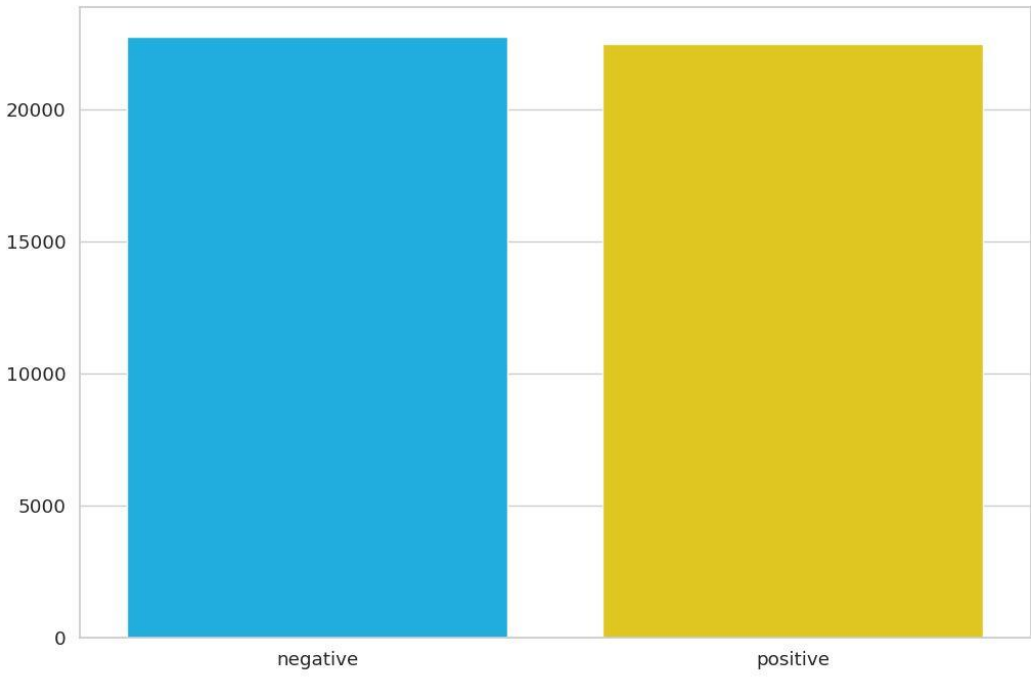

Figure 3.2 Balanced Distribution of Positive and Negative Classes.

sentiment	content
pos	💙 كلام جميل متكامل(من احبه الله جعل محبته ف قلوب البشر)

Figure 3.3 Example for positive Tweet.

sentiment	content
neg	😭 اعترف ان بس كانو شوي شوي يجيبو راسي لكن اليوم بالزايد

Figure 3.4 Example for Negative Tweet.

- **Exploration**

During the exploration phase we employed the use of word clouds to gain insights into the most frequent and impactful words used in the dataset. We created two separate word clouds, one for negative tweets and another for positive tweets. These word clouds provided a visual representation of the words that are prominent in each sentiment category. The word clouds were designed to display the words in Arabic, including the translation of emojis into English words.

Emojis, being an integral part of modern communication, were treated as words within the word clouds. This allowed us to capture the emotional expressions conveyed by users through emojis. Emojis were represented by their corresponding English words, which served as translations in the word clouds.

However, it's important to note that our plan moving forward is to further translate these English words into Arabic. By doing so, we will ensure a more accurate representation and understanding of the sentiments expressed in Arabic tweets, encompassing both textual words and translated emojis.

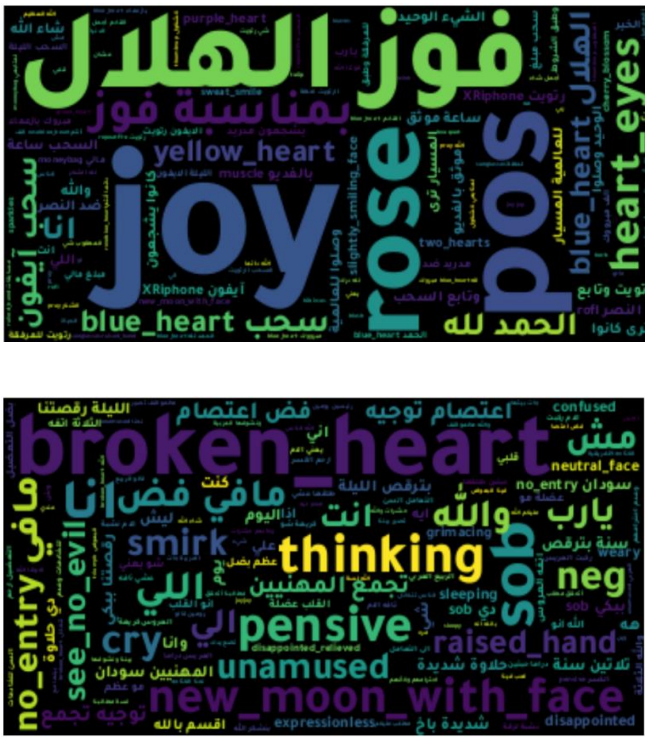

Figure 3.5 Pre-Translation Word Clouds (Visualizing Positive and Negative Sentiments).

Figure 3.6 Post-Translation Word Clouds (Visualizing Positive and Negative Sentiments).

3.2.2 Preprocessing

This phase is regarded as one of the crucial steps that come before the learning process since it focuses on extracting only relevant data. It involves two fundamental processes: basic preprocessing and specific NLP preprocessing.

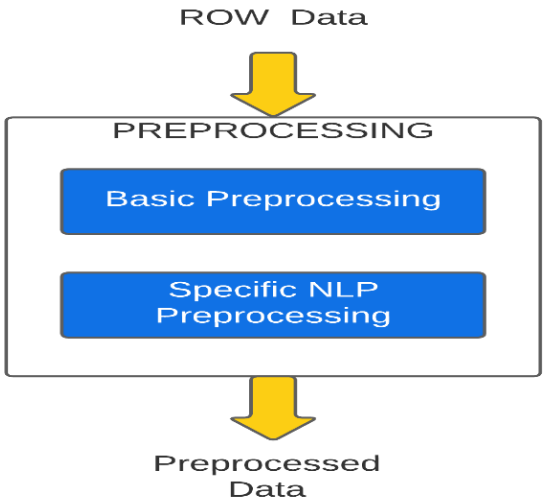

Figure 3.7 Preprocessing procedure.

A. Basic preprocessing

Basic preprocessing involves common text transformations to prepare raw text for analysis such as

- ✓ Normalization: Also referred to as standardization, is the process of converting a document into a standardized format that is easily manageable and manipulate. This process involves a series of steps, with the most important ones being:

• Removing specific characters such as /, ?, *, !, ., etc.

. • Removing numbers.

• Removing non-Arabic words and characters.

Figure 3.8 Process of Normalization.

- ✓ Removing usernames and URLs: Eliminating Twitter usernames and URLs from the text, as they do not contribute to sentiment analysis and may introduce noise.

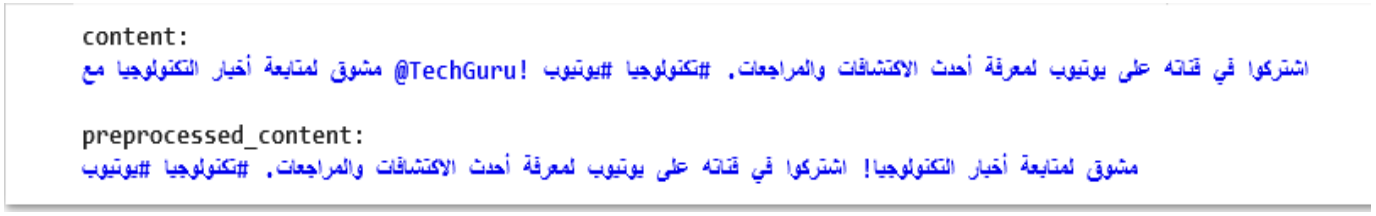

Figure 3.9 Process of Eliminating username.

- ✓ Removing stop words: When working with the Arabic language, the process of normalization often includes removing stop words. Stop words are commonly used words that do not carry significant meaning in the context of text classification or analysis. These words typically include articles, prepositions, conjunctions, and other frequently occurring words.

Examples of common stop words in Arabic include:

من، على، في، الى، ان، هذا، ثم، ف، نحن...

By eliminating these words from the text, we can streamline the analysis process and extract more meaningful insights from the remaining content.

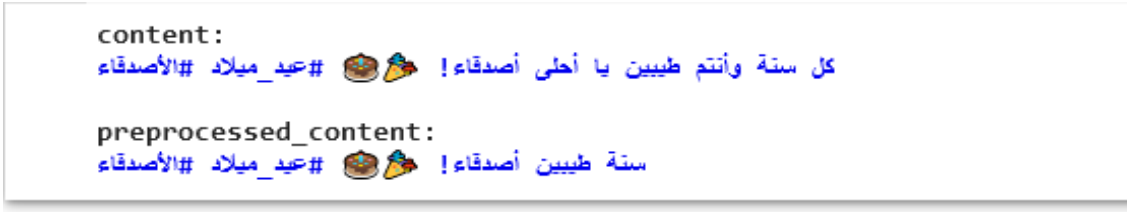

Figure 3.10 Process of Eliminating stop words.

These are the stop words present in the example:

Stop Words
كل
و
أنتم
يا

Table 3.2 Stop Words Detected in the Example.

- ✓ Removes diacritics: Diacritics are small signs or marks placed above or below characters in Arabic script to represent syllables or phonetic modifications. While diacritics serve linguistic purposes, they can often be safely disregarded in certain text analysis tasks.

Removing diacritics involves eliminating these additional marks from the text, resulting in a simplified representation of the Arabic words. By doing so, the focus shifts towards the core letters and their root forms, enabling a more streamlined analysis of the textual content.

B. Specific NLP preprocessing

- ✓ Lemmatization: Converting words to their base or dictionary form to reduce inflectional variations. For example, converting **"كتبت"** to **"كتب"**.

In Arabic, lemmatization can be challenging due to the rich morphological variations of words.

Figure 3.11 Process of Lemmatization.

In this lemmatized version, the words "**طيبين**" and "**أصدقاء**" have been lemmatized to their base forms "**طيب**" and "**صديق**", respectively. This process helps in reducing inflectional variations and bringing the words to their canonical forms, which can be useful for analysis or further processing.

- ✓ Emoticon/Emoji Analysis: we explore the concept of Emoticon/Emoji Analysis, which involves understanding and interpreting the sentiment conveyed by emoticons or emojis used in digital communication. Emoticons and emojis are graphical symbols that people use to express their emotions, ideas, or reactions in text-based conversations. Emoticons or emojis can carry valuable sentiment information, as certain symbols are commonly associated with specific emotions or feelings. By analyzing these symbols, we aim to enhance the accuracy of sentiment analysis algorithms.

Figure 3.12 Process of Emojis analysis.

- ✓ Vectorization : We recognize the importance of addressing the informality of texts by converting them into numerical vectors before the learning process. This step allows us to capture the formality of the texts effectively. To achieve this, we have implemented a set of steps using the BERT model. Here are the steps involved in creating vectors:
  - Tokenization: Splitting the tweet text into individual words or tokens. In Arabic, tokenization requires considering the specific rules of the Arabic language, as words are often connected. The sentence " كل سنة وأنتم طيبين يا أحلى أصدقاء! 🎂🎉 #عيد_ميلاد #الأصدقاء" is tokenized into subword units using WordPiece tokenization. For example, "يا احلى اصدقاء" might be tokenized into ["يا", "احلى ", "أصدقاء"].
  - Adding Special Tokens: Special tokens like [CLS] and [SEP] are added to the beginning and end of the tokenized sequence. The sentence becomes "[CLS] كل سنة وأنتم طيبين يا أحلى أصدقاء! 🎂🎉 #عيد_ميلاد #الأصدقاء [SEP]".
  - Mapping to IDs: Each token, including the special tokens, is mapped to a unique numerical ID from the BERT vocabulary. The token "كل" might be mapped to ID 101.
  - Padding and Truncation: If needed, the tokenized sequence is padded or truncated to a fixed length to ensure consistent input size for the model.
  - Generating Embeddings: The tokenized and mapped sequence is fed into the BERT model, which performs contextualized word embedding. It captures the contextual

meaning of each word based on its surrounding words. The embeddings for the tokens in the example sentence are computed.

- Output Vectors: The BERT model outputs numerical vectors representing each token in the sequence, including the special tokens. These vectors capture the contextual information and semantic associations of the words in the sentence.
- Sentence Embedding: BERT also generates an overall sentence embedding that summarizes the contextual information and captures the meaning of the entire input text.

3.2.3 Data Loading

Data Loading phase involves preparing the data for analysis, we start by creating a variable called `train_dataset` that represents a dataset containing the training samples. Similarly, we create another variable called `test_dataset` to hold the dataset for testing. These datasets are created using the available data, such as `X_train` and `X_test`, along with a tokenizer and a maximum length parameter. To facilitate the analysis process, we utilize data loaders. Specifically, we create a `train_loader` which is responsible for providing batches of training samples during the analysis, and a `test_loader` does the same for testing, to handle the management of the respective datasets.

By utilizing these data loaders, we ensure efficient handling of the datasets, enabling us to process the data in batches rather than all at once. This approach improves the overall performance and memory usage during the analysis phase.

The use of data loaders in our system streamlines the process of loading and managing the data, making it easier to work with large datasets and enhancing the overall efficiency of the analysis.

3.2.4 Classification

After the Data Loading phase, the next crucial step in our system is classification which contain two parts learning and test. This phase involves learning from labeled training data and using the acquired knowledge to predict the sentiment or class labels of unseen text samples. By leveraging the extracted features, our classification model aims to make accurate predictions and gain valuable insights from the text data.

A. Learning

During the training phase of our sentiment analysis system, we focus on training the model to accurately classify sentiment in Arabic text. We leverage the Arabertv02 model, which is specifically designed for Arabic language processing tasks, and utilize the Transformers library for implementation. To train the model, we split the training data into a training set and a validation set, with the validation set containing 20% of the data. We iterate through the training set in multiple epochs, processing the data in batches. Each batch consists of a subset of the training data.

For each batch, we feed the input text data and their corresponding sentiment labels into the Arabertv02 model. Through a forward pass, the model generates predictions for the sentiment labels. We compare these predictions with the ground truth labels using a loss function to compute the loss. Using the AdamW optimizer, we update the model's parameters by back propagating the gradients through the network. This optimization process allows the model to learn from the training data and adjust its weights to better capture sentiment patterns in the text.

Throughout training, we monitor the training loss, which represents the average loss over all batches in each epoch. This loss provides insights into the convergence and progress of the training process. We also evaluate the model's performance on the validation set after each epoch by computing the validation loss. This loss serves as a measure of how well the model generalizes to unseen data.

Through this training process, our goal is to train the model to accurately classify sentiment in Arabic text, enabling it to generalize well to unseen data and make reliable predictions.

B. Test

Once the training phase is complete, we move on to the testing phase of our sentiment analysis system. This phase allows us to assess the performance and effectiveness of the trained model on unseen data. During testing, we utilize a separate test set that consists of Arabic text samples with their corresponding sentiment labels. We apply the trained model to this test set and generate predictions for the sentiment labels.

By comparing these predicted labels with the ground truth labels, we can evaluate the accuracy of the model's sentiment classification. We calculate the accuracy as the ratio of correctly classified samples to the total number of samples in the test set. The accuracy serves as a measure of how well the trained model performs in accurately classifying sentiment in real-world Arabic text. A higher accuracy indicates a better performance and a stronger ability to generalize to new, unseen data.

Through the testing phase, we gain valuable insights into the model's overall performance and its effectiveness in real-world scenarios. This evaluation allows us to make informed decisions about the model's readiness for deployment and its potential use in sentiment analysis tasks in Arabic language applications.

4. Results and Discussion

	BERT model	LSTM model	Cross-validated model
Accuracy	**0.82**	**0.79**	**0.80**

Table 3.3 Accuracy Results.

- **BERT model**

In our experimentation, we carefully selected the "bert-base-arabertv02" model as our foundation for sentiment analysis on Arabic text. This pre-trained model, combined with a specialized tokenizer, provided a powerful platform for our task.

To optimize the model's performance, we fine-tuned several key hyper parameters. We set the batch size to 16, enabling efficient parallel processing of multiple samples during training. The maximum sequence length was defined as 128 tokens, striking a balance between capturing sufficient contextual information and controlling computational resources. Moreover, we employed a learning rate of 2e-5, allowing the model to effectively adjust its parameters during training. This value was carefully chosen to ensure a stable convergence and prevent overfitting. The model was trained for a total of 10 epochs.

- **LSTM model**

we conducted several trials to optimize the performance of our LSTM model for sentiment analysis. We carefully selected hyper parameters to fine-tune the model's behavior. The model was trained using a batch size of 16, ensuring efficient processing of multiple samples simultaneously. The model architecture consisted of an embedding layer with a dimension of 128, which mapped the input tokens to dense vector representations. This embedding dimension allowed the model to capture intricate semantic relationships between words.

The LSTM layer had a hidden dimension of 256, which determined the number of hidden units in the LSTM cells, and to optimize the learning process we used a learning rate of 0.001, which controls the step size during parameter updates. The model was trained for a total of 10 epochs.

- **Cross-validated model**

In this experimentation, we employed a cross-validation using our BERT model. The model was trained with a batch size of 16, a learning rate of 2e-5, and a maximum sequence length of 128. We implemented a 5-fold cross-validation technique to ensure robust evaluation of the model's performance. The training process was conducted over 7 epochs.

4.1 Performances Evolution

we present the confusion matrices and the curves depicting the validation loss and training loss for each experiment which provide insights into the evolution of performance throughout the training process.

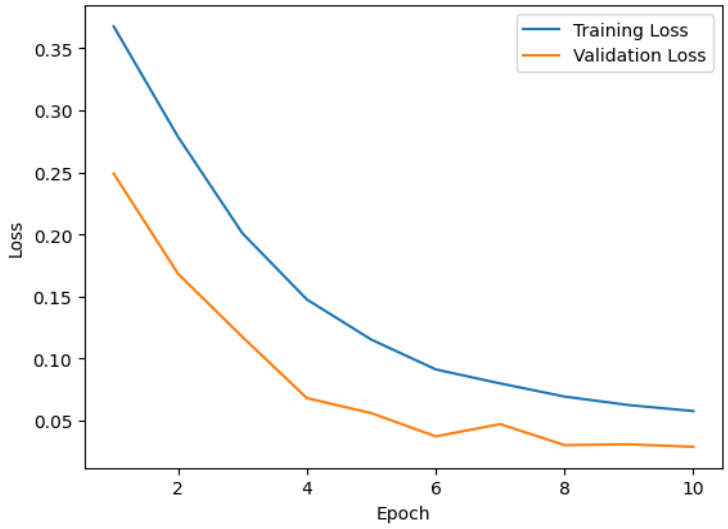

Figure 3.13 BERT Model Curves.

The training and validation loss values are decreasing with each epoch, indicating that the model is effectively learning and improving its performance. And The validation loss is consistently lower than the training loss, indicating that the model is not overfitting and is performing well on unseen data.

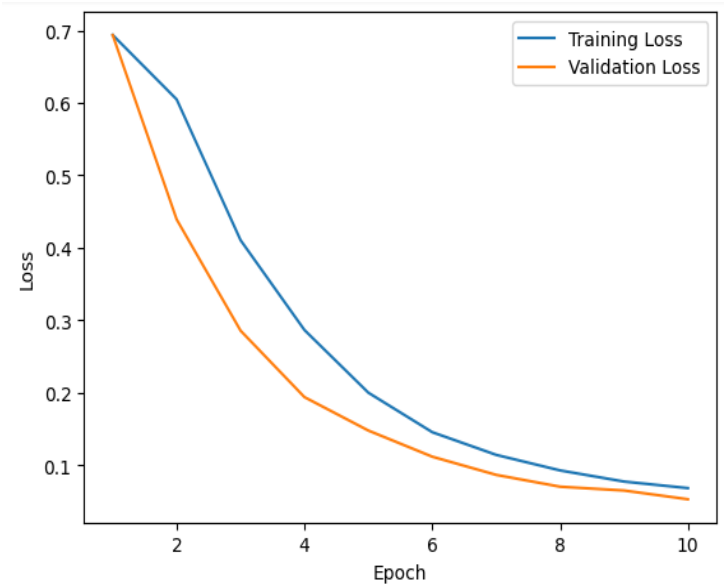

Figure 3.14 LSTM Model Curves.

The training and the validation loss decreases consistently with each epoch, this suggests that the model is effectively learning from the training data and generalizes well to unseen data and making accurate predictions.

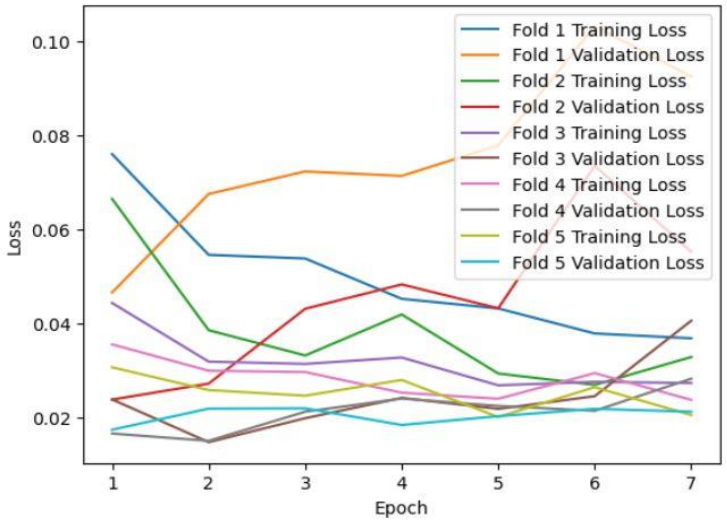

Figure 3.15 Cross-Validated Model Curves.

In the training and the validation loss we observe a decreasing and stable trend consistently with each epoch across all folds because the model already has good accuracy so has reached a relatively optimal state and has learned to effectively capture the patterns and features in the training data.

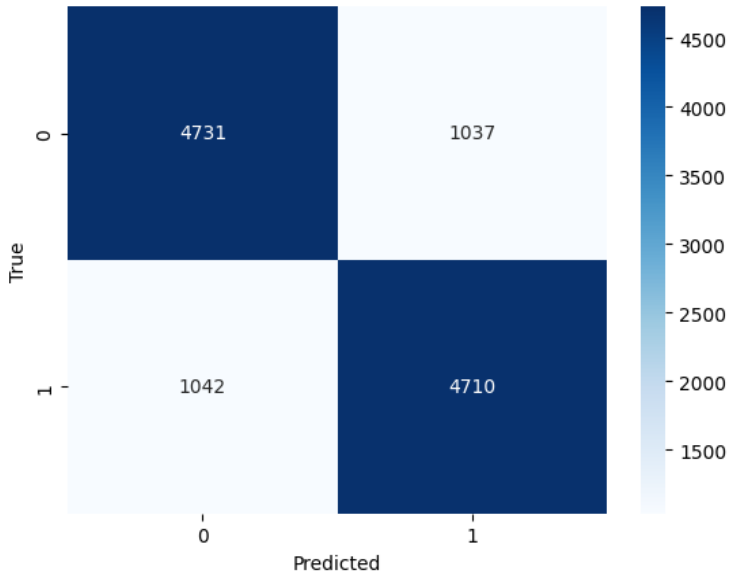

Figure 3.16 BERT Model Confusion Matrix.

In this case, the model correctly predicted 4731 instances as Class 0 (True Negative) and 4710 instances as Class 1 (True Positive). However, it also misclassified 1037 instances of Class 0 as Class 1 (False Positive) and 1042 instances of Class 1 as Class 0 (False Negative).

The precision, recall, and F1-score for both classes are approximately equal, indicating a balanced performance in predicting each class. The overall accuracy of the model is 82%, which considers the correct predictions for both classes. These results highlight the model's ability to capture patterns and make accurate predictions, but there is still room for improvement to reduce misclassifications.

	precision	recall	f1-score	support
0	0.82	0.82	0.82	5768
1	0.82	0.82	0.82	5752
accuracy			0.82	11520
macro avg	0.82	0.82	0.82	11520
weighted avg	0.82	0.82	0.82	11520

Figure 3.17 BERT Model Classification Report.

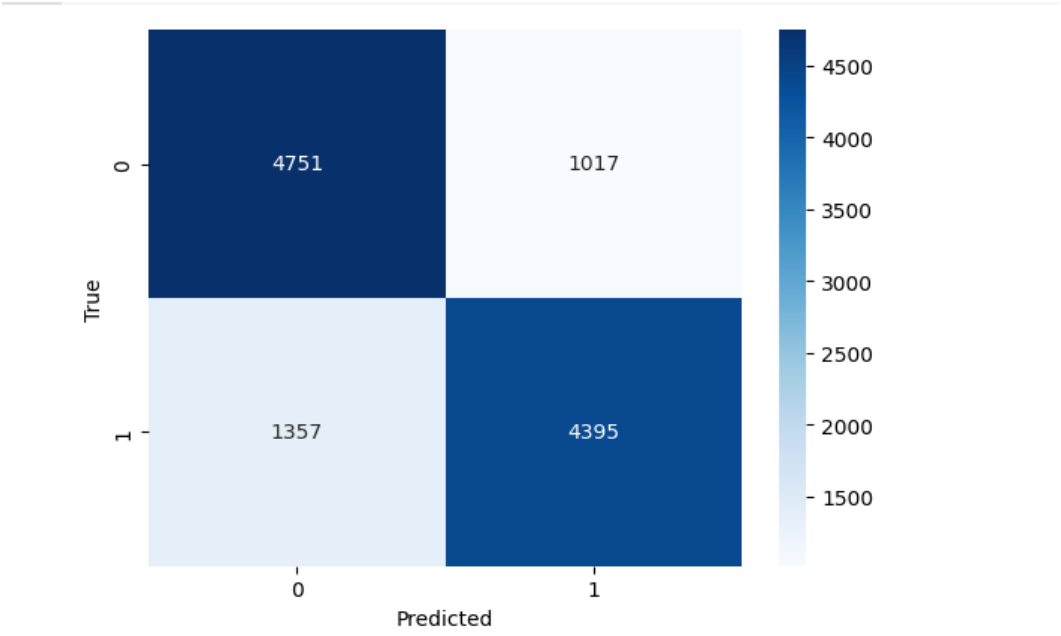

Figure 3.18 LSTM Model Confusion Matrix.

In this case, the model accurately predicted 4751 instances as Class 0 (True Negative) and 4395 instances as Class 1 (True Positive). However, it made 1017 false positive predictions, classifying instances of Class 0 as Class 1, and 1357 false negative predictions, classifying instances of Class 1 as Class 0.

The precision, recall, and F1-score for both classes are approximately equal, indicating a balanced performance in predicting each class. The overall accuracy of the model is 79%, considering the correct predictions for both classes. These results demonstrate the model's ability to capture patterns and make accurate predictions, but there is still room for improvement to reduce the number of misclassifications.

	precision	recall	f1-score	support
0	0.78	0.82	0.80	5768
1	0.81	0.76	0.79	5752
accuracy			0.79	11520
macro avg	0.79	0.79	0.79	11520
weighted avg	0.79	0.79	0.79	11520

Figure 3.19 LSTM Model Classification Report.

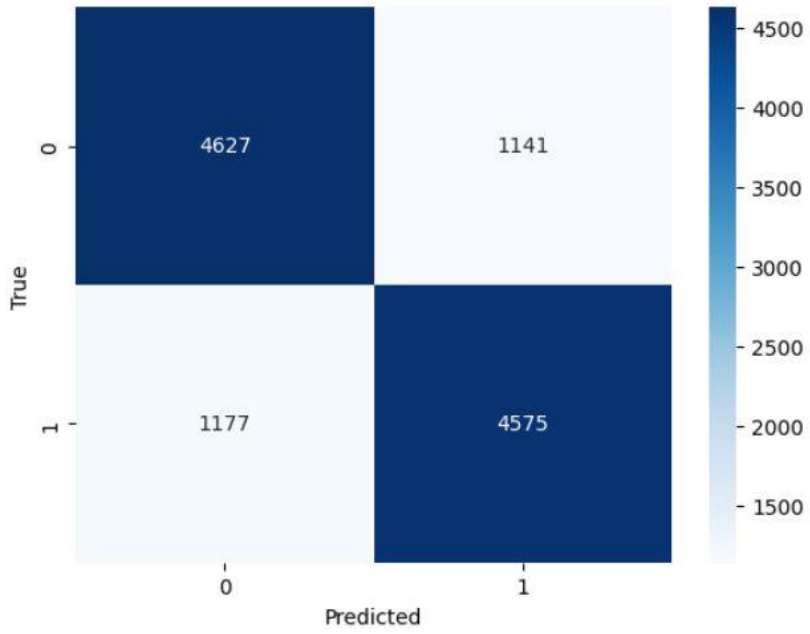

Figure 3.20 Cross-Validated Model Confusion Matrix.

In this case, the model correctly predicted 4627 instances as Class 0 (True Negative) and 4575 instances as Class 1 (True Positive). However, it also misclassified 1141 instances of Class 0 as Class 1 (False Positive) and 1177 instances of Class 1 as Class 0 (False Negative).

The precision, recall, and F1-score for both classes are approximately equal, indicating a balanced performance in predicting each class. The overall accuracy of the model is 80%, which considers the correct predictions for both classes. These results demonstrate the model's ability to capture patterns and make accurate predictions, although there is still room for improvement to reduce misclassifications.

	precision	recall	f1-score	support
0	0.80	0.80	0.80	5768
1	0.80	0.80	0.80	5752
accuracy			0.80	11520
macro avg	0.80	0.80	0.80	11520
weighted avg	0.80	0.80	0.80	11520

Figure 3.21 Cross-Validated Model Classification Report.

5. Conclusion

In this chapter, we have presented how to implement the different steps to create a sentiment analysis model. We have previously explained in detail the design of these steps.

Furthermore, we have conducted a simple comparison between the BERT model, LSTM model, and a cross-validated model. We observed that the BERT model pretrained on the Arabic language yielded relatively better results compared to the others.

General Conclusion

In this study, we have developed a sentiment analysis system based on the capabilities of deep learning, particularly the Transformer model. We leveraged the power of Transformers, such as BERT, to convert textual sentiments into numerical vectors, aligning them with the sequential nature of our data. This approach has shown promising results in accurately analysing sentiments in Arabic texts. Lastly, this work provided us with a practical application of our knowledge in neural networks, allowing us to enhance our understanding. Most importantly, it served as our initial foray into the realm of deep learning, one of the most significant fields of artificial intelligence.

To further enhance our model, several future directions can be explored:

- Testing our model on additional datasets: It is important to evaluate the performance and generalizability of our sentiment analysis model on different datasets to ensure its effectiveness across various contexts.
- Looking ahead, we envision extending our sentiment analysis systems to handle Arabic texts expressed in Latin letters and those containing non-Arabic words. These represent vast amounts of data that have yet to be extensively explored. By utilizing the Transformer architecture, we aim to unlock new opportunities for sentiment analysis in diverse Arabic text domains.
- Improving negation detection: Negation plays a crucial role in sentiment analysis, as it can significantly alter the polarity of a statement. Enhancing our model's ability to accurately detect and handle negation will lead to more precise sentiment analysis results.
- Introducing a "neutral" label: Currently, our model focuses on detecting positive and negative sentiments. However, including a separate label for neutral comments would provide a more comprehensive analysis, acknowledging comments that express neither positive nor negative opinions.

By addressing these areas, we aim to refine and expand the capabilities of our sentiment analysis model, ensuring its applicability to diverse datasets and enhancing its accuracy in sentiment detection.

REFERENCES

[1] Christopher W., Gerasimos S, Gerhard W., *Flexible Deep Neural Network structure with application to Natural Language Processing*. Department of Knowledge Engineering, Maastricht, University, The Netherlands.

[2] Fateme Behzadi, *Natural Language Processing and Machine Learning: A Review* International Journal of Computer Science and Information Security. (IJCSIS)

[3] Andreas V., Douglas T., Milco W., Jeremy A., Katie G., Lei X., Holly W., 2017. *Final Report on Natural Language Processing*. Version: 1.0 (final)

[4] Gobinda G. Chowdhury, *Natural Language Processing.* Dept. of Computer and Information, Sciences, University of Strathclyde, Glasgow, UK

[5] Taweh Beysolow, *Applied Natural Language Processing with Python Implementing Machine Learning and Deep Learning Algorithms for Natural Language Processing*

[6]https://www.supinfo.com/articles/single/6041-machine-learning-introductionapprentissage-automatique access 3/5/2023

[7] Ronan C., Jason W., Léon B., Michael K., Koray K., Pavel K., 2011. *Natural Language Processing (almost) from Scratch*

[8] Akshay K., Adarsha S., *Natural Language Processing Recipes Unlocking Text Data with Machine Learning and Deep Learning using Python*

[9] Silvia F., Eric S., Juan M., Torres M., 2007. *Énergie textuelle de mémoires associatives*.

[10] Ronan C., Jason W., *A Unified Architecture for Natural Language Processing: Deep Neural, Networks with Multitask Learning.*

[11] Alexis C., Holger S., Yann Le Cun, 2016. *Very Deep Convolutional Networks for Natural Language Processing.*

[12] Diksha K., Aditya K., Kiran K., Sukhdev S., *Natural Language Processing: State of The Art, Current Trends and Challenges.*

[13] Brooke, J., Tofiloski, M., Taboada, M.: Cross-linguistic sentiment analysis: From English to Spanish. In: Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing, Borovets, Bulgaria, pp. 50–54 (2009)

[14] Denecke, K.: Using SentiWordNet for multilingual sentiment analysis. In: IEEE 24th International Conference on Data Engineering Workshop, ICDEW, pp. 507–512. IEEE (2008).

[15] Elarnaoty, M., AbdelRahman, S., Fahmy, A.: A Machine Learning Approach For Opinion Holder Extraction Arabic Language. CoRR, abs/1206.1011 (2012).

[16] Abbasi, A., Chen, H., Salem, A.: Sentiment analysis in multiple languages: Feature selection for opinion classification in Web forums. ACM Transactions on Information Systems.

[17] Liu, B.: Sentiment analysis and subjectivity. In: Handbook of Natural Language Processing, pp. 627–666 (2010) (TOIS) 26(3), 12 (2008)

[18] Oxford Dictionary. (n.d.). Definition of Sentiment Analysis. Retrieved from https://en.oxforddictionaries.com/definition/sentiment access 05/7/2023

[19] Sentiment analysis: mining opinions sentiments, and emotions. Bing Liu. First edition USA 2015.

[20] Study of Different Levels for Sentiment Analysis. Seema Kolkur, Gayatri Dantal and Reena Mahe. International Journal of Current Engineering and Technology. 2015.

[21] Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135. Retrieved from http://www.cs.cornell.edu/home/llee/omsa/omsapublished.pdf. DOI: 10.1561/1500000001.

[22] Exploiting subjectivity classification to improve information extraction. E. Riloff, J. Wiebe, and W. Phillips. in Proceedings of AAAI. 2005.

[23] Qualitative dimensions in question answering: Extending the definitional QA task. L.V. Lita, A.H. Schlaikjer, W. Hong, and E. Nyberg. 2005.

[24] Baynes, C. (2019, March 27). Trump says he would not be President without Twitter. Retrieved from https://www.independent.co.uk/news/world/americas/us-politics/donald-trump-tweets-twitter-social-media-facebook-instagram-fox-business-network-would-not-be-a8013491.html access 05/7/2023

[25] Spyros E. Polykalas, George N. Prezerakos and Agisilaos Konidaris, « A General-Purpose Model for Future Prediction Based on Web Search Data: Predicting Greek and

Spanish Election »,27th International Conference on Advanced Information Networking and Applications Workshops, 2013

[26] Cynthia Van Hee, L'analyse des sentiments appliquée sur des tweets politiques : une étude de corpus, Faculté associée de linguistique appliquée Université Bruxelles Belgique, 2013.

[27] Damien Poirier et Françoise Fessant et Cécile Bothorel et Emilie Guimier de Neef et Marc Boullé, Approches Statistique et Linguistique Pour la Classification de Textes d'Opinion Portant sur les Films, Revue des Nouvelles Technologies de l'Information RNTI-E-17, 2010, 147-169.

[28] https://www.educba.com/introduction-to-nlp/ access 3/5/2023

[29] https://byteiota.com/pos-tagging/ access 3/5/2023

[30] https://monkeylearn.com/blog/named-entity-recognition/ access 3/5/2023

[31] https://www.sciencedirect.com/topics/computer-science/polarity-classification access 3/5/2023

[32] Yann LeCun, Yoshua Bengio Geoffrey Hinton Nature 521, 436–444 (28 May 2015)

[33] "What is Deep Learning and How Does It Work?" by John Brownlee: https://builtin.com/artificial-intelligence/deep-learning access 3/5/2023

[34] An Introduction to Machine Learning Interpretability" by Christoph Molnar. This PDF provides an overview of methods for interpreting and explaining machine learning models. Link: https://christophm.github.io/interpretable-ml-book/ access 3/5/2023

[35] https://www.turing.com/kb/ultimate-battle-between-deep-learning-and-machine-learning access 24/6/2023

[36] Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research, 57, 345-420.

[37]https://medium.com/@hari4om/word-embedding-d816f643140 access 21/6/2023

[38] Python Deep Learning. Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, Valentino Zocca. 2019

[39] https://mriquestions.com/what-is-a-neural-network.html access 3/5/2023

[40] McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5(4), 115-133.

[41] Chaima KIHEL ; 2019/2020 ; Biskra ; " Analyse des sentiments en utilisant l'apprentissage Profond : Cas de la langue Arabe "

[42] Haykin, Simon. Neural networks and learning machines. 3rd ed. Pearson Education, Inc., 2009.p (21-24)

[43]https://www.researchgate.net/figure/Single-layer-feedforward-neural-network_fig9_301344011 access 3/5/2023 access 21/6/2023

[44] https://www.mdpi.com/2504-2289/5/4/62 access 3/5/2023 access 21/6/2023

[45]https://www.researchgate.net/figure/A-recurrent-network-with-no-self-feedback-loops-and-no-hidden-neurons-is-presented-1_fig7_246626189 access 21/6/2023

[46] https://www.researchgate.net/figure/A-recurrent-network-with-self-feedback-loops-and-hidden-neurons-is-presented-1_fig8_246626189 access 21/6/2023

[47] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[48]https://penseeartificielle.fr/comprendre-lstm-gru-fonctionnement-schema/ access 21/6/2023

[49] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 5998-6008.

[50] https://blog.dataiku.com/decoding-nlp-attention-mechanisms-to-understand-transformer-models encoder w decoder zro9 access 21/6/2023

[51] https://www.researchgate.net/figure/Transformer-model-architecture-this-figures-left-and-right-halves-sketch-how-the_fig1_357410305 access 21/6/2023

[52] https://ledatascientist.com/a-la-decouverte-du-transformer/ access 21/6/2023

[53] https://ledatascientist.com/a-la-decouverte-du-transformer/ access 21/6/2023

[54] https://torch.classcat.com/category/nlp-2/page/2/ access 21/6/2023

[55]https://www.reddit.com/r/MachineLearning/comments/xv9sni/d_confused_about_selfattention_mechanism/ access 15/5/2023

[56] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems (NeurIPS) (pp. 8024-8035). Retrieved from https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf access 05/07/2023

[57] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.

[58] Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2020). Attention Based Hierarchical LSTM Model for Sentiment Analysis. In Proceedings of the 2020 4th International Conference on Algorithms, Computing and Systems (pp. 263-267).

[59] Sun, C., Qiu, X., Xu, Y., Huang, X., & Xiong, C. (2020). MobileBERT: A Compact Task-Agnostic BERT for Resource-Limited Devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 6063-6068).

[60] Huang, L., & Carley, K. M. (2021). Masked Emotional Language Model for Sentiment Analysis. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) (pp. 1410-1415).

[61] https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3 access 20/6/2023

[62] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-LT, (Mlm), 2018.

[63] https://www.kaggle.com/datasets/mksaad/arabic-sentiment-twitter-corpus access 26/6/2023.

ANNEX:
TOOLS AND IMPLEMENTATION

1 Introduction

In this annex, we will introduce the working environment, the programming language, and the tools we used to build the system.

2 Environment and development tools

In this section, we will discuss the development environment, programming language, and tools we utilized to build the system. Additionally, we will explain the experiments conducted on the proposed method and the resulting outcomes.

To develop our system and validate our approach, we utilized the Python programming language and the Google Colab environment for writing our programs. For deep learning tasks, we employed PyTorch, a popular library developed by Google. We also utilized the NumPy library for efficient matrix manipulation.

By leveraging these tools, we were able to effectively implement and evaluate our method. Python served as the programming language, Google Colab provided a convenient and collaborative environment, PyTorch enabled us to perform deep learning tasks, and NumPy facilitated efficient handling of matrices.

2.1 Development environment

Python is a high-level programming language created by Guido van Rossum and released in 1991. Python features a portable, dynamic, extensible, free, and user-friendly type system. Its syntax is simpler and results in shorter code compared to languages like C or Java. Python supports multithreading, object-oriented programming, and scalability [37].

Figure 1 Python logo [41].

2.2 Google Colab

Google Colaboratory, often shortened to "Colab," is a cloud-based service provided by Google (free of charge). It is based on Jupyter Notebook and is designed for education and research in machine learning. This platform allows for training machine learning models directly in the cloud, eliminating the need to install any software on your computer other than a browser. With Colab, you can write and execute Python code in your browser without any required configuration. It provides free access to GPUs, making it convenient for computationally intensive tasks. Additionally, Colab allows for easy sharing of notebooks, facilitating collaboration and knowledge dissemination [41].

Figure 2 Google Colab logo [41].

2.3 NumPy

NumPy is a library for the Python programming language that adds support for large, multi-dimensional arrays and matrices, along with a vast collection of high-level mathematical functions to operate on these arrays. The predecessor of NumPy, called Numeric, was originally created by Jim Hugunin, with contributions from several other developers.

In 2005, Travis Oliphant created NumPy by incorporating the features of Numarray into Numeric, along with numerous enhancements. NumPy is an open-source software project with a large community of contributors, continuously improving and expanding its capabilities [41].

Figure 3 NumPy logo [41].

2.4 PyTorch

PyTorch is an open-source deep learning framework that provides a flexible and efficient platform for building and training neural networks. It offers dynamic computation graphs, allowing for easy model customization and debugging. PyTorch is widely used in the research and development of various machine learning applications, including computer vision, natural language processing, and reinforcement learning [41].

Figure 4 PyTorch logo [61].

3 Implementation

These are parts from our code

3.1 Load the data

```
# Load and preprocess data
column_names = ["sentiment", "content"]
train_tweets_negative = pd.read_csv('/content/drive/MyDrive/train_Arabic_tweets_negative_20190413.tsv', sep='\t', names=column_names)
train_tweets_positive = pd.read_csv('/content/drive/MyDrive/train_Arabic_tweets_positive_20190413.tsv', sep='\t', names=column_names)
test_tweets_positive = pd.read_table("/content/drive/MyDrive/test_Arabic_tweets_positive_20190413.tsv", sep='\t', names=column_names)
test_tweets_negative = pd.read_table("/content/drive/MyDrive/test_Arabic_tweets_negative_20190413.tsv", sep='\t', names=column_names)
```

Figure 5 Load the data.

In this part of the code, we load and preprocess the data. The data consists of Arabic tweets that are labeled as either positive or negative sentiment. We use the Pandas library to read the data from separate files and store it in dataframes. Each dataframe contains two columns: "sentiment" and "content". The "sentiment" column represents the sentiment label, while the "content" column contains the actual text of the tweets. This preprocessing step is essential for further analysis and modeling of the sentiment in Arabic tweets.

3.2 Combine the data

```
# Combine positive and negative training tweets into a single dataset
X_train = pd.concat([train_tweets_negative, train_tweets_positive])
X_test = pd.concat([test_tweets_negative, test_tweets_positive])
display(X_train)
display(X_test)
```

Figure 6 Combine the data.

In this part of the code, we combine the positive and negative training tweets into a single dataset for further processing. The “pd.concat()” function from the Pandas library is used to concatenate the dataframes “train_tweets_negative” and “train_tweets_positive”, resulting in the combined dataset “X_train”. Similarly, we concatenate the dataframes “test_tweets_negative” and “test_tweets_positive” to create the dataset “X_test”. This step is performed to have a unified dataset containing both positive and negative sentiment tweets, which will be used for training and testing our sentiment analysis model.

3.3 Applying the preprocessing

```
# Apply preprocessing to train and test datasets
X_train['content'] = X_train['content'].apply(lambda x: remove_stop_words(x))
X_train['content'] = X_train['content'].apply(lambda x: remove_emojis(x))
X_train['content'] = X_train['content'].apply(lambda x: decode_emojis(x))
X_train['content'] = X_train['content'].apply(lambda x: remove_punctuation(x))
X_train['content'] = X_train['content'].apply(lambda x: remove_diactrics(x))
X_train['content'] = X_train['content'].apply(lambda x: translate_emojis_to_arabic(x))

X_test['content'] = X_test['content'].apply(lambda x: remove_stop_words(x))
X_test['content'] = X_test['content'].apply(lambda x: remove_emojis(x))
X_test['content'] = X_test['content'].apply(lambda x: decode_emojis(x))
X_test['content'] = X_test['content'].apply(lambda x: remove_punctuation(x))
X_test['content'] = X_test['content'].apply(lambda x: remove_diactrics(x))
X_test['content'] = X_test['content'].apply(lambda x: translate_emojis_to_arabic(x))
```

Figure 7 Preprocessing the data.

In this code, we apply various preprocessing steps to the train and test datasets to clean and normalize the text data before feeding it to the sentiment analysis model. The following preprocessing steps are performed:

- Removing stop words: The “remove_stop_words()” function is applied to remove common words that do not contribute much to the sentiment analysis task.
- Removing emojis: The “remove_emojis()” function is used to eliminate emojis from the text as they do not carry significant sentiment information.
- Decoding emojis: The “decode_emojis()” function is applied to convert any encoded emojis to their corresponding textual representation.
- Removing punctuation: The “remove_punctuation()” function is used to remove punctuation marks from the text.
- Removing diacritics: The “remove_diactrics()” function is applied to remove diacritics, such as accents and tashkeel, from the text.
- Translating emojis to Arabic: The “translate_emojis_to_arabic()” function is used to translate any remaining emojis into their Arabic equivalents.

These preprocessing steps help in standardizing and cleaning the text data, ensuring that the sentiment analysis model receives consistent and meaningful input.

3.4 Define the custom

```
# Define dataset class
class ArabicSentimentDataset(Dataset):
    def __init__(self, tweets, tokenizer, max_length):
        self.tweets = tweets
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.tweets)

    def __getitem__(self, index):
        tweet = str(self.tweets.iloc[index]['content'])
        sentiment = self.tweets.iloc[index]['sentiment']

        encoding = self.tokenizer.encode_plus(
            tweet,
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_token_type_ids=False,
            return_attention_mask=True,
            return_tensors='pt'
        )

        input_ids = encoding['input_ids'].squeeze()
        attention_mask = encoding['attention_mask'].squeeze()

        return {
            'input_ids': input_ids,
            'attention_mask': attention_mask,
            'sentiment': torch.tensor(sentiment, dtype=torch.long)
        }
```

Figure 8 Custom dataset.

In this code, we define a custom dataset class called “ArabicSentimentDataset” for handling the sentiment analysis data.

- “__init__()”: The constructor method initializes the dataset with the provided tweets, tokenizer, and maximum length. It stores these values as attributes of the dataset.
- “__len__()”: This method returns the total number of samples in the dataset, which is determined by the length of the tweets attribute.
- “__getitem__()”: This method retrieves a specific sample from the dataset given its index.

By implementing this dataset class, we can easily create instances of the dataset and pass them to the data loader for training the sentiment analysis model. The dataset takes care of tokenizing the text data and preparing it in the appropriate format for the model's input.

3.5 Model and tokenizer

```
# Define model and tokenizer
model_name = 'aubmindlab/bert-base-arabertv02'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2).to(device)
```

Figure 9 Model and tokenizer.

In this code, we define the BERT model and tokenizer for sentiment analysis in Arabic. By utilizing the pre-trained BERT model and tokenizer, we can benefit from the powerful language representation capabilities of BERT for sentiment analysis.

3.6 downloading

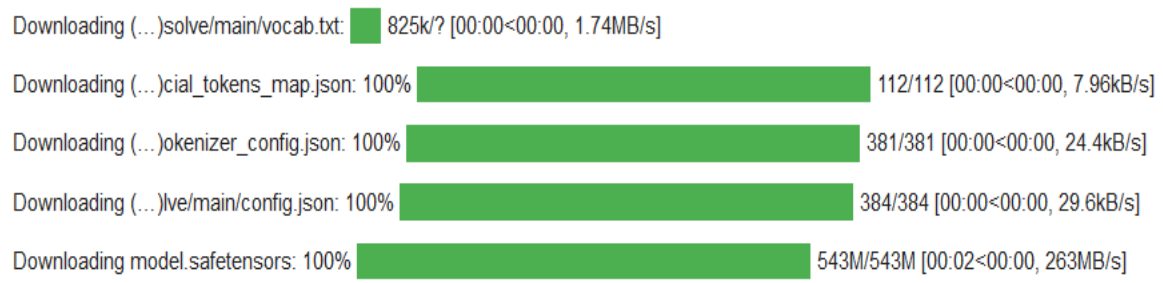

Figure 10 Success downloading.

And this represents the success of the previous part.

3.7 Incorrect and correct predictions

```
incorrect_index = None

for i in range(len(true_sentiments)):
    if true_sentiments[i] != predicted_sentiments[i]:
        incorrect_index = i
        break

if incorrect_index is not None:
    print("Incorrectly Predicted Tweet:")
    print("Content:", X_valid.iloc[incorrect_index]['content'])
    print("True Label:", true_sentiments[incorrect_index])
    print("Predicted Label:", predicted_sentiments[incorrect_index])
else:
    print("No incorrectly predicted tweets found.")
```

Figure 11 Incorrect predicted tweets.

```
correct_index = None

for i in range(len(true_sentiments)):
    if true_sentiments[i] == predicted_sentiments[i]:
        correct_index = i
        break

if correct_index is not None:
    print("Correctly Predicted Tweet:")
    print("Content:", X_valid.iloc[correct_index]['content'])
    print("True Label:", true_sentiments[correct_index])
    print("Predicted Label:", predicted_sentiments[correct_index])
else:
    print("No correctly predicted tweets found.")
```

Figure 12 Correct predicted tweets.

These steps allow us to visually analyze the model's performance using the confusion matrix and gain insights into specific instances where the model made correct and incorrect predictions by examining the content of the tweets.

Conclusion

In conclusion, this annex provided an overview of the working environment, programming language, and tools used in the development of our system. We have established a solid foundation for building our system by selecting appropriate tools and technologies that align with our project requirements.