People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

UNIVERSITY OF SAAD DAHLEB BLIDA

Faculty of sciences

Department of computer science

Graduation Project

Report submitted for the fulfillment of the master degree in Computer Science

Option: Software Engineering

Realized by**:** CHITA Ramzi & BOUCHELARAM Ishak

Project

ENCODER-DECODER NEURAL NETWORK ARCHITECTURES FOR AUTOMATIC AUDIO CAPTIONING

On 25/9/2022 before the jury composed of:

- **Supervisor**: Mr A. KAMECHE, Computer Science Department, Blida1.
- **President :** Mme BERRAMDANE, Computer Science Département, Blida1.
- **Examiner :** Mme GUESSOUM, Computer Science Département, Blida1.

Academic year: 2021/2022

ABSTRACT

The main purpose of this project is to design an environmental general audio content description using text, where a system accepts as an input an audio signal and outputs the textual description of that signal.

This task has drawn lots of attention during the past several years as a result of quick devolvement of different methods that can provide captions for a general audio recording. To accomplish the automatic audio captioning task, we have performed multiple experiments using a Clotho dataset. Two deep neural networks have been employed in the construction of our systems Recurrent Neural Network and Gated Recurrent Unit, along with encoder-decoder architecture and a combination of feature representations based on audio processing techniques like Mel Spectrogram and text processing techniques used in text decoding from word embeddings like one-hot-encoding and BERT.

Keywords: Audio Captioning, Machine Learning, Encoder Decoder Models, Signal Processing, Natural Language Processing.

RÉSUMÉ

L'objectif principal de ce projet est de concevoir une description de contenu audio général environnemental à l'aide de texte, où un système accepte en entrée un signal audio et produit la description textuelle de ce signal.

Cette tâche a attiré beaucoup d'attention au cours des dernières années en raison de l'évolution rapide des différentes méthodes qui peuvent fournir des sous-titres pour un enregistrement audio général. Pour accomplir la tâche de sous-titrage audio automatique, nous avons effectué plusieurs expériences à l'aide d'un ensemble de données Clotho. Deux réseaux de neurones profonds ont été utilisés dans la construction de nos systèmes Recurrent Neural Network et Gated Recurrent Unit, ainsi qu'une architecture d'encodeur-décodeur et une combinaison de représentations de caractéristiques basées sur des techniques de traitement audio telles que Mel Spectrogram et des techniques de traitement de texte utilisées dans le décodage de texte à partir des incorporations de mots comme l'encodage à chaud et le BERT.

Mots-clés : Traduction vocale, Apprentissage Automatique, modèles de décodage, traitement du signal, Traitement du langage naturel

ملخص

الغرض الرئيسي من هذا المشروع هو تصميم وصف بيئي عام للمحتوى الصوتي باستخدام النص الحر، حيث يقبل النظام كمدخل إشارة صوتية ويخرج الوصف النصي لتلك الإشارة.

جذبت هذه المهمة الكثير من الاهتمام خلال السنوات العديدة الماضية نتيجة للانتقال السريع للطرق المختلفة التي يمكن أن توفر تسميات توضيحية لتسجيل صوتي عام. لإنجاز مهمة التسميات التوضيحية الصوتية التلقائية، أجرينا تجارب متعددة باستخدام مجموعة بيانات كلوتو. تم استخدام شبكتين عصبيتين عميقتين في بناء أنظمتنا الشبكة العصبية المتكررة والوحدة المتكررة ذات البوابات، جنبًا إلى جنب مع بنية وحدة فك التشفير ومجموعة من تمثيلات الميزات القائمة على تقنيات معالجة الصوت مثل Mel Spectrogram وتقنيات معالجة النصوص المستخدمة في فك تشفير النص من حفلات الزفاف كلمة مثل BERT او ONE-HOT-ENCODING.

الكلمات الرئيسية: الترجمة الصوتية، التعلم الآلي، نماذج فك التشفير، معالجة الإشارات، .
معالجة اللغة الطبيعية

Acknowledgement

First, we thank ALLAH Sobhanou for giving us the will and courage to undertake and complete this work. We would like to express our deepest and most sincere gratitude to our promoter and coach ABDALLAH HICHAM KAMECHE, ASSISTANT PROFESSOR B, University of Blida, for guiding and enriching our work. We thank him for his availability, his valuable advice, his confidence despite our rather light knowledge in the field of audio processing. We also thank him for his attention to detail and for giving us the opportunity to conduct this research and providing us invaluable guidance throughout our work that led to this work been completed.

Our thanks also go to the members of the jury for agreeing to examine our work and enrich it with their proposals. We would also like to thank the faculty and administrative staff at BLIDA University who have contributed to the success of our university studies.

Finally, many thanks go to all the people who have contributed to the completion of our research work directly or indirectly.

Thanks again.

Contents

List of acronyms and abbreviations

AAC	Automated audio captioning
ADC	Analogue-to-digital-Converter
AE	Amplitude Enveloppe
AI	Artificiel intelligence
BERT	Bidirectionnel Encoder Représentations frome Transformers
CNN	Convolution neural network
DAC	digital-to-analogue-Converter
DFT	Discreet Fourier transform
DL	DEEP Learning
FN	False Negative
FP	False Positive
FT	Fourier transform
GRU	Gated Recurrent Units
KNN	k-nearest neighbors
MIR	Music Information Retrieval
MLP	multiple layers prediction
NL	natural language
NLP	natural language processing
NN	Neural network
RELU	Rectified Linear Units
RMS	Root-mean-square
RMSE	Root-Mean-Square-Energy
RNN	recurrent Neural Networks
STFT	short-time Fourier transform
SVM	support Vector Machine
TDSP	Team Data Science Process
TN	True Negative
TP	True Positive
ZCR	The Zero-Crossing Rate

List of figures

List of Tables

Introduction

In this introduction, we aim to present the tasks that our dissertation aims to address and the broader field related to them. Firstly, we discuss how linguistics, Natural Language Processing and Artificial Intelligence are intertwined in our view. We present our motivation and our main research topic. Lastly, we present our main contributions to the field and memory organization.

1. AI and Language

Language is on each own a very complex system. According to Descartes, language is power only we humans possess, a power that sets us apart, in a qualitative, unbridgeable way from everything else there is, notably from animals and machines. in 1950, Alan Turing wrote a paper describing a test for a “thinking” machine. He argued that if a machine could have a conversation through the use of a teleprinter, and it imitated a human without noticeable differences then the machine could be considered capable of thinking. Many different paradigms have been proposed in the field of linguistics to approach a broader understanding of language. In the early 1900s, a Swiss linguistics professor named Ferdinand de Saussure aimed to attack the concept of language as a product of human speech, describing languages as "systems of difference". He argued that words are just acoustic images unhinged on themself of any particular meaning. Recent progress in Artificial Intelligence and Natural Language Processing has produced models that perform surprisingly well at generating text, textual descriptions of images, answering questions, summarizing large documents, etc.

Nonetheless, we are fascinated by AI systems. Their capability of generating natural language shifts our understanding of reality and how we experience the world. Besides language their other modalities that play a crucial role in the human understanding of the world such as images and sounds. We are interested in the ways these modalities are intertwined with language and how machines are able to mutually process and understand them.

In this dissertation, we research a multi-modal task called Automated Audio Captioning. It can be viewed as a cross-modal translation task that aims to generate

natural language descriptions of sound and audio events. It is a task that has received increasing attention in recent years, but is still largely unexplored, compared to other multimodal tasks such as Image Captioning.

2. Motivations

Artificial intelligence with its popularization and gradual emergence as a core technology that drives tremendous developments in many fields, the use of machine learning and deep learning has grown tremendously[1]. According to numerous surveys and studies, AI and machine learning are expected to be among the best rewarding and most lucrative career paths in the coming years.

Like all other areas that are constantly developing and preparing for the future, the area of Automated audio captioning also benefits from deep learning. Similar to image captioning, audio captioning is mostly based on an encoder-decoder architecture. For Automated audio captioning who rely on general audio content description using free text, deep learning can be a solution for exploring and developing different methods that can provide some kind of captions for a general audio recording.

3. Problems

Automated audio captioning (AAC) is an inter-modal task describing an audio signal using textual descriptions (referred to as captions). Like other natural language-related tasks, an appropriate caption should match the contents present in the audio and aligns with the descriptions provided by a human. The example captions are "many birds are chirping in the trees as cars drive by" or "many birds are chirping in the trees as cars drive by". AAC does not simply detect and classify sounds but explores the inner relationships between events and associates them with high-level concepts and information. Being an inter-modal task, AAC is exposed to the challenges related to audio and natural language processing (NLP).

- First, an event may sound significantly in different environments, devices, and settings.
- Second, real-life audios usually contain mixtures of overlapped sounds.

- Third, the language model needs to be sufficiently good to generate close-to-human created captions.

Finally, the language model plays an important role during the caption generation phase. The modelled language needs to describe relationships between sound events (e.g., "A jet sound roars continuously and then gets a bit louder"), discriminate source locations (e.g., "Birds chirping outside while people are talking in the background"), the properties of sounds and environments (e.g., "on a hard surface", "a wooden cutting board") and sound characteristics (e.g., "a loud banging of a metal material"). This requires building a diverse vocabulary set that the language model can utilize to formulate descriptive texts.

4. Contributions

Automated audio captioning is a new and challenging task that involves different modalities. It could be described as generating a textual description given an audio signal, where the caption should be as close as possible to a human-assigned one. In contrast to automatic speech recognition which just converts speech to text, AAC converts environmental sound to text[2]. It is also different from sound event detection and audio tagging tasks, which output exact labels with start and end time or not. Generating accurate captions needs more information, including identification of sound events, acoustic scenes, foreground versus background discrimination, concepts, and physical properties of objects and environment. This report proposes an audio captioning system for the Detection and Classification of Acoustic Scenes and events (DCASE) 2021 challenge task. Our audio captioning system consists of a Gated Recurrent Units audio encoder and a Recurrent Neural Networks text decoder.

The main goal of this work is to design an audio captioning system trained on the newly publicly available dataset Clotho v2T[3]. The proper use of such information can considerably improve the captioning performance. The second goal would be to test the impact of various semantic embeddings; and to perform a comparative study among several encoder-decoder architectures.

Our main contributions are the following:

• We propose a novel architecture for Automated Audio Captioning utilizing state-of-the-art strategies to reduce the computational complexity of our model.

• We investigate the quality vs diversity trade of language generation in our proposed model.

• We propose an application of Automated Audio Captioning in generating textual descriptions of sound events in Audios.

• We propose a novel evaluation metric in order to evaluate the performance of our system.

5. Memory organization

a. Chapter 1

In the 1rst part we delve into Artificial Intelligence in Machine Learning and we explained some of the most frequently used methods of classification.

In The 2rd part is devoted to deep learning, we explain what a Neural network is and how it works and lastly, we dive into some of Neural network architectures that we have used in our encoder-decoder model.

b. Chapter 2

In the 1rst part we provide an overview of Audio Processing and deferent concepts that are relevant for audio feature extraction.

In the 2nd part we present text processing. We provide a description for the tools and setup that we frequently used for text feature extraction, then we present and discuss the deferent methods used in Natural Language Processing

c. Chapter 3

This chapter is devoted to the presentation of the experimental results as well as their interpretations we showed the deferent models we used as well as their architecture.

d. Chapter 4

In this chapter we present the results. We provide a description for the tools and setup that we used in our project, then we end it with desktop interface representation and future work.

Chapter 1: Machine learning and Deep learning

Introduction

As mentioned in the Introduction, in this chapter we present our theoretical background, regarding Machine Learning (ML) literature and especially the Deep Learning (DL) subfield.

Part 1: Machine learning

1. Introduction

Machine learning has enjoyed a diverse history finding its roots in many interdisciplinary fields including artificial intelligence, neuroscience, cognitive science and various other areas as it eventually connected more closely with the field statistics. As early as 1921, when Capek coined the term Robot [4], the idea that a machine could be intelligent and potentially learn from observations began emerging.

Machine learning has shown great success in building models for pattern recognition in domains ranging from computer vision over speech recognition and text understanding to Game AI[5]. It's an artificial intelligence area that assists computers in estimating future events and modelling based on experiences gained from previous information. Compared to the classical methods, the process of obtaining information is much more accurate and faster[6].

2. Machine learning tasks

Machine learning takes different forms, depending on the algorithm and its objectives. You can divide machine learning algorithms into three main groups based on their purpose as showed in the figure 1 [7].

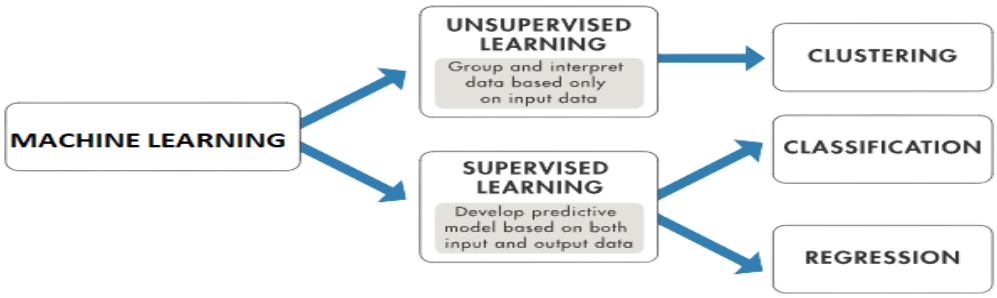

Figure 1: Machine Learning algorithms[7]

In machine learning we have multiple methods of learning and showing a computer how to make deferent predictions with deferent cases, which we will all explain in the next parts[7].

3. Supervised learning

Supervised learning algorithms take direct feedback for the prediction. the machine already knows the answers that are expected of it[8]. It works from labelled data. Supervised learning can be categorized in classification and regression methods. In supervised learning: the goal is to use input-label pairs, (x; y) to learn a function *F* that predicts a label given the input, ^y = f (x).

KNN, SVM [9], are some popular algorithms of supervised learning. This next figure 2 shows an example of supervised learning algorithm.

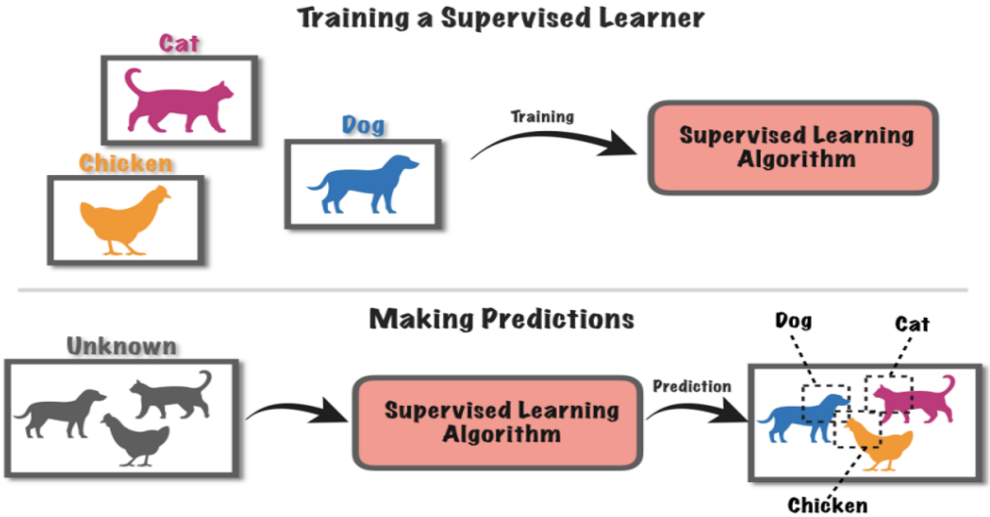

Figure 2: Supervised Learning Example[10]

A supervised learning algorithm always has a target or outcome variable, which is detected from a provided set of predictors. The algorithm uses this set of variables to create a function that maps inputs to desired outputs. This training process is repeated for as long as it takes for the model to achieve a high level of accuracy this figure 3 illustrates how a supervised learning algorithm works.

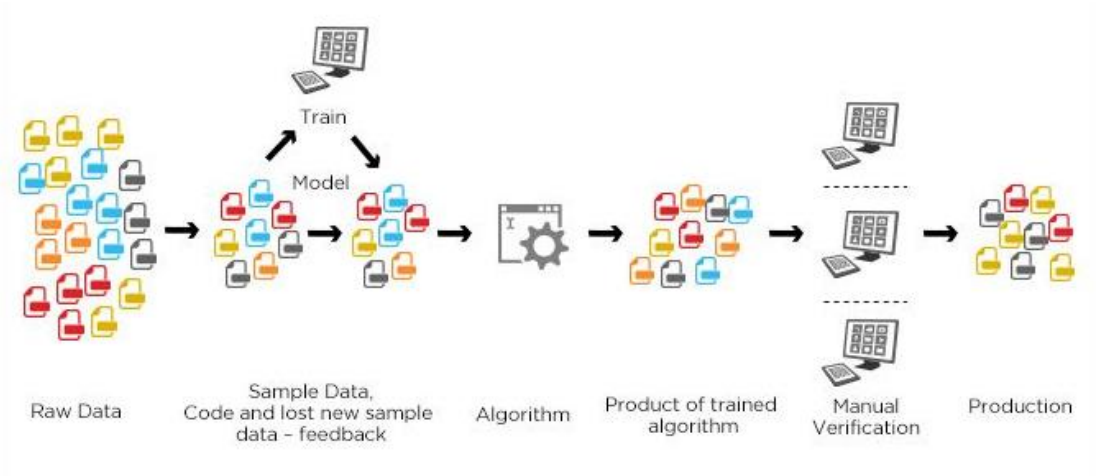

Figure 3: How Supervised Learning Works[11]

3.1. Classification

These tasks consist of assigning a class to objects[6]. Classification algorithms utilize input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories as shown in the figure 4.

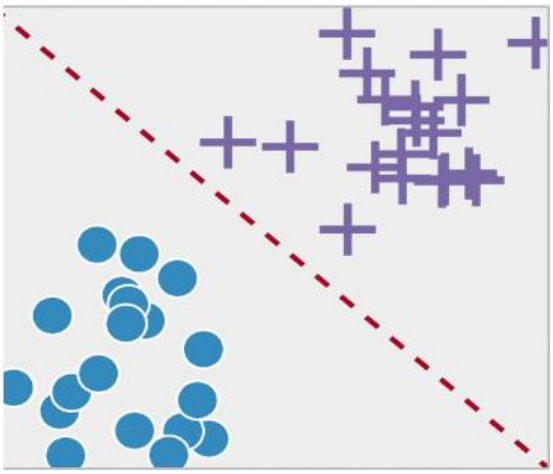

Figure 4: Classification Example[12]

After the system has been trained to identify a category of an input X, when new inputs are added, it'll automatically be classified in their categories. Classification problems, requires items to be divided into different categories, based on past data.

Classification can be multi label as well as shown in this next figure 5.

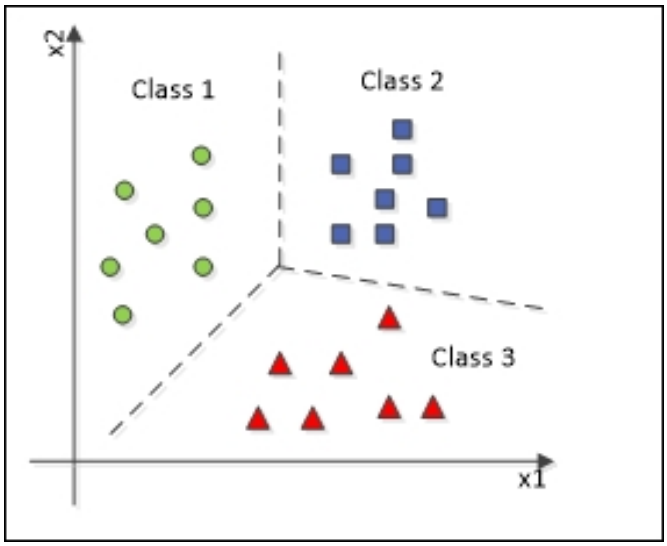

Figure 5: multi label classification[13]

3.2.Regression

In this case, we are not assigning a class but a mathematical value: a percentage or an absolute value. Regression is a process of finding the correlations between dependent and independent variables. Now with regression problem, the system attempts to predict a value for an input based on past data. Unlike classification, we are predicting a value based on past data, rather than classifying them into different categories. This figure 6 shows how in regression we don't have multiple categories classes.

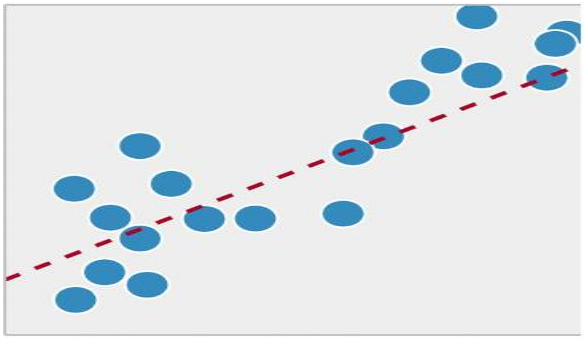

Figure 6: Regression example[14]

The task of the Regression algorithm is to find the mapping function to map the input variable(x) to the continuous output variable(y).

4. Unsupervised learning

In unsupervised learning is that the answers we are trying to predict are not available in the datasets. The algorithm uses an unlabeled dataset[15]. The machine is then asked to create its own responses. the algorithm seeks to maximize on the one hand the homogeneity of the data within the groups of data and to form groups as distinct as possible. In unsupervised learning no label or another target is provided. The data consists of a set of examples x and the objective is to learn about the statistical structure of x itself.

4.1. Clustering

The machine group objects into data sets that are as homogeneous as possible[16]. This technique may seem close to that of classification in supervised learning, but unlike the latter, the classes are not pre-filled by a human, it is the machine that creates its own classes.

From this figure 7 we can see how the machine makes prediction on what object to be grouped to gather given the same labels.

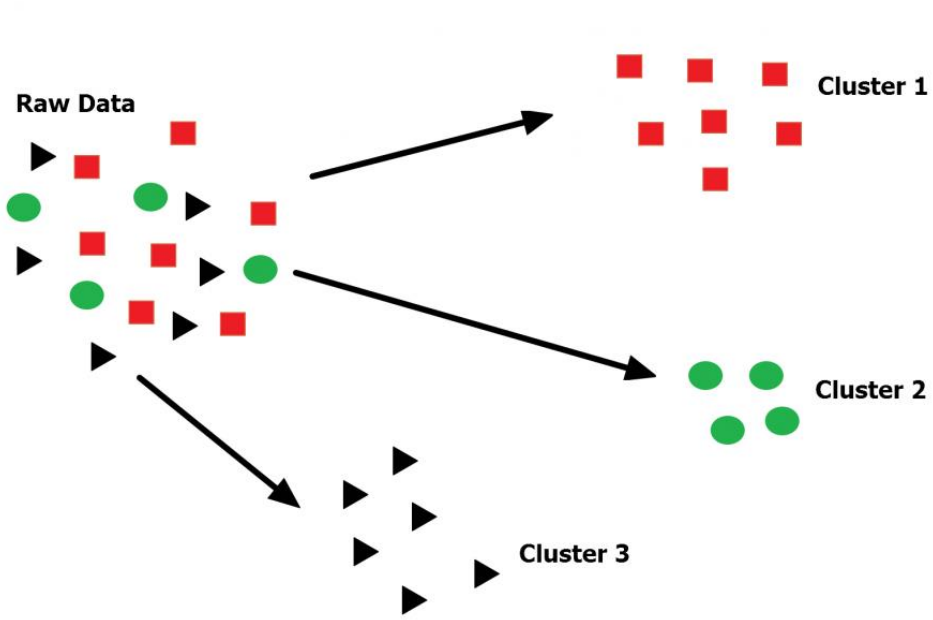

Figure 7: Clustering exemple[17], [18]

5. Model evaluation

To be able to estimate the performance of a machine learning model First, we feed the training data to our learning algorithm to learn a model. Second, we predict the labels of our test set. Third, we count the number of wrong predictions on the test dataset to compute the model's prediction accuracy[21]. The evaluation metric is a crucial element in achieving the optimal classifier during the training process. Thus, a selection of a suitable evaluation metric is an important key for discriminating and obtaining the optimal classifier.

For classification problems, the evaluation of the optimal solution during the training stage can be defined based on confusion matrix.

5.1. Confusion matrix

A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem[22], it helps visualize its outcomes. As shown in the table 1 The row of the table represents the predicted class, while the column represents the true class.

From this confusion matrix, TP and TN denote the number of positive and negative instances that are correctly classified. Meanwhile, FP and FN denote the number of misclassified negative and positive instances, respectively. It plots a table of all the predicted and actual values of a classifier.

	Predicted: NO	**Predicted: YES**
Actual: NO	**TN**	**FP**
Actual: YES	**FN**	**TP**

Table 1 - Confusion matrix [18]

- True Positive: The Numbers of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- False Positive: The Numbers of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- True Negative: The Numbers of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.
- False Negative: The Numbers of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

From this table, several measures can be derived to assess the performance of the classifier with different evaluation objectives, as shown in Table 2.

Metrics	**Formula**
Accuracy (ac)	$\frac{tp + tn}{tp + fp + tn + fn}$
Sensitivity (se)	$\frac{tp}{tp + fn}$

Specificity (SP)	$\frac{tn}{tn + fp}$
Error Rate (err)	$\frac{fp + fn}{tp + fp + tn + fn}$
Precision (p)	$\frac{tp}{tp + fp}$
Recall (r)	$\frac{tp}{tp + tn}$
F-Measure (FM)	$2 * \frac{p * r}{p + r}$

Table 2- Evaluation metrics for Classification [18]

5.2.Bias

The term bias refers to the statistical bias. In general terms, the bias of an estimator ˆβ is the difference between its expected value E[ˆβ] and the true value of a parameter β being estimated, we compute the prediction bias as the difference between the expected prediction accuracy of a model and its true prediction accuracy[23].

5.3.Variance

The variance is a measure of the variability of a model's predictions if we repeat the learning process multiple times with small fluctuations in the training set. The more sensitive the model-building process is towards these fluctuations, the higher the variance.

5.4.Overfit and Underfit

A key balancing act in machine learning is choosing an appropriate level of model complexity, if the model is too complex, it will fit the data used to construct the model very well but generalize poorly to unseen data there for the term Overfit.

If the complexity is too low the model won't capture all the information in the data there for the term underfitting. since a complex model exhibits large variance while an overly simple one is strongly biased[24]. Most general-purpose methods feature hyperparameters to control this trade-off.

Part 2: Deep learning

In this part, we discuss Deep Learning approaches to Natural Language Processing and Audio Signal Processing and focus on models such as the Transformer which is the backbone of our proposed approach, and its usage in sequence-to-sequence modelling.

1. Introduction

Since 2012, deep neural networks have revolutionized machine learning. Although relatively old, this technique has made very significant progress in recent years, especially for the recognition of texts, sounds, images and videos. Understand the issues of these methods raises questions at the interface between mathematics and algorithms.

In this part, we will explain the structure of these networks as well as the key concepts of their learning.

2. Neural network

An artificial neural network is built around a biological metaphor. We know relatively well the structure of the primary visual cortex. Thus, in an extremely simplified view of the functioning of the brain, neurons are organized in layers, each neuron retrieves information from a previous layer, performs a very simple calculation, and communicates its result to neurons in the next layer.

Figure 8 details an example of such an artificial network. This type of neuron was introduced in 1943 by McCulloch and Pitts[26].

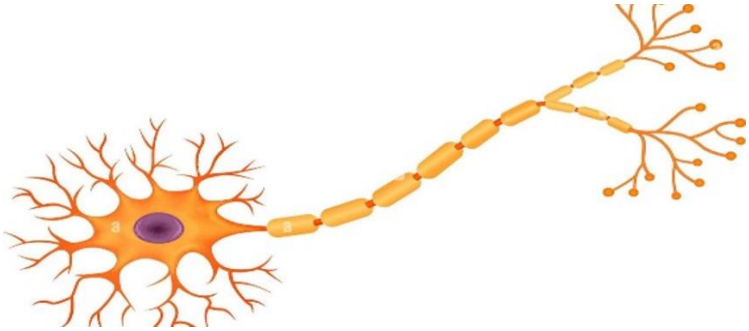

Figure 8: Neuron Anatomy

The transition from one layer to another is done through a set of artificial neurons. A neuron layers are represented in figure 9.

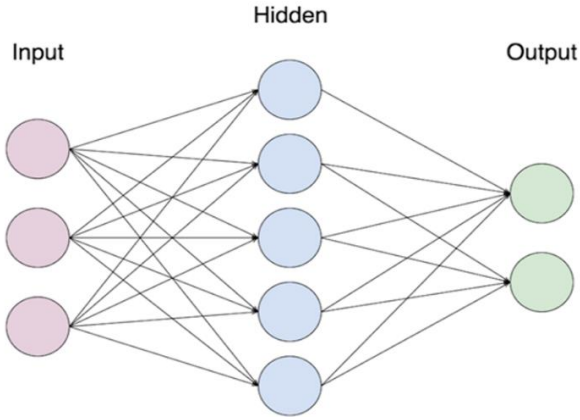

Figure 9: Neural network layers

The transition from one layer to another is done through a set of artificial neurons wish is represented in figure 10.

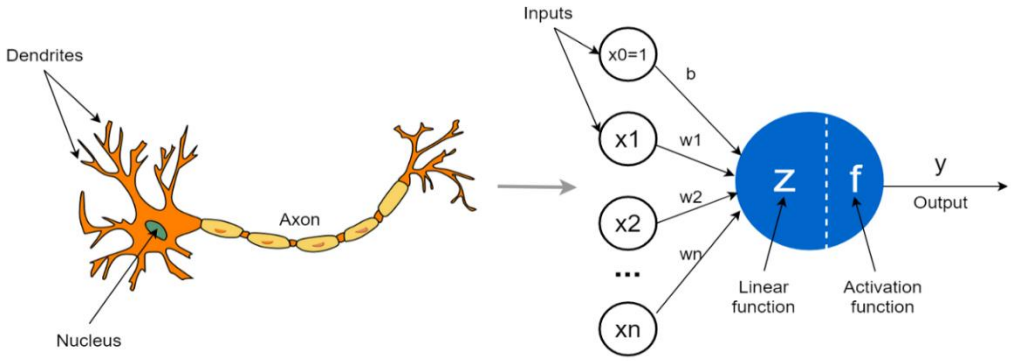

Figure 10: Artificial Neurons[27]

It's the first neuron, the one that calculates the first value that composes the layer. This neuron connects a number of elements from the first layer to a single element from the second[28]. The formula calculated by the neuron is:

$$y1 = \max\,(w1 * x1 + w2 * x2 + w3 * x3 + w4,0)$$

The neuron thus performs a weighted sum of the three inputs, with three weights w1, w2, w3, and we also add w4, which is a bias. Then the neuron calculates the maximum between this sum and zero. We can also use another activation function than the maximum function[28].

Thus, if the weighted sum w1x1 + w2x2 + w3x3 + w4 is less than 0, then the neuron returns the value y1 = 0, otherwise it returns the value of this sum and places it in u1. Several activation functions can be considered in the classification task. Table 3 shows the most commonly used activation functions.

Activation Function	Formula
Sigmoid	$\varphi(x) = \frac{1}{1+e^{-x}}$
Hyperbolic tangent	$\varphi(x) = \frac{e^{2x}-1}{e^{2x}+1}$
SoftMax	$\varphi(x)\frac{e^{xi}}{\sum j e^{xj}}$
Rectified Linear Unit	$\varphi(x) = \max\,(0,x)$

Table 3- Activation fonctions for classification[28]

2.1.Neural network layer

A neural network is made up of neurons that are organized in layers. There are three types of layers: an input layer, an output layer, and a hidden layer. In most cases, there will be multiple hidden layers in a neural network[28]. The neurons in the input layer receive the input objects. When a neuron is activated, it activates the neurons in the next layer. Every neuron in one layer passes an output to the neurons in the next layer. This output is defined by two factors: weight and bias.

The weight defines how important a particular input is to the next neuron, and it also directs the flow of values from input to output. In the figure 10, the black arrows represent weights. On the other hand, bias is an added constant value that defines how easy it is for a neuron to get fired.

2.2.Forward and backward propagation

The process of sending data from one layer to the next is called propagation. There are two types of propagation: forward propagation and backward propagation. In forward propagation the data moves from input to hidden layer to output. It ends in a prediction based on the input, which can be accurate or inaccurate. In backward propagation, a prediction from the output layer is back-tracked from the output to the input layer, which shows the error rate. This is then used to modify the weights and biases of each neuron, giving the neurons with a higher error rate and greater adjustment[29]. It is important to constantly readjust the weights to minimize errors and gain higher accuracy.

The key is to get started quickly and then adjust weights to optimize for more accurate outputs. adjusting the weight at the end of each batch (known as learning rate optimization), and also change how much influence the errors from the previous batches have on the current one (which is called the momentum). In addition, it's possible to use algorithms to tune the neural networks, including gradient descent or stochastic gradient descent, as well as, Adam. More about these algorithms later in the chapter.

3. Optimization algorithms

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. At the same time, every state-of-the-art deep learning library contains implementations of various algorithms to optimize gradient descent[30].

3.1.Dropout

One frequently used method of optimizing neural networks is called dropout. This reduces the problem of over-fitting, where statistical noise enters a neural network that is too large for a small data set[31].

Dropout works by randomly dropping out certain outputs from a layer, which makes the previous layer look like it has fewer neurons. This reduces noise and improves the accuracy of the neural network. The exact amount of dropout you need will vary based on the dataset and the architecture of the neural network.

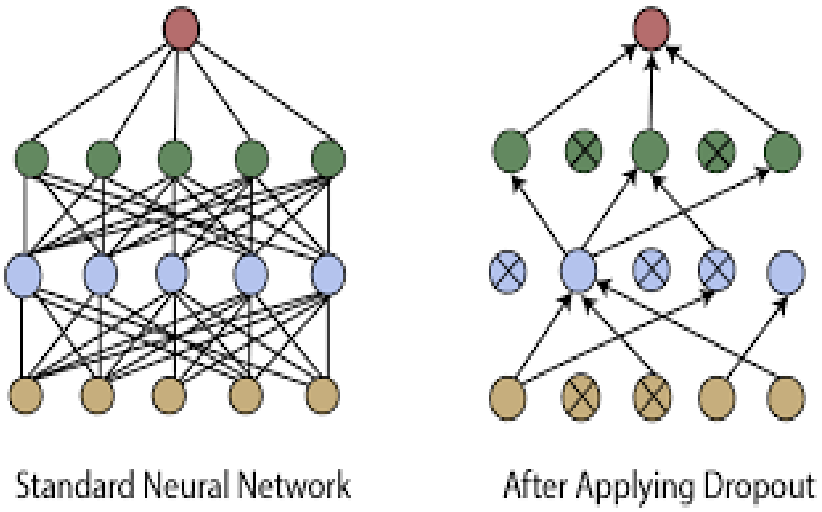

Figure 11: Dropout exemple

3.2.Adam

Adaptive Moment Estimation Adam is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients Adam also keeps an exponentially decaying average of past gradients. We compute the decaying averages of past and past squared gradients[32].

3.3.Learning rate scheduling

Learning rate schedules seek to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule. Common learning rate schedules include time-based decay, step decay and exponential decay[33].

So far, we focused on optimization algorithms for how to update the weight vectors rather than on the rate at which they are being updated. Nonetheless, adjusting the learning rate is often just as important as the actual algorithm. There are a number of aspects to consider:

- Most obviously the magnitude of the learning rate matters. If it is too large, optimization diverges, if it is too small, it takes too long to train or we end up with a suboptimal result.
- Secondly, the rate of decay is just as important. If the learning rate remains large, we may simply end up bouncing around the minimum and thus not reach optimality.
- Lastly it is equally important is initialization. This pertains both to how the parameters are set initially and also how they evolve initially.

4. Neural networks architectures

4.1. Recurrent neural networks

Deep learning models are built on the idea of neural networks, and this is what allows the models to learn from raw data where information is propagated forward. However, this 'feed-forward' type of model is not always applicable, and their fundamental architecture makes it difficult to apply them to sequential problem[2].

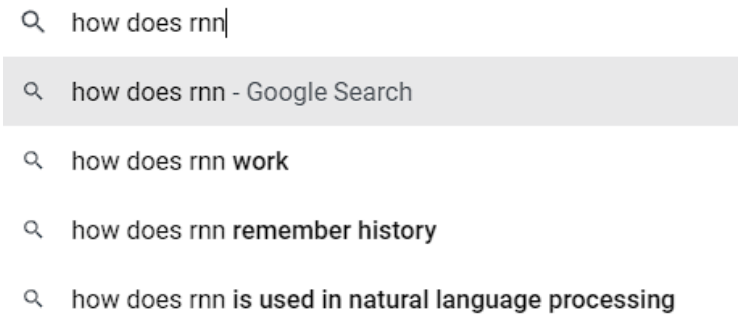

Figure 12: Search Engine

The figure 12 shows an example of sequential task where the most appropriate next word depends on the words which came before it.

The best way to overcome this problem is to have an entirely new network structure, one that can update information over time. This is a Recurrent Neural

Network. This is similar to a perceptron in that over time, information is being forward through the system by a set of inputs, x, and each input has a weight, w. Each corresponding input and weight are then multiplied, and the sum of products is calculated. The sum then passes through a non-linear activation function, and an output, y, is generated as shown in the figure 13.

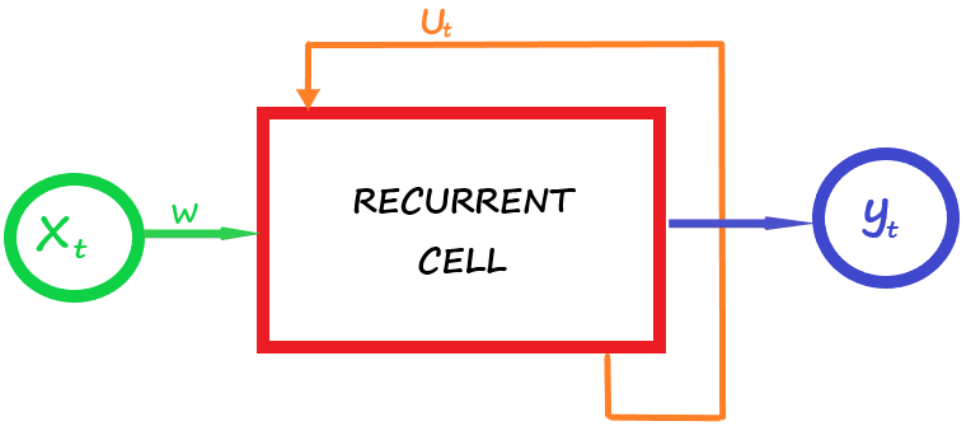

Figure 13: Recurrent Cell Architecture

The difference in this architecture is that, in addition to the output, the network is also generating an internal state update, U. This update is then used when analyzing the next set of input information and provides a different output that is also dependent on the previous information. This is ideal because information persists throughout the network over time. As the name suggests, this update function is essentially a recurrence relation that happens at every step of the sequential process, where u is a function of the previous u and the current input, x.

The RNN as a set of singular feed-forward models, where each model is linked together by the internal state update as shown in the figure 14. At each step of the sequence, there is an input, a process being performed on that input, and a related output. For the next step of the sequence, the step before must have some influence does not affect the input but affects the related output[36].

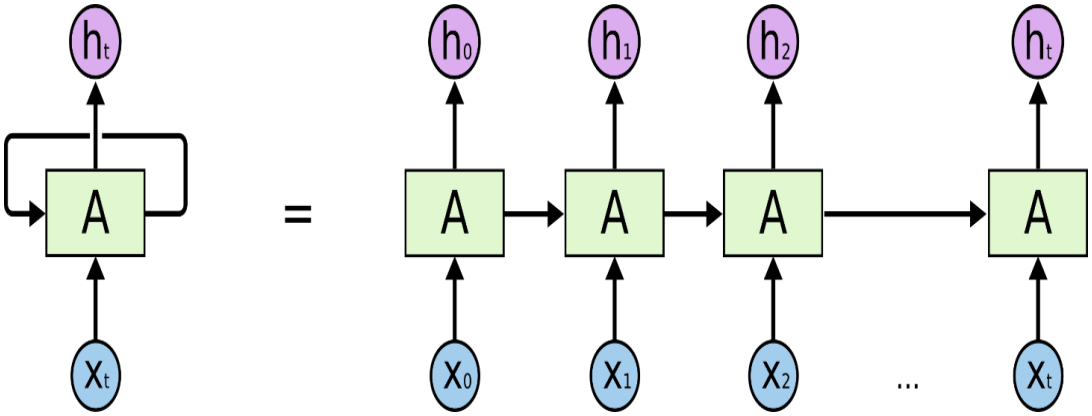

Figure 14: Different time steps of RNN[37]

As demonstrated in the figure 14 we input one example at a time and produce one result, both of which are single words. The difference with a feedforward network comes in the fact that we also need to be informed about the previous input words before evaluating the result[36].

Since plain text cannot be used in a neural network, we need to encode the words into vectors. The best approach is to use word embeddings word2vec or Glove for this examples we will go for the one-hot encoded vectors. These are (V,1) vectors V is the number of words in our vocabulary where all the values are 0, except the one at the i^{th} position[36].

Typically, the vocabulary contains all English words. That is why it is necessary to use word embeddings. This is the equations needed for training:

$$h_t = f(w^{hh} h_{t-1} + w^{hx} x_t)$$

- h_t holds information about the previous words in the sequenceh_t is calculated using the previous h_{t-1} vector and current word vectorx_t . We also apply a non-linear activation function f to the final summation.

$$y_t = softmax(w^s h_t)$$

- y_t calculates the predicted word vector at a given time step t. We use the SoftMax to produce a (V,1) vector with all elements summing up to 1. This probability distribution gives us the index of the most likely next word from the vocabulary.

$$j^t = \sum_{i=1}^{|v|} (y_{ti}' \, log \, y_{ti})$$

- j^t uses the cross-entropy loss function at each time step t to calculate the error between the predicted and actual word.
- w^x represent the weights of the network at a certain stage.

The weights are initialized with random elements, adjusted using the error from the loss function. We do this adjusting using back-propagation algorithm which updates the weights. Once we have obtained the correct weights, predicting the next word in the sentence is quite straightforward, this figure 15 represents a summary of what we just explain:

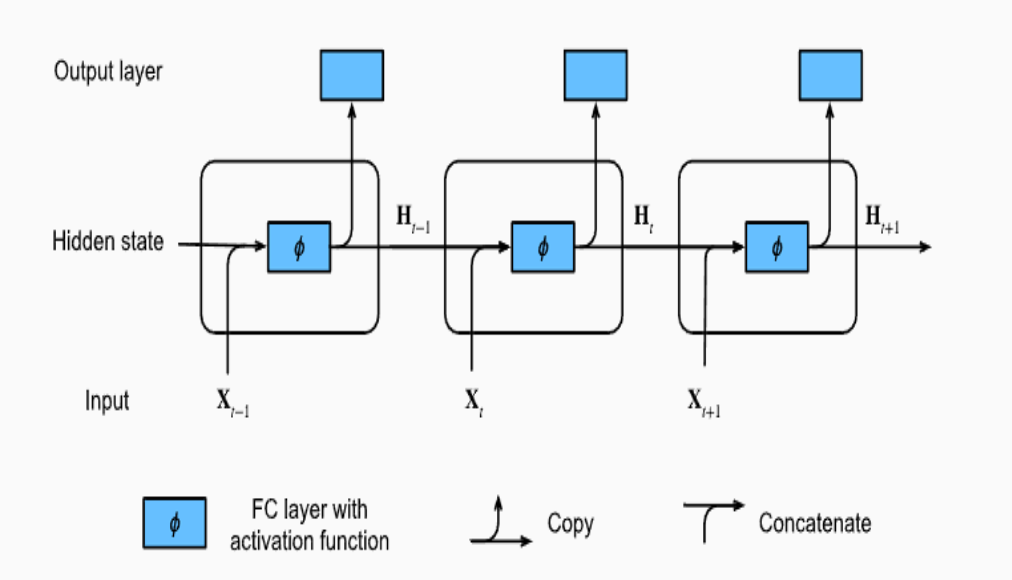

Figure 15: RNN model [36]

4.2. Gated Recurrent Units

GRUs are improved version of standard recurrent neural network. They were introduced to solve the vanishing gradient problem of a standard RNN, GRU uses update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction[38].

a. Gated Hidden State

The key distinction between vanilla RNNs and GRUs is that the latter support gating of the hidden state. This means that we have dedicated mechanisms for when a hidden state should be updated and also when it should be reset. For instance, if the first token is of great importance, we will learn not to update the hidden state after the first observation. Likewise, we will learn to skip irrelevant temporary observations. Last, we will learn to reset the latent state whenever needed. We discuss this in detail below.

b. Reset Gate and Update Gate

The first thing we need to introduce are the reset gate and the update gate, a reset gate would allow to control how much of the previous state we might still want to remember. Likewise, an update gate would allow to control how much of the new state is just a copy of the old state.

Figure 16 illustrates the inputs for both the reset and update gates in a GRU, given the input of the current time step and the hidden state of the previous time step. The outputs of two gates are given by two fully-connected layers with a sigmoid activation function.

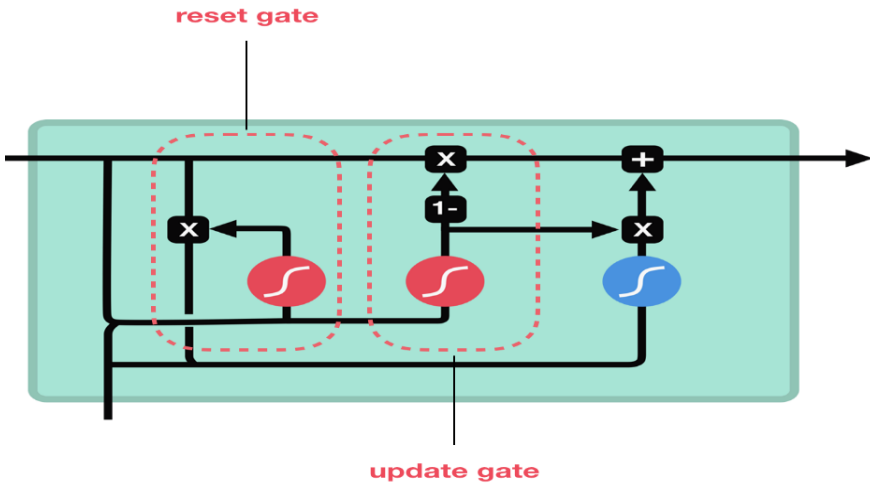

Figure 16: Reset Gate and Update Gate[39]

Mathematically, for a given time step t, the input is a minibatch x_t and the hidden state of the previous time step is h_{t-1}.Then, the reset gate R_t and update gate U_t are computed as follows:

$$R_t = \theta(x_t W_x + h_{t-1} W_h + b)$$

$$U_t = \theta(x_t W_x + h_{t-1} W_h + b)$$

Where Wx are weight parameters and b are bias.

c. Hidden State

Hidden states are a new memory content which will be used in the reset gate to store the relevant information from the past. It is calculated as follows:

1. Multiply the input x_t with a weight W and h_{t-1} with a weight U.
2. Calculate the product between the reset gate R_t and $U_{h_{t-1}}$.That determines what to remove from the previous time steps.
3. If vector close to 0 the past input is deleted and only focus on the last entered input. And if the vector is close to 1, the old state is retained.
4. Sum up the results of step 1 and 2.
5. Apply the nonlinear activation function tanh.

$$H_t{}' = tanh(W_{x_t} + R_t * U_{h_{t-1}})$$

d. Final memory at current time step

As the last step, the network needs to calculate H_t vector which holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content $H_t{}'$ and from the previous steps H_{t-1}. That is done as follows:

$$H_t = U_t * H_{t-1} + (1 - U_t) * H_t{}'$$

1. Apply element-wise multiplication to the update gate $U_t$and H_{t-1}.
2. Apply element-wise multiplication to (1-z_t) and$H_t{}'$.
3. Sum the results from step 1 and 2.

The model can learn to set the vector U_t close to 1 and keep a majority of the previous information. Since U_t will be close to 1 at this time step, $1 - U_t$ will be close to 0 which will ignore big portion of the current content.

Here is a figure 17 which explains where and at what time step the above equations happen:

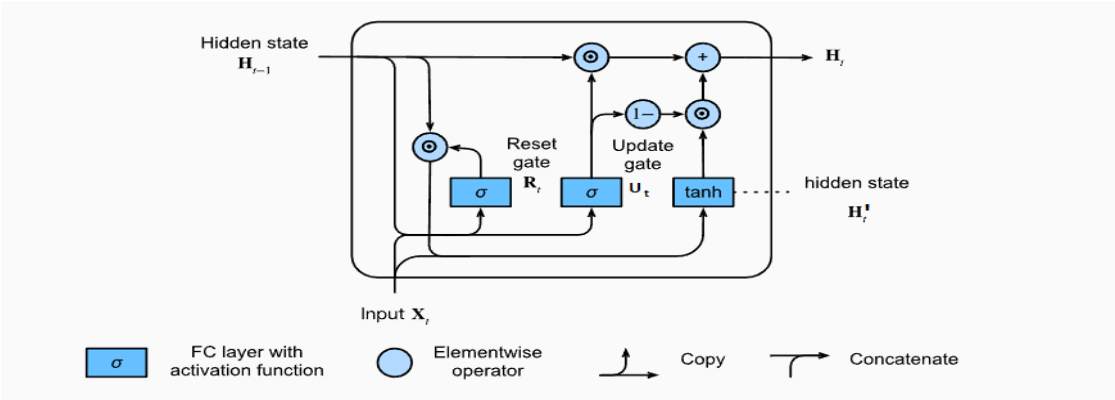

Figure 17: GRU model[40]

This is how GRUs are able to store and filter the information using their update and reset gates. That eliminates the vanishing gradient problem since the model is not washing out the new input every single time but keeps the relevant information and passes it down to the next time steps of the network.

5. Encoder-Decoder

In audio Captioning Encoder decoder models allow for a process in which a machine learning model generates a sentence describing an audio. It receives the audio as the input and outputs a sequence of words describing this audio. It is a major problem for sequenced-data based models, whose input and output are both variable-length sequences. To handle this type of inputs and outputs, we can design an architecture with two major components[41].

The first component is an encoder: it takes a variable-length sequence as the input and transforms it into a state with a fixed shape.

The second component is a decoder: it maps the encoded state of a fixed shape to a variable-length sequence.

This is called an encoder-decoder architecture, which is depicted in the figure 18

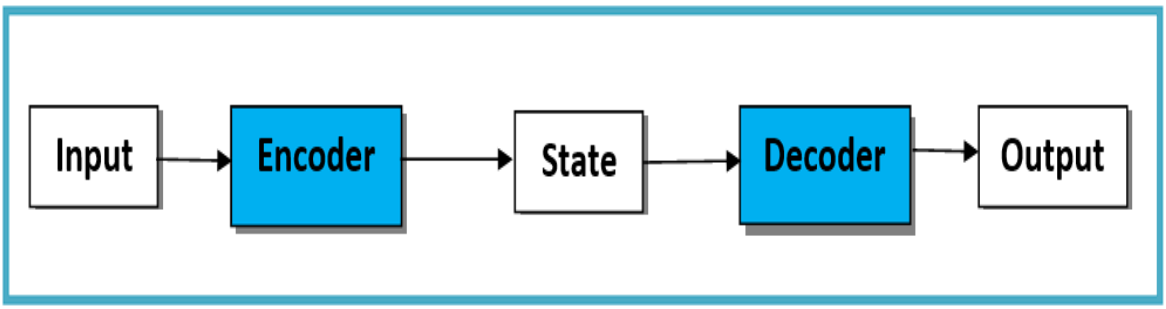

Figure 18: The encoder-decoder architecture

5.1.Encoder

Encoding means to convert data into a required format. For example, we convert an audio into a two-dimensional vector, this two-dimensional vector[42]. For this example, the encoder is built by stacking GRUs. We use this type of layers because its structure allows the model to understand context and temporal dependencies of the sequences. The output of the encoder, the hidden state, is the state of the last GRU timestep. The output of the encoder, a two-dimensional vector that encapsulates the whole meaning of the input sequence[41]. The length of the vector depends on the number of cells in the GRU as demonstrated in the figure 19.

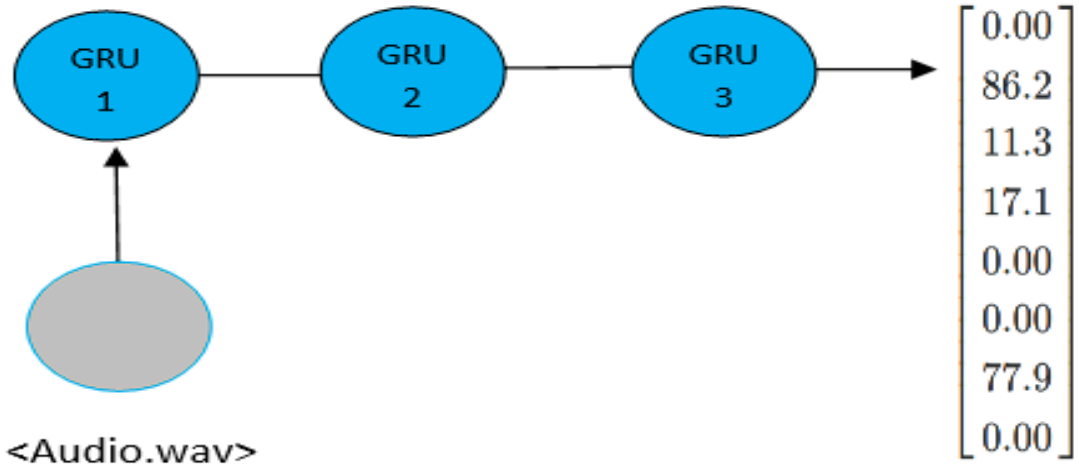

Figure 19: Encoder architecture

5.2.Decoder

To decode means to convert a coded message into intelligible language. In the machine learning model, the role of the decoder will be to convert the two-dimensional vector into the output sequence, the English sentence[43]. It is built with RNN layers and a dense layer to predict the English word as demonstrated in the figure 20.

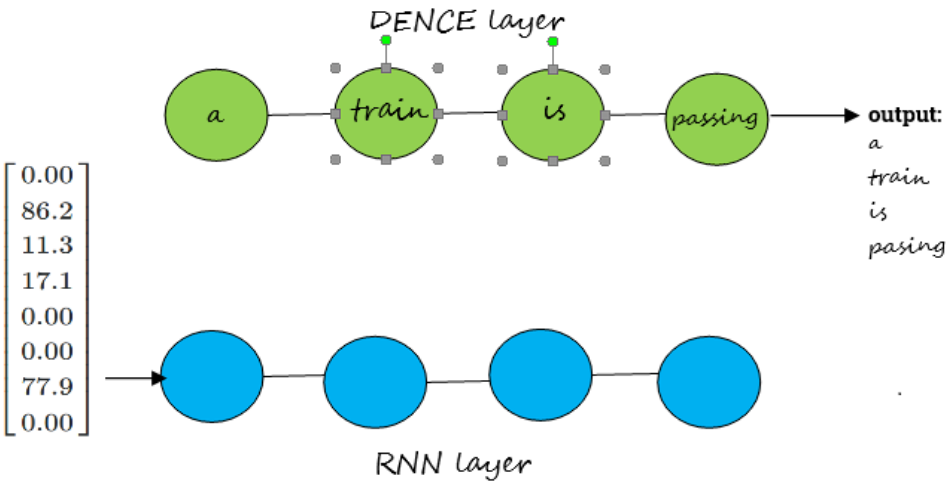

Figure 20: Decoder architecture

In the end, the encoder-decoder architecture contains both an encoder and a decoder, with optionally extra arguments. In the forward propagation, the output of the encoder is used to produce the encoded state, and this state will be further used by the decoder as one of its inputs. One of the major advantages of this model is that the length of the input and output sequences may differ[43]. The major limit of this simple encoder decoder model is that all the information needs to be summarized in one dimensional vector, for long input sequences that can be extremely difficult to achieve.

6. Transformer

The game-changing part for sequencer data was developed when a new architecture was introduced called Transformers and which was 1rst time shown in an article that based on a concept called Attention Is Everything. The document Attention is all you need [44] presents an architecture called Transformers. Transformers is an architecture for turning a sequence into an antidote while helping two other parts, namely encoders and decoders, but it differs from the sequence described earlier in RNN or GRUs. It therefore does not implement recurrent neural networks[44].

The recurrent neural network was so far one of the best ways to capture the small dependency on a sequence. However, the team presenting this paper titled specified above proves that the architecture with only an attention mechanism does not use RNN can improve its results in the task of translation and other NLP tasks. In transformers both encoder and decoder are made up of modules that can talk to each other multiple times.

First, we have to encode our inputs. A small but important part of this model is the position and coding of different inputs. Since we don't have a recurrent neural network that can remember how sequence is fed into the model, we have to somehow give each input or part of a sequence a relative position since a sequence depends on the order of elements[44]. These positions are added to the embedded representation of each word. The figure 21 below represents the detailed architecture of transformer model[44].

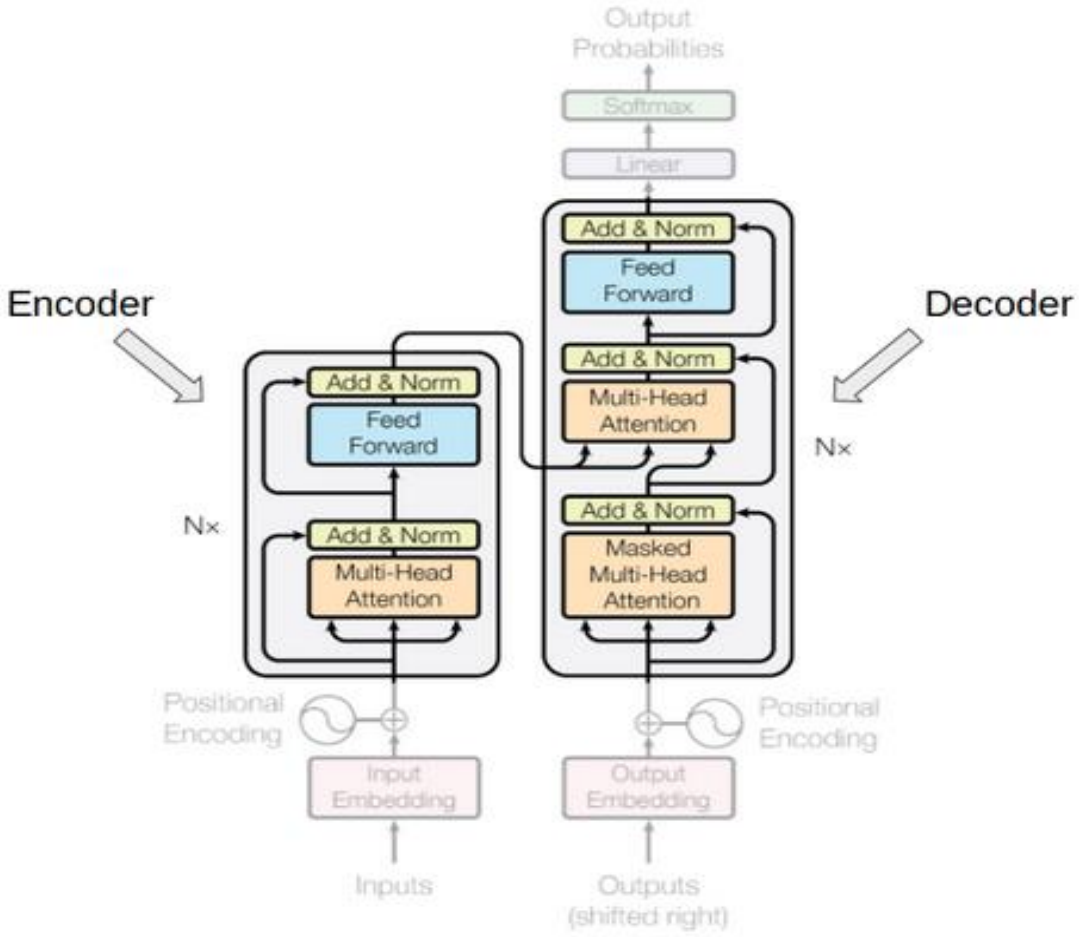

Figure 21: Transformer architecture [44]

6.1.Deferent transformers models

These are some of the most language models based on the transformer's architecture illustrated in the table 4.

language models	Description
GPT3 (Open-AI)	The most controversial pre-trained models, by Open-AI, the large-scale transformer-based language model was trained on 175 billion parameters, which is 10 times more -sparse language model. The model was trained to perform well on many NLP datasets.
language models	**Description**
BERT (Google)	These are the bi-directional encoder representations of Transformers. Is a pre-trained NLP model, which is developed by Google in 2018. BERT has been pre-trained on 250 million Wikipedia words and 800 million book corpus words. anyone working with BERT can train their own module with up to 30 minutes.
ELMO (Alan LP)	ELMO is also known as integration for the language model is a deep contextualized word representation that model's syntax and semantic words, as well as logistical context. The model has been pre-trained on a huge corpus of text and learns functions from bidirectional models.

Table 4- lagunage model exemple

6.2.Transformer Architecture

The Transformer architecture inherited the Encoder-Decoder pattern. The encoding part contains 6 encoders mounted one after the other. The decoding part consists of 6 decoders also mounted one after the other but each taking, as an additional input, the output of the 6th encoder[44].

The input of one encoder is the output of the previous one. The input of the first encoder is embedding vector. Also, the input of a decoder is the output of the previous decoder plus the words already encoded as shown in the figure 1-23. The last decoder is connected to a Linear neural network and a SoftMax block. The role of this block is to make it possible to identify which words of the vocabulary correspond to the outputs of the last encoder[44]. The elementary blocks of the Transformer are the encoders and the decoders.

Let's take a closer look at these two elements, the figure 22 shows how transformer architecture is built.

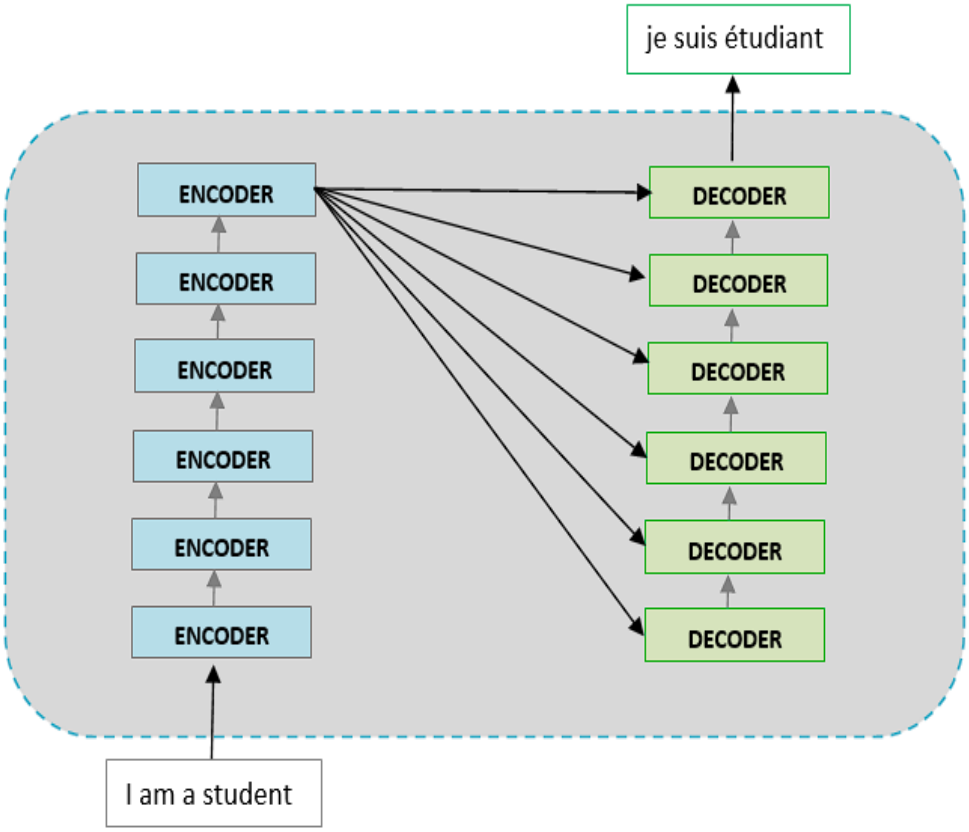

Figure 22: Transformer Architecture [45]

The encoder consists of two blocks which are both neural networks: A Self-attention layer and a forward propagation network or Feed-forward Neural Network as demonstrated in the figure 23. The Self-Attention layer is the central element of the Transformer architecture. Its role is to maintain the interdependence of words in the representation of sequences. We will see the attention mechanism

in more detail below. The decoder is also composed of a Self-attention block and a Feed-forward but it also contains an Encoder-Decoder Attention layer which aims to allow the decoder to implement the attention mechanism between the input sequence encoded and output sequence being decoded.

6.3.The Attention mechanism

The concept of attention is to measure how closely two elements of two sequences are related. In a sequence-to-sequence context in NLP, the aim of the attention mechanism will be to tell the rest of the model which words of sequence B should be paid the most attention to when processing a word of sequence A.

In the figure 23 we show an example, the clearer a cell the stronger the link between the two words to which it corresponds. we see that, a word has a strong link with its literal translation [44].

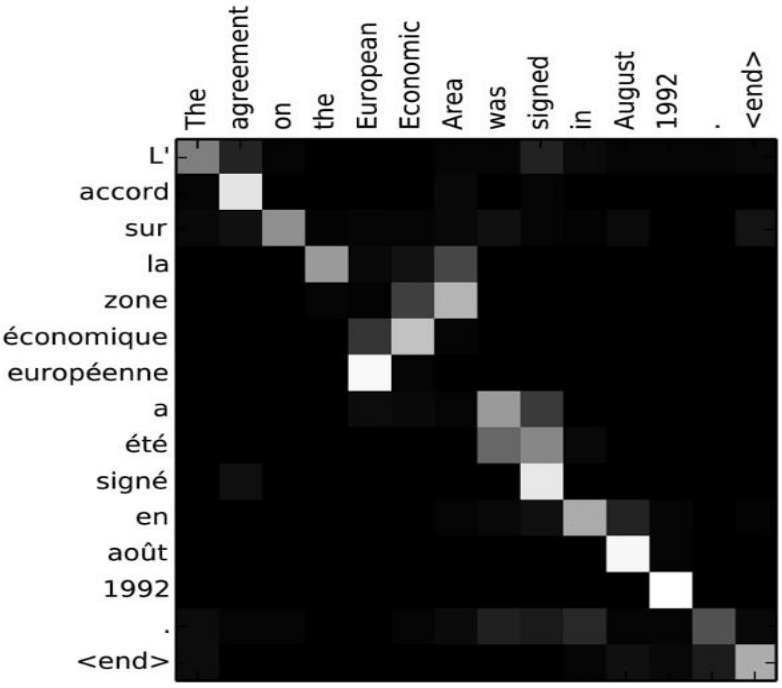

Figure 23: Distribution of attention between two sequences[46]

6.4.Self-attention

Self-attention is the mechanism of attention applied to a single sequence. The self-attention layer determines the interdependence of the different words of the same sequence in order to associate a relevant representation encoding with it.

The self-attention process will therefore aim to detect the link between two words in the same sentence[44].

6.5.Multi-head attention

Multi-head Attention is intended to have multiple representation subspaces that prevent the representation from being fully biased if one (head) layer of attention is. The self-attention vector in multi-head is nothing but the concatenation of the output vectors of each head[44]. The Transformer was clearly a revolution when it was released in that it was both very efficient as a translation model and much faster to train compared to its predecessors. But especially by the influence he has on the NLP through the pre-trained models.

7. Conclusion

This chapter was split between two parts in each part we presented some definitions and some frequently used methods.

Details of the first part was that we represented what are the basic machine learning definitions and tasks, we also dive into some evaluation models and parameters.

Second part was about deep learning we presented the basic architecture of a neuron network then we dive deeper to some NN based models that are frequently used in deep learning and we ended this part with introduction to transformers and their attention mechanism.

In the next chapter expect to see more about data pre-processing in specific the deferent tools and methods the handle both types of the data that we need in our work audio and text.

Chapter 2: Audio processing / Text processing

Introduction

In this Chapter, we introduce a few fundamental concepts behind audio signal processing and text processing that will be required to perform our work.

Part 1: Audio processing

1. Introduction

Sound is one of our primary means of perceiving the world around us. It is essential to communicate with our environment and our peers. It can also be a vehicle for artistic expression and experience, for example music. Our thesis includes sound as a component.

2. Sound wave

Sound is frequently defined as either an auditory sensation or a disturbance in a medium that produces an auditory sensation. Sound is a physical phenomenon that describes waves that originate in one location and travel through a medium to another location where they can be heard or measured[48].

A sound wave is a pressure vibration caused by the movement of energy traveling through a medium e.g., air as it propagates away from its source. As sound passes through the air, the air particles move left and right due to the energy of the sound wave passing through it. It's the vibrating air molecules that cause the human eardrum to vibrate, which the brain then interprets as sound. Air molecules do not travel from the noise source to the ear[47]. Each individual molecule only moves a small distance as it vibrates, which causes the adjacent molecules to vibrate in a rippling effect all the way to the ear as shown in the figure 24.

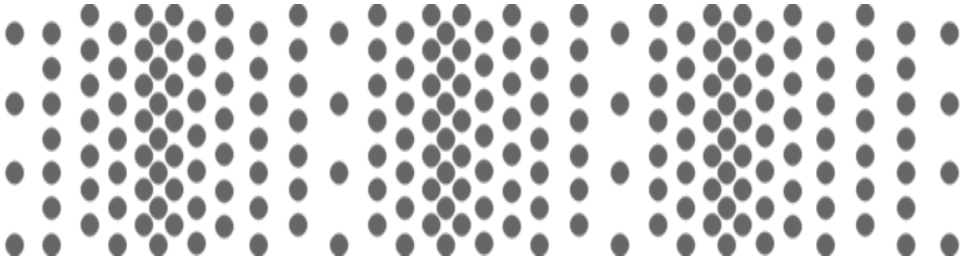

Figure 24: Longitudinal sound wave showing compression and rarefaction of air particles

Compression happens in the region in a longitudinal wave where the particles are closest together. Rarefaction is a region in a longitudinal wave where the particles are furthest apart. Sound waves are longitudinal and should not be confused with transverse waves. Most waves are transverse, including light and the ripples we see on water.

Transverse waves vibrate at 90 degrees to the direction of the wave. In contrast, longitudinal waves have vibrations along the same axis as the direction in which the wave is traveling [48] .this figure 25 below shows how a transverse wave looks.

Figure 25: A simple sine wave, shown as a transverse wave

3. Wave form

A waveform is a graphical representation of a sound wave as it moves through a medium over time. A waveform is a two-dimensional representation of a sound. The two dimensions in a waveform display are time and intensity[47].

Waveforms are also known as time domain representations of sound as they are representations of changes in intensity over time. The intensity dimension actually displays sound pressure. Sound pressure is a measure of the tiny variations in air pressure that we are able to perceive as sound. The greater the change in pressure, the louder the sound that we hear as demonstrated in the figure 26.

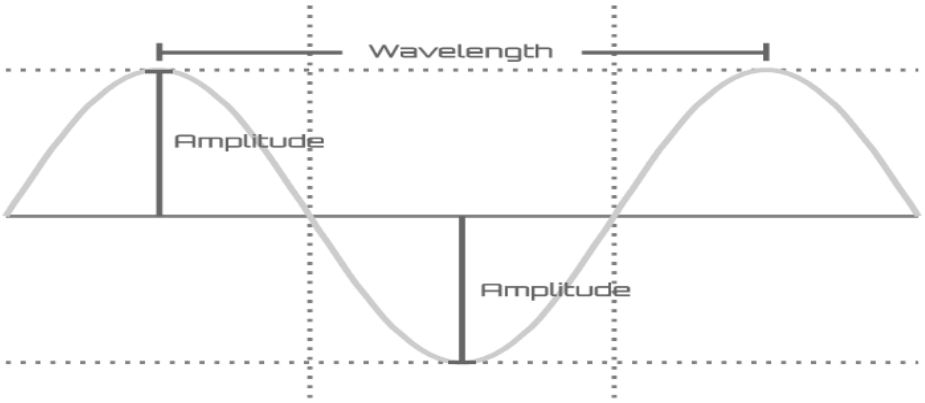

Figure 26: Waveform diagram showing the wavelength and amplitude of a soundwave

Any waveform has four fundamental characteristics: Wavelength, Amplitude, Frequency, Velocity.

3.1. Wavelength

The wavelength of a wave is the length in meters from the start to the end of one full cycle of the waveform from crest to crest.

3.2. Amplitude

The amplitude is the maximum displacement of a wave from the centerline to the peak, not from peak to peak. The greater the distance from the centerline of a waveform the more intense the pressure variation will be within a medium, hence the louder it is perceived.

Amplitude is measured in two ways:

- Zero to peak value which measures the maximum positive or negative signal level
- RMS value measures a more meaningful average level, like that at which humans can hear.

3.3. Frequency

Frequency is how many complete waves there are per second passing a certain point. The frequency indicates the rate of pressure variations or cycles per second of a wave. Frequency is measured in Hertz[49]. The frequency of a sound determines the pitch, the sensation of how low or how high a sound is. Lower frequency sound

waves have longer wavelengths and lower pitch. Higher frequency sound waves have shorter wavelengths and a higher pitch as demonstrated in the next figure 27.

Figure 27: Low-frequency wave in comparison to a high-frequency wave

The frequency range in which humans can hear is 20Hz to 20,000Hz and is called the audible range. The formula for frequency is:

$$f = 1 / t$$

- where: f represents the frequency in Hertz and t represents the period in seconds. So, for our bell striking example, for a period of $0.0023s$:

$$f = 1/t = 1/0.0023 = 434.7\ Hz.$$

The frequency of the sound generated by striking a bell is about 435 Hz.

3.4. Velocity

The velocity is the speed and direction of a soundwave. Soundwaves travel at different speeds through different mediums. Through the air, sound travels at 344 meters per second. Generally speaking, the denser the medium the faster sound travels through it[50]. To find the velocity of a wave the following equation is used:

$$\text{Velocity (V)} = \text{Frequency (f)} \times \text{Wavelength } (\lambda)$$

4. Sound envelope

The envelope of a sound displays how the level of a sound wave changes over time. The envelope of a wave helps establish the sound's unique individual quality; it has a significant influence on how we interpret sound[48].

5. Audio signal Representation

waveforms make up the basic ingredients of sound; sine wave, square wave, triangle wave, and sawtooth wave.

5.1 Sine wave

A sine wave is the simplest of all waveforms and contains only a single fundamental frequency and no harmonics or overtones as shown in the figure 28.

It is the fundamental frequency that determines the pitch of a sound. Virtually all musical sounds have waves that are more complex than a sine wave. It is the addition of harmonics and overtones to a wave that makes it possible to distinguish between different sounds and instruments.

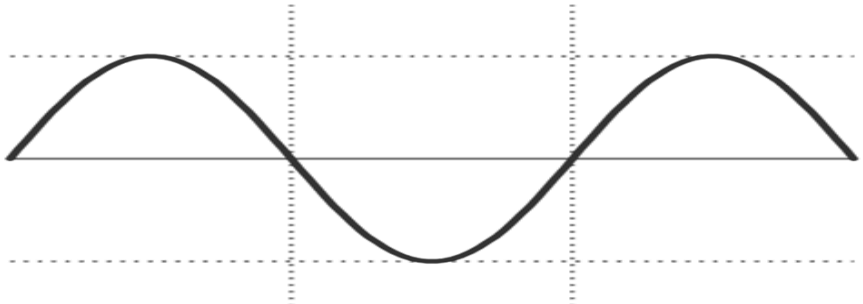

Figure 28: Visual representation of a sine wave

6. Phases of an audio signal

Sound waves occur in cycles, they proceed through repetitions. Phase is defined as how far along a waveform is in its current cycle the deferent wave phases are defined in the figure 29. The starting point of a wave is 0 degrees, the peak of a wave is 90 degrees, the next neutral pressure point is 180 degrees, the peak low-pressure zone is 270 degrees, and the pressure rises to zero again at 360 degrees.

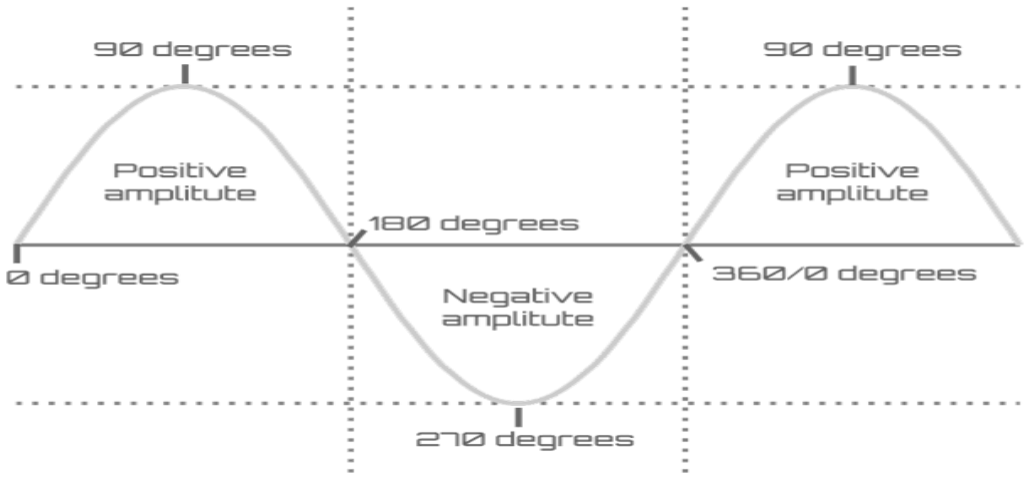

Figure 29: Graph representing the different phases of a wave

7. Digital audio processing

This Part is about the different steps for audio signal processing, we will define sampling and quantization and explain how they work. This thesis revolves around audio signal processing, which is the theory and methods of processing audio signals.

Audio signals are simply audible audio signals, while a signal is something that is measured in time[53]. In this context, that thing is pressure, because what we perceive with our ear's changes with pressure. We may wish to manipulate audio signals for technical purposes, such as creating music, or we may wish to compress audio or music signals. Before we start processing audio, we must first understand the nature of audio signals.

7.1. Audio signal pre-processing

If needed, the audio data is pre-processed. The role of this step is to enhance certain characteristics of the signal for further analysis. This is achieved by reducing the effects of noise or by emphasizing the target sounds in the signal.

Knowledge about the recording conditions and characteristics of target sounds can be utilized in the pre-processing stage to enhance the signal. In the case where the audio data is captured in non-uniform recording settings, down-mixing the audio signal into a fixed number of channels along with re-sampling it into fixed sampling frequency will result in converting the input data into a uniform format for further analysis. After the pre-processing phase, the audio data is now appropriate to be used in the feature extraction phase.

7.2. Digital audio signal

As previously stated, sound can refer to either an auditory sensation in the air or a disturbance in a medium that causes such a sensation. As a physical phenomenon, sound can be understood as vibration, or the movement of molecules that causes pressure changes in a medium. These air vibrations can then be measured using a microphone, where changing pressure causes a diaphragm to move. A voice coil and a magnet convert this movement into a voltage signal. So, we now understand how a signal transforms from the movement of molecules in a

gas to an electrical signal. A microphone's voltage signal is a continuous signal, which means it can take on a value, just like pressure at a point in space. Furthermore, even if the range is limited, it can take on an infinite number of different values at any given time[53].

A computer can only store a limited number of numbers, and those numbers can only have a limited number of values. As a result, we must transition from measuring an infinite number of time values that can take on an endless number of different values to that. To put it another way, we must convert analog signals to digital signals. An ADC is a device or chip that converts analog signals to digital signals through sampling and quantization. A DAC, on the other hand, reconstructs digital signals into analog ones. Figure 30 depicts an example of an audio processing system.

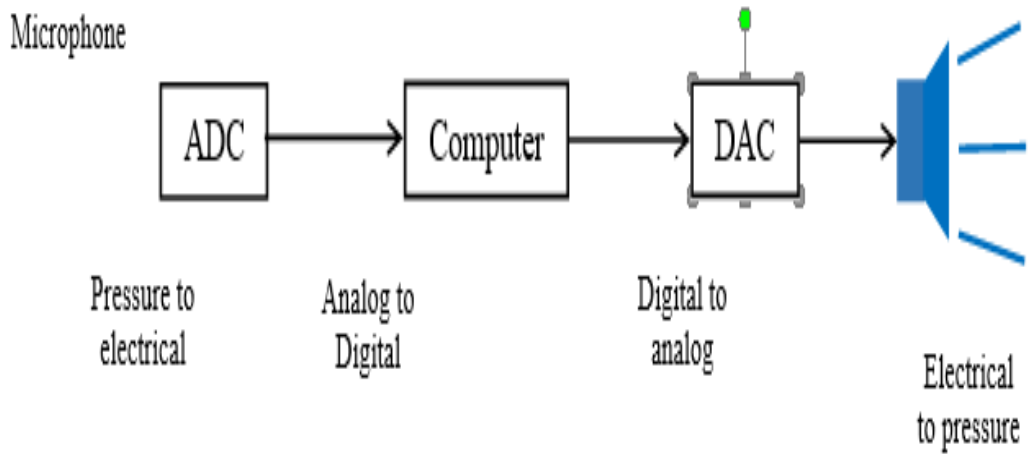

Figure 30: Audio processing system[54]

Vibrations in the air are converted from a pressure signal to an electrical signal by the microphone. The analog signal from the microphone is converted into a digital signal by the ADC, and then the digital signal can be processed by the computer. The processed digital signal can then be converted into an analog signal by a DAC, and finally, it can be converted back into a compression signal by an active amplifier.

7.3.Sampling

Samples are created by recording an analog signal at evenly spaced points in time. Sampling is the process of recording an analog signal at regular discrete moments of time. The sampling rate f s represents the number of samples taken per second[55].

The sampling interval:

$$T_s = 1/f_s$$

Is the time interval between samples. Although any sampling frequency above 40 kHz would be sufficient to capture the entire range of audible frequencies, 44,100 Hz (or 44.1 kHz) is a widely used sampling rate. When sampling at higher rates, more samples are generated, resulting in a much higher demand for memory to store the samples.

7.4.How is sound sampled and stored in digital form?

To do this, sound is captured usually by a microphone and then converted into a digital signal. An ADC converter will capture a sound wave at regular time intervals. This recording is known as a sample. This data is then stored in a file for later use.

7.5.Signal sampling

When a computer records digital audio, it measures the sound pressure level multiple times per second. These measurements are often called samples. Being digital, the samples are quantized that is, they can only take on certain discrete values as compared to the continuous range of possible values in the actual analog sound wave. Commonly, the samples can take on the integer values between -32768 and +32767 the range of numbers representable with 16 bits with positive values representing the sound pressure level being above the ambient atmospheric pressure and negative values representing the sound pressure level being below the ambient atmospheric pressure[55].

We could plot the samples on a graph represented in the figure 0-3. The X axis would represent time and the Y axis the sound pressure level (the value of the sample). In the graph below, the distance between adjacent peaks represents 100 samples with a sampling rate of 44,100 Hz. Thus, the sound has a fundamental frequency of 441 Hz (44,100 / 100 = 441). Listening to it would sound like the A above middle C which has a fundamental frequency of 440 Hz0

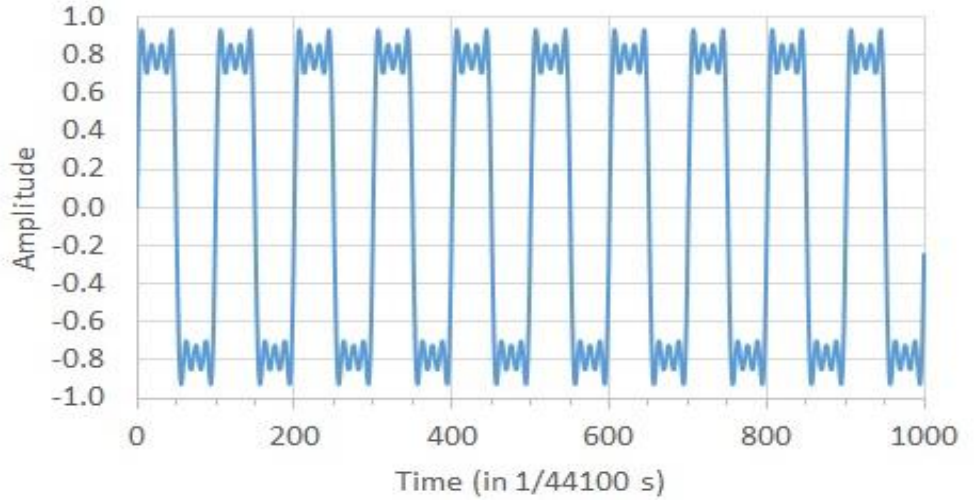

Figure 31: Time and sound pressure plot

The sound wave shown above in the figure 40 looks somewhat like a square wave. If you compare it to a 441 Hz pure tone shown below in the figure 32, it is obvious that it is not a pure sinusoid and not a pure tone.

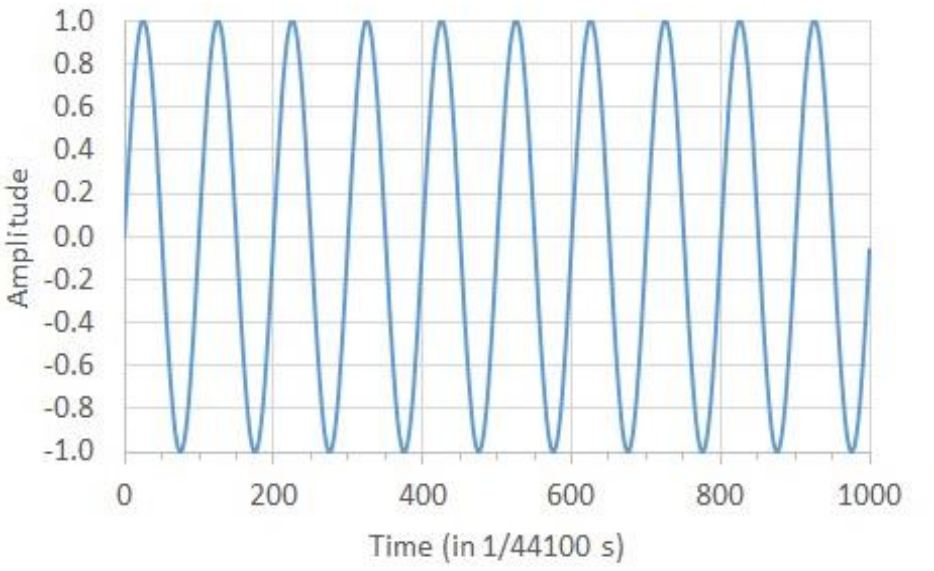

Figure 32:Time and sound pressure plot

7.6.Audio signal quantization

The digitization of a sampled signal with continuous amplitude is called quantization. A quantizer is a signal processing block, that maps a continuous amplitude to a discrete amplitude. The output of the quantizer is discrete, meaning that it can only output Q different values[53]. Practically, the quantizer is an ADC, since it

maps the continuous input amplitude to a digital representation of this value. Formally, the quantized output Q[x] of some input value x, is given by:

$$Q[x] = \arg\min 1 \ \in |1 - x|$$

In order for a signal to be suitable for treatment by numerical circuitry, it must first be represented in a numerical format, or quantized. That is, a continuous range of values is replaced by a limited set of values separated by discrete steps. Usually, the number of steps is chosen to be a power of two, for the reason that it yields the most economical representation in binary digital electronics. Naturally, the quality of the approximation depends on the number of steps used to approximate the original signal.

7.7. The Fourier transforms

The Fourier transform converts a set of time-domain data to frequency-domain data and vice versa. It means that it can be used to take samples and determine the sinusoids that could be used to create the samples. This is known as a Fourier analysis. The Fourier transform can also be used to reverse the process, taking the sinusoids and recreating the samples[56]. This is known as Fourier synthesis. When you perform a Fourier analysis on the samples in the first graph, you get the following results in figure 33.

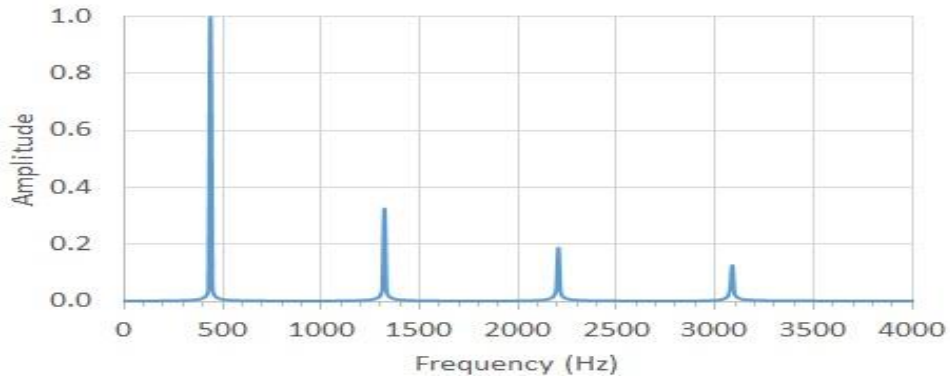

Figure 33: The Fourier Transforms Plot

Looking at the results of the Fourier analysis tells us that there are four frequencies present in the sound wave shown in the table 5:

Frequency	Amplitude
441 Hz	1
1323 Hz = 3 X 441 Hz	1/3
2205 Hz = 5 * 441 Hz	1/5
3087 Hz = 7 * 441 Hz	1/7

Table 5- frequencies present in the sound wave

It is important to note that we will be translating the audio from the time domain to the frequency domain by using this transformation. Here are some key differences between the two:

1. The time-domain examines the amplitude variation of the signal over time. This is useful for comprehending its physical form. We'll need time on the x-axis and amplitude on the y-axis to plot this. The shape helps us predict how loud or quiet the sound will be.
2. The frequency domain examines the constituent signals in our recording. This allows us to find a fingerprint of the sound. We need frequency on the x-axis and magnitude on the y-axis to plot this. The greater the magnitude, the more significant the frequency. The magnitude is simply the absolute value of our FFT results.

7.8 The Short-Time Fourier Transform

The Fourier transform tells us how much of each frequency is present in a signal. If the spectral content of the signal does not change significantly over time. However, if the signal changes over time, the Fourier transform will be unable to distinguish between the various spectral content changes[56].

The STFT attempts to address the classic Fourier transform's lack of time resolution. The input data is divided into many small sequential pieces called frames, and the Fourier transform is applied sequentially to each of these frames. The result is a time-dependent representation that shows how the spectrum changes as the signal progresses.

As a result, at the frame boundaries, there is frequently a discontinuity or break in the signal. This introduces spectral components into the transform that were not present in the original signal, which is known as spectral leakage. The solution is to apply a windowing function to the frame, which gently scales the signal's amplitude to zero at each end, reducing discontinuity at frame boundaries. When these windowing functions are applied to a signal, some information near the frame boundaries is clearly lost. As a result, overlapping the frames is a further improvement to the STFT[56]. Information that is lost at a frame boundary is picked up between the boundaries of the next frame when each part of the signal is analyzed in more than one frame.

8. Feature Extraction

Most real-world data, and in particular sound data, is very large and contains much redundancy, and important features are lost in the cacophony of unreduced data. The data reduction stage is often called feature extraction, and consists of discovering a few important facts about each data item. The features that are extracted from each case are the same, so that they can be compared.

8.1 The amplitude envelope

AE aims to extract the maximum amplitude within each frame and string them all together. It is important to remember that the amplitude represents the volume of the signal.

First, we split up the signal into its constituent windows and find the maximum amplitude within each window. From there, we plot the maximum amplitude in each window along time[57]. We can use the AE for onset detection, or the detection of the beginning of a sound. In various speech processing applications this could be someone speaking or external noise, whereas in Music Information Retrieval this could be the beginning of a note or instrument. The main downfall of the AE is that is not as robust to outliers RMSE which will we see next in this chapter. Here is how we can formalize this concept:

$$\mathrm{AE}_t = \max_{k=t\cdot K}^{(t+1)\cdot(K-1)} s(k)$$

9. Spectrogram

A spectrogram is a detailed view of audio, able to represent time, frequency, and amplitude all on one graph [60] .Sound spectrum is a representation of a sound usually a short sample of a sound around 10 to 30s of length in terms of the amount of vibration at each individual frequency. It is usually presented as a graph of either power or pressure as a function of frequency. A spectrogram is built from a sequence of spectra by stacking them together. The final graph has time along the horizontal axis, frequency along the vertical axis, and the amplitude of the signal at any given time and frequency.

9.1.Mel scale

The Mel Scale is a logarithmic transformation of a signal's frequency. The core idea of this transformation is that sounds of equal distance on the Mel Scale are perceived to be of equal distance to humans[61]. For example, most human beings can easily tell the difference between a 100 Hz and 200 Hz sound. However, it is actually much harder for humans to be able to differentiate between higher frequencies, and easier for lower frequencies.

9.2.Mel spectrograms

Mel Spectrograms are spectrograms that visualize sounds on the Mel scale as opposed to the frequency domain. We can see in the next figure 34 how each sound takes a unique shape based off of the sound it actually produces[62].

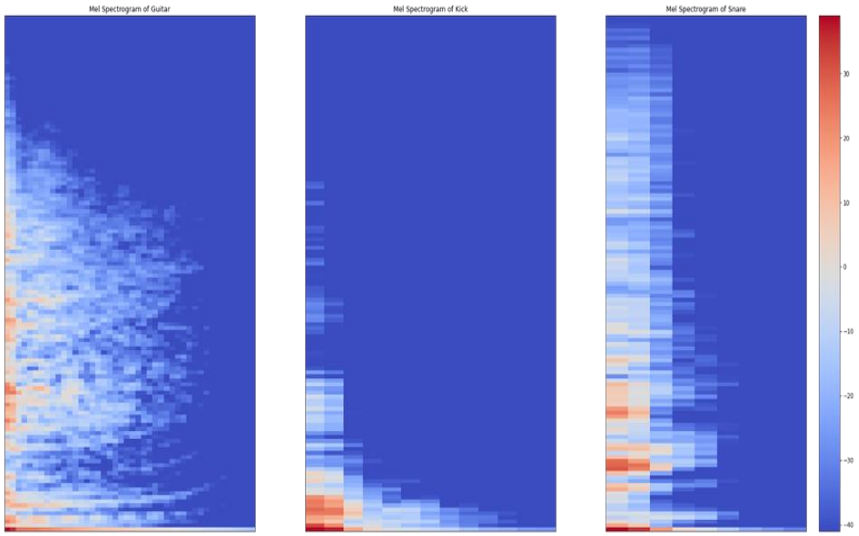

Figure 34: Mel Spectrograms Plot

Part 2: Text Processing

1. Introduction

In this part, five levels of representations for texts are defined. will provide background information on NLP and descriptions on the different levels of language processing an NLP system can employ. NLP is a subfield of Artificial Intelligence which tries to accomplish human-like language processing of naturally occurring texts by computer systems.

Humans utilize different levels of processes in order to understand language. Similarly, NLP applications utilize different levels of language processing to achieve their goals.

2. Natural language

A natural language does not define a language in the strict sense of the term, but it is the natural way of expressing humans, unlike binary and the languages used in programming. It is the language of emails, descriptions, chat...

NL is a symbolic system that is embodied externally as voice and consists of vocabulary and grammar. The biggest difference between natural and artificial languages lies in ambiguity.

NLP is a technology that uses computers as tools to perform various processing on human specific written and verbal natural language information. NLP is a branch discipline in the fields of AI. It studies various theories and methods for effective communication between human beings and computers using natural languages[64].

NLP needs to develop models that express language capability and language application, establish a computing framework to implement such language models, propose corresponding methods to continuously improve the models, design various practical systems based on the models, and explore evaluation technologies of these systems.

NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to understand its full meaning,

3. Natural language processing tasks

NLP is used to understand the structure and meaning of human language by analyzing different aspects like syntax, semantics, pragmatics, and morphology. Then, computer science transforms this linguistic knowledge into rule-based, machine learning algorithms that can solve specific problems and perform desired tasks[64].

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data[65].

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

1. Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data
2. Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context.
3. Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context.
4. Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

4. Challenges of natural language processing

There are a number of challenges of natural language processing and most of them boil down to the fact that natural language is ever evolving and always somewhat ambiguous[65]. They include:

1. Precision. Computers traditionally require humans to speak to them in a programming language that is precise, unambiguous and highly structured or through a limited number of clearly enunciated voice commands. Human speech, however, is not always precise; it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

2. Tone of voice and inflection. Other difficulties include the fact that the abstract use of language is typically tricky for programs to understand
3. Evolving use of language. Natural language processing is also challenged by the fact that language and the way people use it is continually changing.

5. Text preprocessing

There are different ways to preprocess text. Here are some of the approaches that are widely used.

5.1.Lowercasing

Lowercasing ALL your text data, is one of the simplest and most effective form of text preprocessing[66]. Here is an example of how lowercasing solves the sparsity issue, where the same words with different cases map to the same lowercase form. Showed in this table 6.

Raw	Lowercased
Canada **CanadA** **CANADA**	**canada**
Raw	**Lowercased**
TOMCAT **Tomcat** **tomcat**	**tomcat**

Table 6- Lowe casing Example

5.2.Stemming

Stemming is the process of reducing inflection in words to their root form. Stemming uses a crude heuristic process that chops off the ends of words in the hope of correctly transforming words into its root form. There are different algorithms for stemming[66]. The most common algorithm, which is also known to be empirically effective for English, is Porters Algorithm. Here is an example of stemming in action shown in the table 7.

	Original Word	Stemmed Words
0	Connect	Connect
1	Connected	Connect
2	Connection	Connect
3	Connections	Connect
4	Connects	Connect
0	Trouble	Trouble
1	Troubled	Trouble
2	Troubles	Trouble

Table 7- Porters Algorithme Example

5.3.Lemmatization

Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It may use a dictionary such as WordNet for mappings[67].

5.4.Stop word Removal

Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. The intuition behind using stop words is that, by removing low information words from text, we can focus on the important words instead[68]. Here is an example of stop word removal in table 8.

Original Sentence = this is a text full of content and we need to clean it up
Sentence with stop words removed = w w w text full w content w w w w clean w w

Table 8- Text Stop word removal example

5.5.Normalization

Text normalization is the process of transforming a text into a standard form. For example, the word “cats” and “cuts” can be transformed to “cat”[66]. Here’s an example of words before and after normalization in this next table 9.

Raw	Normalized
2morow 2moro 2mrw tomrw	tomorrow
b4	before

Table 9- Text Normalization Example

5.6. Tokenization

Is the first step in any NLP pipeline. A tokenizer breaks unstructured data and NL text into chunks of information that can be considered as discrete elements.

The token occurrences in a document can be used directly as a vector representing that document[66]. Tokenization can separate sentences, words, characters, or sub words. When the text is split into sentences, it is called sentence tokenization.

```
sent_tokenize('Life is a matter of choices, and every choice you make makes you.')
```

```
['Life is a matter of choices, and every choice you make makes you.']
```

Figure 35: Example of sentence tokenization

For words, it is called word tokenization.

```
word_tokenize("The sole meaning of life is to serve humanity")
```

```
['The', 'sole', 'meaning', 'of', 'life', 'is', 'to', 'serve', 'humanity']
```

Figure 36:Example of word tokenization

6. Different word representations

6.1. Word Embedding

It is an approach for representing words. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meaning to have a similar representation. They can also approximate meaning. A word vector with 300 values can represent 300 unique features[69].

6.1.1. One-Hot Encoding

One hot encoding means converting words of a document in a V-dimension vector and by combining all this we get a single document so at the end we have a two-dimensional array. This technique is very simple[70].

6.1.2. Bag Of Words

Bag of words is a little bit similar to one-hot encoding where we enter each word as a binary value and in a Bag of words, we keep a single row and entry the count of words in a document. So, we create a vocabulary and for a single document, we enter one entry of which words occur how many times in a document[71].

6.1.3. Word2Vec

Word2Vec creates vectors of the words that are distributed numerical representations of word features these word features could comprise of words that represent the context of the individual words present in our vocabulary[72].

As seen in the figure below 37 where word embeddings are plotted, similar meaning words are closer in space, indicating their semantic similarity.

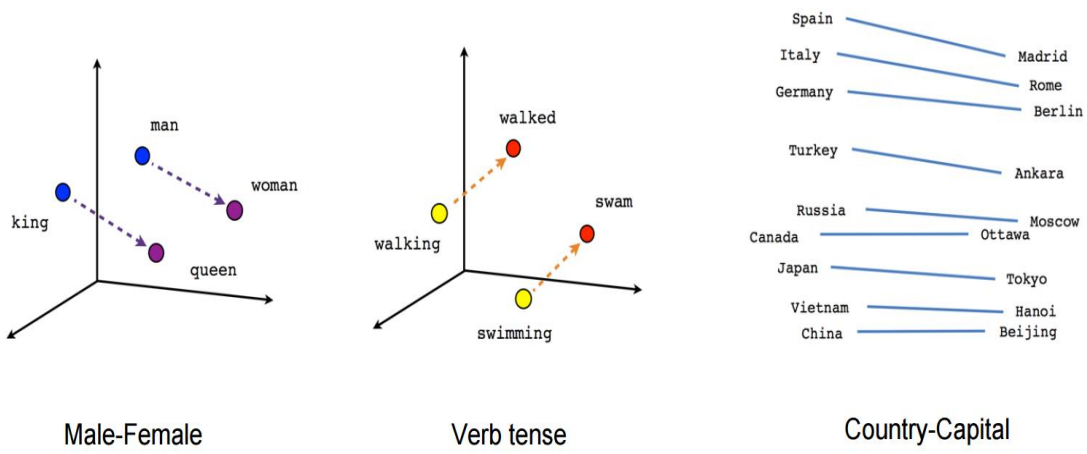

Figure 37:example of embeddings in a graph[73]

Two different model architectures that can be used by Word2Vec to create the word embeddings are the Continuous Bag of Words (CBOW) model and the Skip-Gram model both explain below.

6.1.4. CBOW

Even though Word2Vec is an unsupervised model where you can give a corpus without any label information and the model can create dense word embeddings, Word2Vec internally leverages a supervised classification model to get these embeddings from the corpus. The CBOW architecture comprises a deep learning classification model in which we take in context words as input, X, and try to predict our target word, Y.

6.1.5. Skip-gram

In the skip-gram model, given a target word, the context words are predicted since the skip-gram model has to predict multiple words from a single given word, we feed the model pairs of (X, Y) where X is our input and Y is our label. This is done by creating positive input samples and negative input samples. Positive Input Samples will have the training data in this form: [(target, context)1] where the target is the target or center word, context represents the surrounding context words, and label 1 indicates if it is a relevant pair. Negative Input Samples will have the training data in the same form: [(target, random),0]. In this case, instead of the actual surrounding words, randomly selected words are fed in along with the target words with a label of 0 indicating that it's an irrelevant pair. These samples make the model aware of the contextually relevant words and consequently generate similar embeddings for similar meaning words. This figure 38 shows the deference between CBOW and Skip-gram:

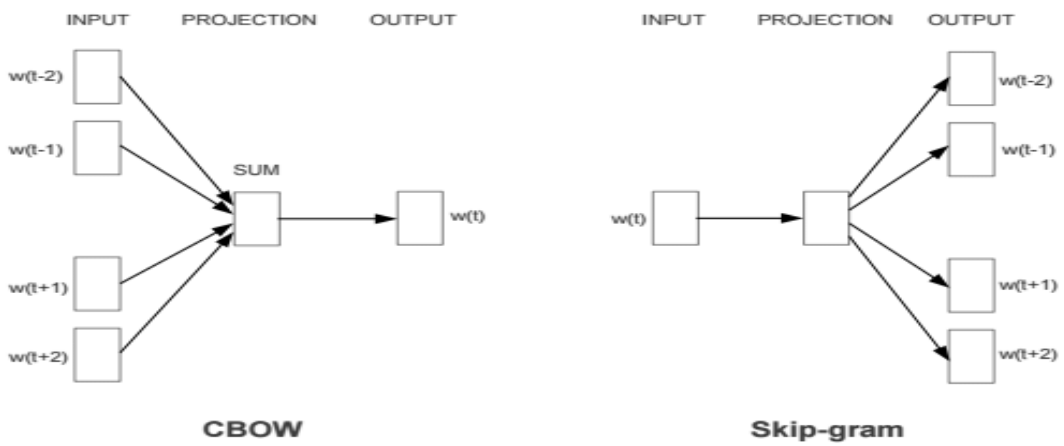

Figure 38: CBOW and Skip-gram examples[74]

7. Word processing with BERT

BERT is a model which is quite bidirectional. Bidirectional indicates that during the training phase, BERT learns information from both the left and right sides of a token's context. A model's bidirectionality is essential for completely comprehending the meaning of a language.

In order to pre-train deep bidirectional representations from unlabeled text, the system uses context conditioning on both the left and right sides of the sentence. As a result, the pre-trained BERT model could also be fine-tuned by adding only one more output layer to produce cutting-edge models for a wide range of NLP tasks[75]. To learn the contextual relationships between words in a text, BERT uses Transformer, a mechanism of attention explained in part 2 of chapter 1.

The transformer implementation has two mechanisms: an encoder that receives a text input and a decoder that predicts the task. Only the encoder mechanism is required because the purpose of BERT is to build a language model. The Transformer encoder reads the entire sequence of words at a time, unlike directional versions that read the text entry sequentially. It is classified as bidirectional as a result of this, while the real term is non-directional. This feature allows the model to learn the context of a word according to its environment this next figure 39 shows an example of this functionality [75].

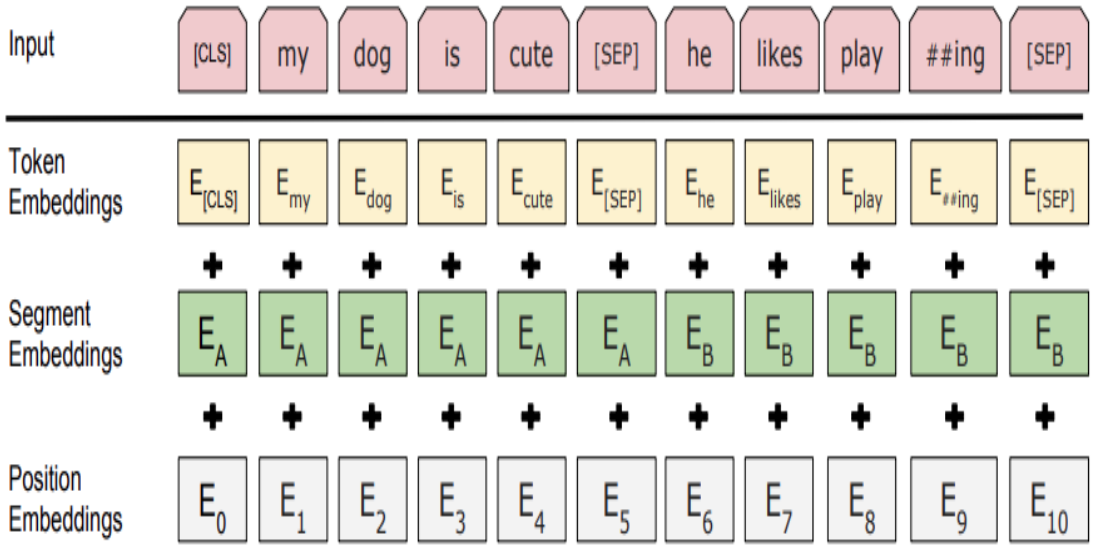

Figure 39:BERT example[76]

During the BERT training process, pairs of sentences are provided as input into the model, and learn how to predict whether or not the second sentence in the pair is the next sentence in the original document. Half of the inputs during training are pairs where the second sentence is the next sentence in the original document while the other half is a random sentence from the group. During training, as described above, the [CLS] code is inserted at the beginning of the first sentence and the [SEP] code is presented at the end of each sentence, with each code containing the insertion sentence indicating the sentence A or sentence B[77].

Finally, topical embedding is customized for each distinctive code that corresponds to its place in the sequence. Before introducing word sequences into BERT, a part of each sequence is replaced by a [MASK] token. The model then attempts to predict the original value of the masked words using the context provided by the other unmasked sentences in the sequence. This is followed by multiplying the encoder output vectors by the integration matrix, transforming them into a vocabulary dimension and calculating the probability of each word of the vocabulary using SoftMax all of this is well represented in the figure 40 [77].

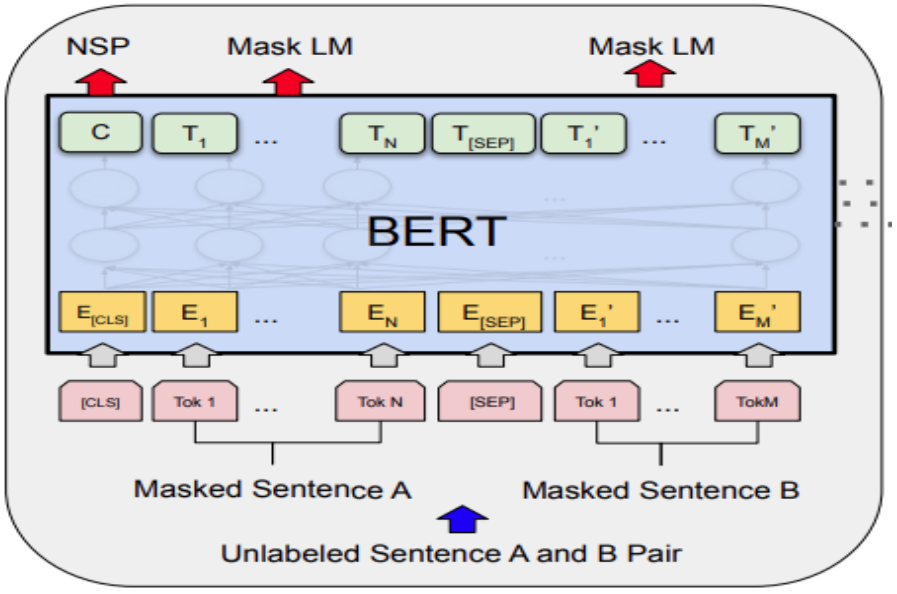

Figure 40: Pre training BERT Example [78]

8. Conclusion

This chapter is split between two parts in the first part we represented audio signal in every form and some of the tools to extracted features from digital audio data.

Second part was about text processing and the deferent methods of feature extraction we ended this part we the most resent developed text processing method wish is BERT.

In the next chapter we will be defining the model we used and the deferent architectures we developed as well as some experiments.

Chapter 3: Proposed approach

1. Introduction

Automated audio captioning (AAC) is an intermodal translation task, where the system receives as an input an audio signal and outputs a textual description of the contents of the audio signal. In this chapter we will show an example of related works for the AAC task as well as our proposed architecture, we will be presenting two model. First model with one-hot-encoding as text feature, second model with sentence BERT as text feature.

Finally, we will show the details of how to two models work and represent each audio and text features that is used in both models.

2. Global architecture

In the past years this field has received increasing attention due to freely available datasets released and being held as a task in DCASE Challenges in 2020 and 2021 A number of papers have been published and the encoder-decoder framework has been adopted as a standard recipe for solving this translation task.

Encoder and decoder are the two components that make up our model. To thoroughly illustrate the reasoning behind the model. Encoder Decoder architecture is sequence-to-sequence model. a sequence-to-sequence model aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ[41].

In the figure 41 we gave an example of AAC system basic architecture, more details will be provided later in the chapter.

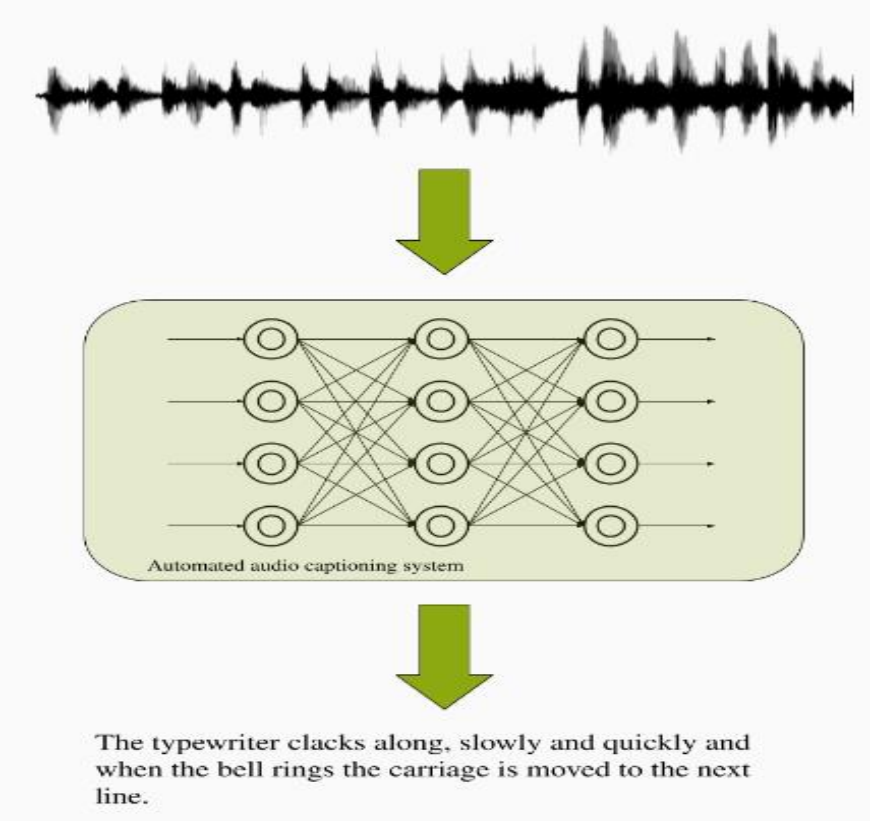

Figure 41: Example of Automated Captioning System [2]

3. Related works

Throw-out our research we have look into deferent architectures that were proposed for the automated audio captioning task, in the past 3 years the most frequently used architecture in DCASE Challenges 2020 and 2021 is an encoder decoder architecture that used an encoder part for encoding the audio and a decoding part for decoding text. The encoder decoder architectures have proven good results for AAC task, this next table 10 represents four deferent encoder decoder architecture that we have researched each using deferent methods to best solve the audio captioning problem.

	Encoder Neuron network	**Decoder Neuron Network**	**Audio Feature**	**Text Feature**	**Loss function**	**Learning set-up**
First Model	CNN	RNN	Log-Mel-Energies	Numeric representation	Cross-entropy	Adam

	Encoder Neuron network	Decoder Neuron Network	Audio Feature	Text Feature	Loss function	Learning set-up
Second Model	LSTM	RNN	Log-Mel-Energies	Embedding	Cross-entropy	Adam
Third Model	CNN	LSTM	Log-Mel-Energies	Embedding	Cross-entropy	Adam
Forth Model	RNN	Transformer	Log-Mel-Energies	One-hot-Encoding	Cross-entropy	Adam

Table 10- Related Works Table

In all these architectures we noticed that the most used architectures are RNNs and CNNs, they all seem to use the same audio feature which is Log-Mel-Energies but deferent text features wish tells us to experiment more with the text features instead of changing the audio features, later in this chapter we will be representing our two deferent features that we experimented with as our text features.

Because the AAC problem is a sequence-to-sequence problem we chose to experiment with two different model both using recurrent neuron network as encoders and decoder architecture both models will be explained later in the chapter.

4. Our proposed models

In this section we present the models architecture that we use in our experiments. The proposed method for the AAC task is a transformer model, which is based on the traditional sequence-to-sequence architecture. The model takes the log-Mel-spectrogram of an audio clip as input and outputs the probabilities of the predicted words.

For our 1rst model in order to train such a model with limited resources we experiment with 3 layered GRU as the encoder as represented in the next figure where (e) is the feature vector and (w) is the word. Our deep neural network method in the baseline model is a sequence-to-sequence system, consisting of an encoder and a decoder. The encoder takes as an input 64 log Mel-band energies, consists of three bi-directional RNN layers, and outputs the summary of the input sequence of features. Each GRU of the encoder has 256 output features.

The input sequence to the encoder has different length from the targeted output sequence. For that reason, there has to be some kind of alignment between these two sequences. Our system does not employ any alignment mechanism. Instead, the encoder outputs the summary vector of the input sequence, and this summary vector is then repeated as an input to the decoder. The decoder consists of two RNN layers it accepts the output of the encoder, and outputs a probability for each of the unique words. The decoder iterates for 22-time, which is the length of the longer caption, this figure 42 shows how it all works.

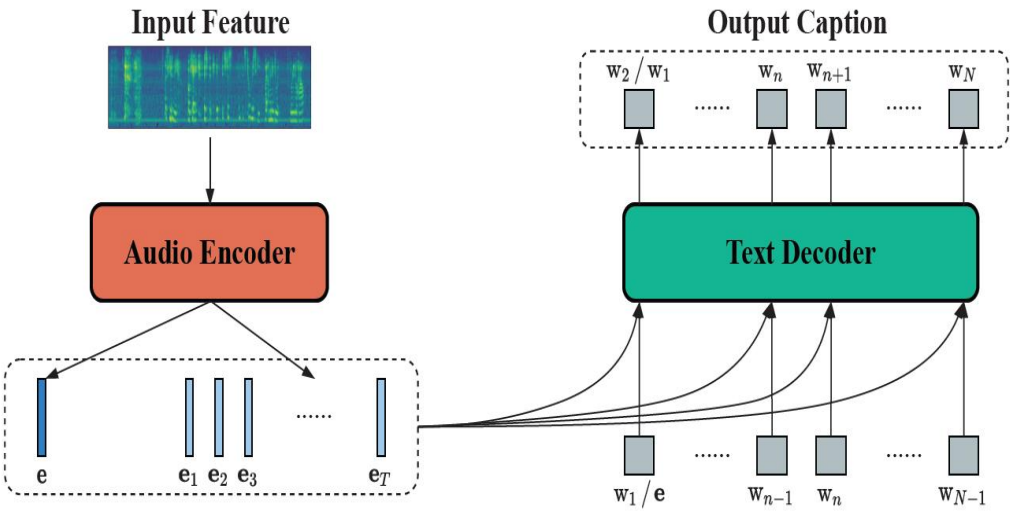

Figure 42: Illustration of encoder decoder AAC architecture[79]

The model we present here is trained to implement the following mapping where A is an audio clip and c is a caption:

$$f: \boldsymbol{A} \rightarrow \mathrm{c}$$

It is composed of an encoder with parameters θe and a decoder with parameters θd and it models the conditional probability distribution:

$$p_{\vartheta_e \vartheta_d}(\mathbf{c}|\mathbf{A})$$

The encoder part encodes the input sequence of patches P1: k to a new sequence X1: n, thus defining the mapping:

$$\mathrm{f: P1: k} \rightarrow \mathrm{X1: k}$$

where X1:h are the audio features. The audio features function as compressed representation of the input audio. This next figure 43 illustrates Our proposed encoder-decoder architecture.

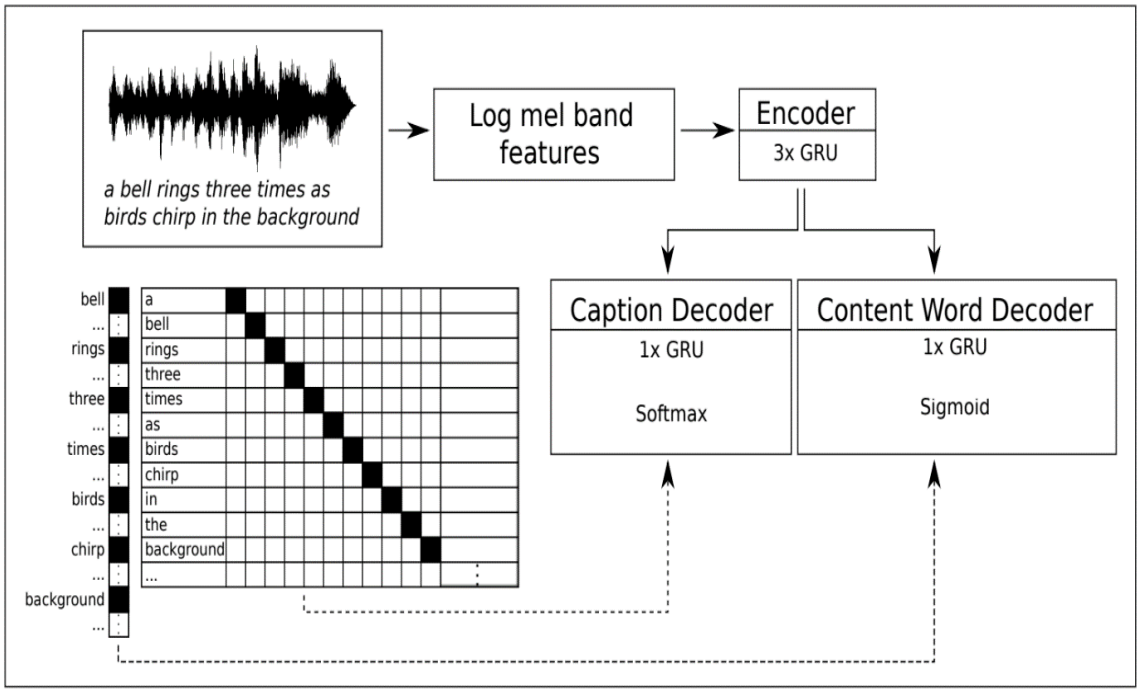

Figure 43: First model encoder decoder architecture

For our 2nd model, we used the same architecture represented in the figure 52 with the exception of changing the word embedding to sentence embedding specifically S-BERT. This time the model takes the same encoder decoder architecture but change the input of the decoder to a 384-dimensional vector, this proved a bit difficult to train and obtain good evaluation result. This next figure 44 represents the above explained model.

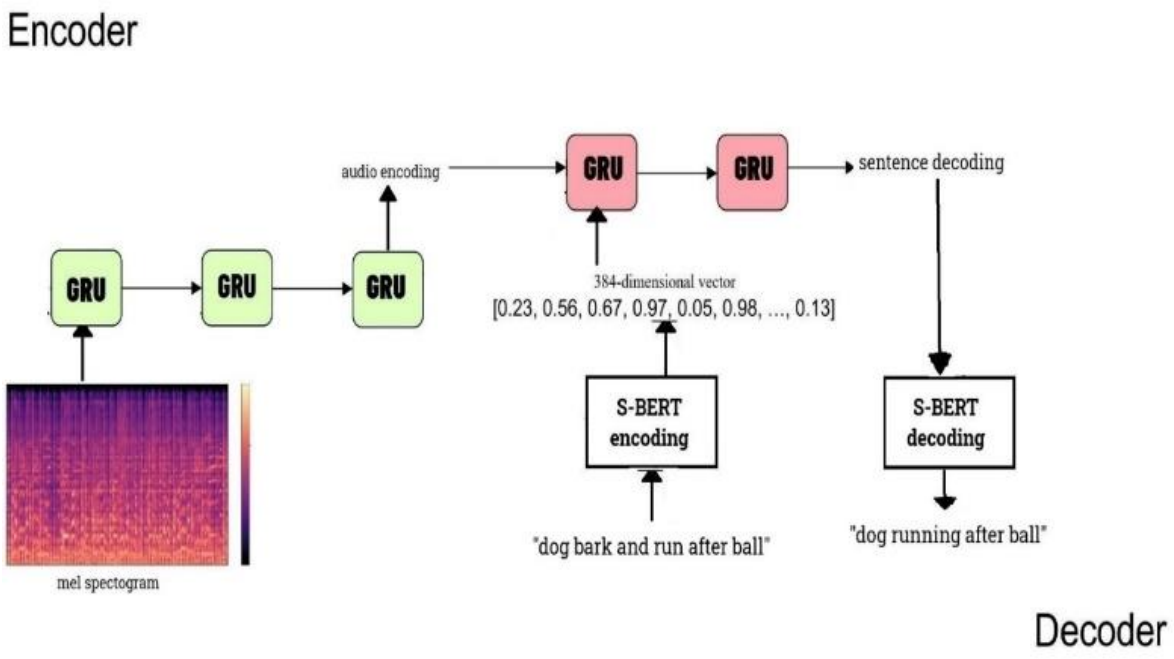

Figure 44: Second model with the use of S-BERT

5. Pre-processing

As mention in the previewed section the dataset contains 4982 audio samples, 15 to 30 seconds long, collected from the Freedsound platform. Each sample is annotated by human annotators with five different captions, 8 to 20 words in length, summing up to a total of 24905 captions in the whole dataset. There are three splits available: development (14465 captions, 2893 audios), evaluation (5225 captions, 1045 audios), and testing (5215 captions, 1025 audios), where only the development and evaluation splits are public and freely available[3].

For the audios, F = 64 log Mel-band energies are extracted using Hamming window of 1024 sampled-long window (around 23ms) with 50% overlap, resulting to $1292 \leq Ta \leq 2584$ feature vectors. For simplicity, no further pre-processing steps or data augmentation techniques are employed in this process. For captions processing, <SOS> (start-of-sentence) and <EOS> (end-of-sentence) tokens are appended at the start and end positions of the captions. The captions are then tokenized -mapped to a pre-built dictionary to from word vectors. Specifically, the dictionary associates each unique word in the development split to a unique number, in which all words are treated equally without any specific set of rules. Since the words are uniformly

distributed, there are no exclusive words in development or evaluation splits. For batch processing, all word vectors are concatenated into $Y \in RB \times L$, where B is the batch size, and L is the length of the longest caption in the batch and left-pad other shorter sentences with < SOS > tokens.

6. Encoder part

6.1. Audio encoding

Analyzing the content of an audio clip largely depends on obtaining an effective feature representation for it, which is the aim of the encoder in an AAC system. Current popular approaches for acoustic encoding consist of two steps. In our model we first extract acoustic features, and then passing them into an encoder to obtain compact audio features.

6.2. Audio features

Time-frequency representations, such as spectrograms, are widely used as the acoustic features. The spectrogram is a 2-D representation whose horizontal axis is time and vertical axis is frequency, the value at each point of the spectrogram represents the energy at a specific time and frequency.

6.3. RNNs

RNNs are designed to process sequential data with variable lengths. Audio is a time series signal, therefore RNNs initial works adopted RNN's. In a simple recipe, a RNN is used to model temporal relationships between acoustic features, and the hidden states of the last layer of the RNN is regarded as the audio feature sequence[36].

6.4. Audio pre-processing

Some audios are get recorded at a different rate-like 44KHz or 22KHz. Using LIBROSA, we can turn it into whatever form we would like and then as showed in the figure 45.

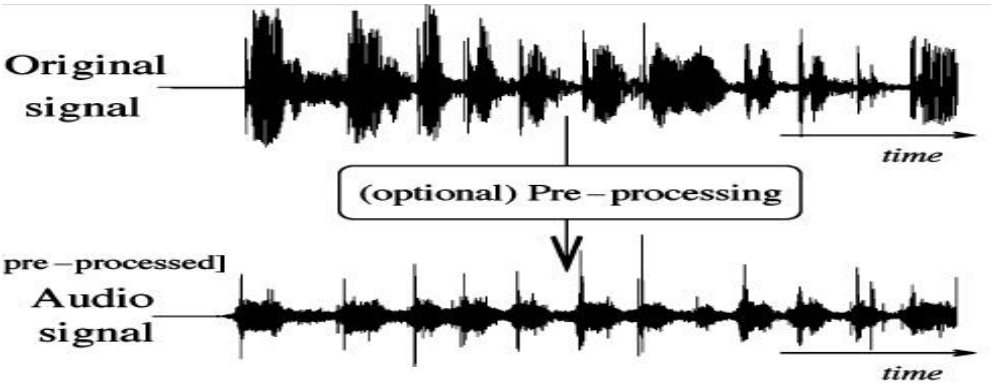

Figure 45: waveform pre-processing example

We can see the data in a normalized pattern. Now, our task is to extract some important information, and keep our data in the form of independent (and dependent features. We will use Mel spectrogram to extract independent features from audio signals. the audio data is pre-processed to enhance certain characteristics of the signal for analysis. This is achieved by reducing the effects of noise or by emphasizing the target sounds in the signal[55].

Clotho data are WAV and CSV files. In order to be used in our model, we had to extract features from the audio clips (WAV files) and the captions in the CSV files have to be turned to a more computational form. Finally, the extracted features and processed words, have to be matched and used as input-output pairs for optimizing the parameters of an audio captioning method. That where we chose to use Mel spectrogram as an input to our audio encoding method.

6.5.MEL-SPECTOGRAM

Deep learning models rarely take raw audio directly as input. In our work we converted the audio into a spectrogram. The spectrogram is a snapshot of an audio wave and since it can be translated to a sequence, it is well suited to being input to RNN-based architectures developed for handling sequenced data. Spectrograms are generated from sound signals using Fourier Transforms. A Fourier Transform decomposes the signal into its constituent frequencies and displays the amplitude of each frequency present in the signal.

A Spectrogram chops up the duration of the sound signal into smaller time segments and then applies the Fourier Transform to each segment, to determine the

frequencies contained in that segment. It then combines the Fourier Transforms for all those segments into a single plot.

It plots Frequency (y-axis) vs Time (x-axis) and uses different colors to indicate the Amplitude of each frequency. The brighter the color the higher the energy of the signal[60].

A Mel Spectrogram makes two important changes relative to a regular Spectrogram that it plots Frequency vs Time. It uses the Mel Scale instead of Frequency on the y-axis. For deep learning models, we usually use this rather than a simple Spectrogram.

This figure 46 below shows an example of our Spectrogram using the Mel Scale then using frequency.

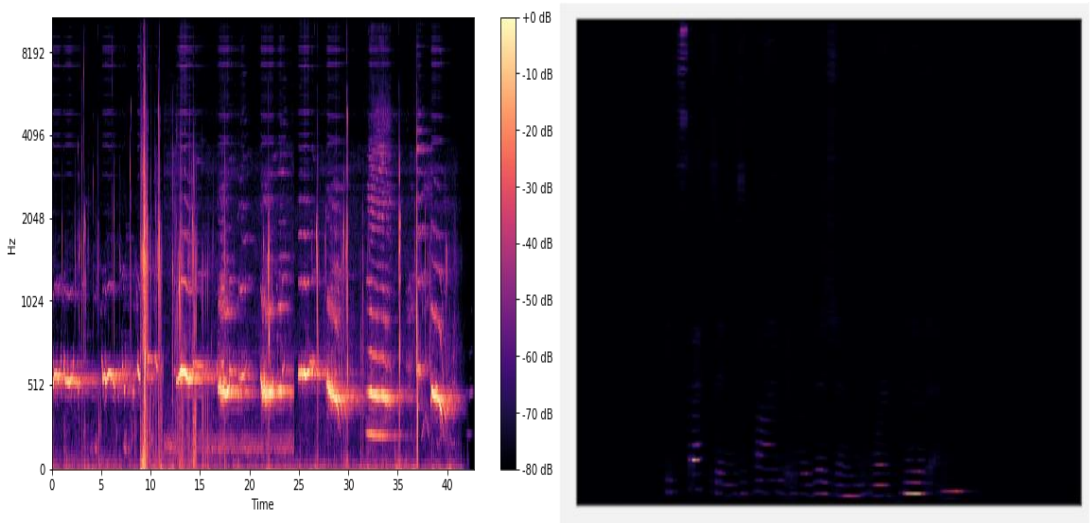

Figure 46: Example of a simple spectrogram and a male spectrogram

7. Decoder Part

7.1.Text decoding

The aim of the language decoder is to generate caption-given audio features from the encoder. All existing work we are aware of adopts an auto-regressive model, where each predicted word is conditioned on previous predictions. In addition to the main decoder block, there is often a word embedding layer before the main decoder block, which embeds each input word into a fixed-dimension vector. In this section,

we first introduce popular word embeddings and then discuss main text decoding approaches.

7.2. Word Embeddings

In natural language processing, word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning. This next figure 47 shows an example of how word embedding works when we introduce a full sentence as an input and how it extracts vectors from it.

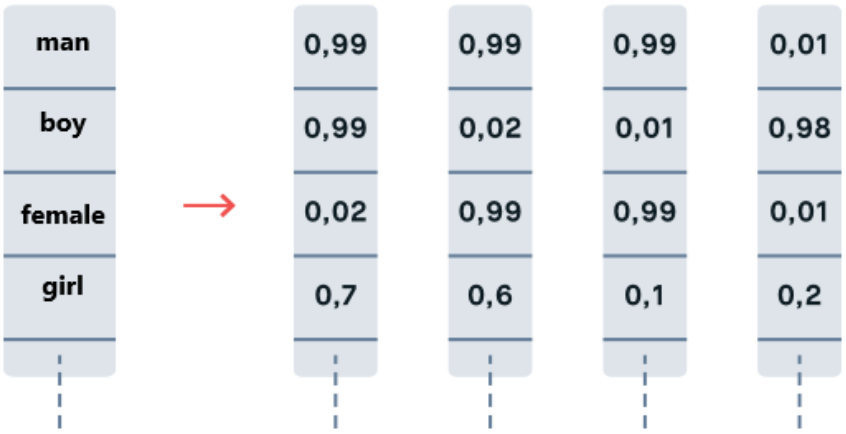

Figure 47: Word embedding example

Pre-trained word embeddings are trained using neural networks with a large corpus and could capture semantic information, that is, semantically similar words are close to each other in the embedding. Word2Vec GLOVE are widely used in existing audio captioning works. In our model we have chosen to use two deferent methods for text embedding to be used in our model.

7.3. One hot vector

The machine cannot understand words and therefore it needs numerical values so as to make it easier for the machine to process the data. To apply any type of algorithm to the data, we need to convert the categorical data to numbers. To achieve this, one hot ending is one way as it converts categorical variables to binary vectors. Suppose we have a sentence we want to get the one hot encoding vector for this, we

convert the text to lower case and then sort the words in ascending form A-Z. Now we'll have an alphabetically organized array filled with our words then we give each word a numerical label as we can see in the table 11 below.

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

Table 11- One hot vector example

This is not the most optimal method that can be used there is another method with Bert word embedding that gives a better representation of each word and the embedding of that word.

7.4.Sentence embedding

To have better results in our models we have also changed the text decoding part by switching from Word embedding to full sentence embedding. Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network.

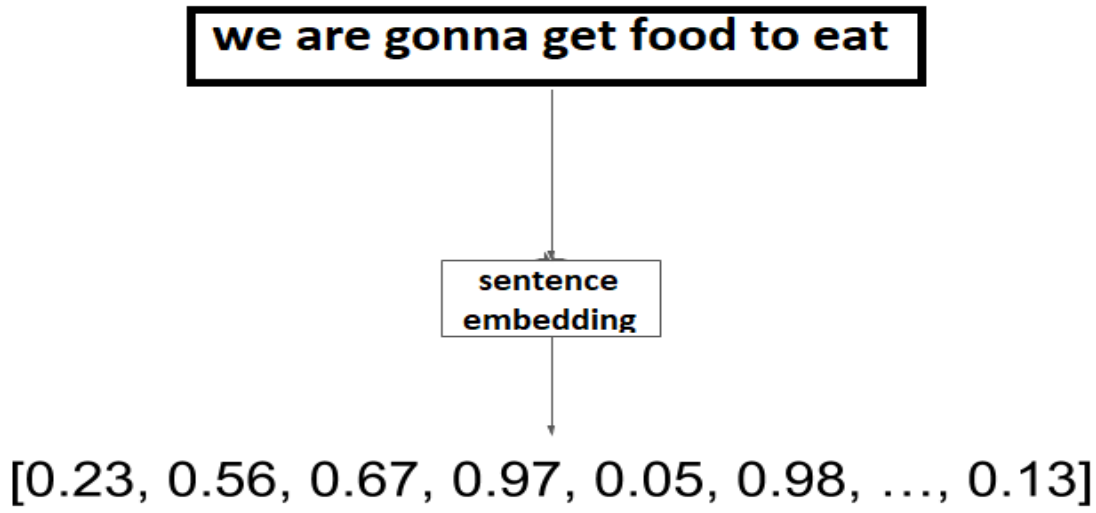

Figure 48:sentence embedding example

From this figure 48 we can see that similar to regular word embeddings, sentence embeddings embed a full sentence into a vector space.

7.5.Sentence BERT

S-BERT is a pre-trained transformer network, which set for various NLP tasks, including question answering, sentence classification, and sentence-pair regression. To derive sentence embeddings from BERT we pass single sentences through BERT and then derive a fixed sized vector by either averaging the outputs similarly to word embeddings or by using the output of the special CLS. Sentence embeddings are a well-studied area with dozens of proposed methods trains an encoder-decoder architecture to predict the surrounding sentences. In our work we have used sentence-transformers MiniLM-L6-v2 model. This is a sentence transform model, it maps sentences & paragraphs to a 384-dimensional dense vector space and can be used for tasks like AAC. This figure 49 illustrates an example of how SBERT works.

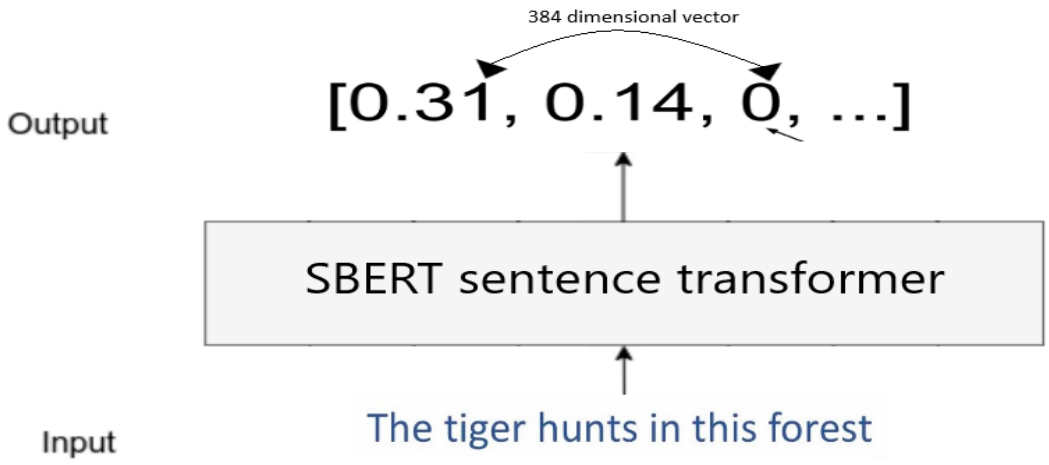

Figure 49: S-BERT sentence transformer example

BERT maps sentence to a vector space that is rather unsuitable to be used with common similarity measures like cosine-similarity. To overcome this shortcoming, we presented Sentence-BERT fine-tunes BERT in a triplet network architecture. where it could achieve a significant improvement over Word embeddings methods.

8. Conclusion

In this chapter, we presented the experimental models for the AAC TASK. At the beginning, we made a presentation of the global encoder decoder architecture and

then we show in detail how it works. Subsequently, we presented the inputs and outputs used in our experiments and the models we used.

We have presented the different steps necessary for extracting both the audio features and the text features that we have used and also showed the deferent methods that we have chosen to use in our model for pre-processing both type data audio and text. The Details of how transformers and Bert works are provided in a separate section in chapter 3. Our approach proves its interest by providing good solutions in a fairly reasonable time.

Chapter 4: Achievement

1. Introduction

This chapter presents the setup defined to conduct our experiments. In chapter 3 we present our approach method for the AAC task. Next, in chapter 4, we lay out the numerous tools that we have used in the development of our systems. we describe our implemented AAC system and present the features used to train the system as well as the model topology, we represent the deferent evaluation tools used to evaluate our models. Finally, we explain the procedure that we have followed in order to analyses and discuss our results we obtained and also, we compare our results. This chapter also describes the supported development tools and the programming language used. We end this chapter with a conclusion.

2. Used Tools

Deep learning research relies on exhaustive datasets and heavy computations during training which is generally time-consuming and resource-hungry. Thus, the use of parallel computing is necessary given that it considerably accelerates the training process[80]. For this purpose, Graphics Processing Units GPU is considered to be the leading parallel computing device used to conduct deep learning experiments. A GPU is an integrated single-chip processor, consisting of a highly parallel structure designed to perform extensive graphical and mathematical computations[81]. The structure of GPUs allows parallel computing through thousands of threads at a time hence giving this category of hardware the upper hand in deep learning executions[82].

However, the use of such hardware resources can be not only costly in terms of purchase and maintenance, but also risky if events such as over utilization or equipment depreciation occur, making the deep learning project very cost effective.

That is why we have chosen to use COLAB which is a product from Google Research. COLAB allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning[82].

3. Development Software

In this part, we will present the development tools that we used for the realization of our platform for AAC.

Python: Python is a general-purpose, interactive, object-oriented, high-level interpreted programming language. It was created by programmer 'Guido van Rossum' in 1991. Python's elegant syntax and dynamic typing, along with its interpreted nature, make Python an ideal language for rapid application development in many fields. Python is a free language placed under the PSFL license Python Software Foundation License, which can be used in many contexts and can be adapted to any type of use thanks to specialized libraries[83].

IDE PYCHARM: PyCharm is a dedicated Python integrated development environment IDE that provides a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It offers intelligent code entry, code inspections, error highlighting, and quick fixes[84].

QT Designer: Qt Designer is Qt's tool for designing and building GUI graphical user interfaces with Qt Widgets. You can compose and customize your windows or dialogs, and test them using different styles and resolutions. Widgets and forms created with Qt Designer integrate seamlessly with programmed code, using Qt's signals and slots mechanism, so you can easily assign behavior to graphical elements. All properties defined in Qt Designer can be changed dynamically in code. Additionally, features like widget promotion and custom plugins allow you to use your own components with Qt Designer[85].

Google Drive Storage: Regarding the storage of our data, we have chosen Google Drive. It is an online storage, synchronization and sharing service that offers 15GB of free space, and several packages for different storage spaces. This way, the dataset is easier to access and load in the chosen runtime environment which is Google Collaboratory for this we chose to porches the 5 £ package which we have been paying

for the last 7 months since the start of our project this offers us 200 gigabytes of storage to work with the only problem, we have faced is that we need to keep buying this package every month or all our data would be lost.

Google Collaboratory: Google Collab or is a cloud service, offered by Google, based on JUPYTER Notebook and intended for training and research in machine learning[86]. This platform makes it possible to train Machine Learning models directly in the cloud. Without therefore needing to install anything on our computer except a browser. The free package provides fully configured runtime for deep learning using Python and free access to Tensor Processing Unit (TPU); which offers up to 35 GB of RAM and 107 GB of disk space, and a TESLA k80 GPU. We have made use of the provided TPU to perform audio feature extraction taking into account that this process is costly in terms of RAM, while we have performed training of our models using the provided GPU[87][88]. After testing with the free collab we discovered that the amount off ram and space offered was not effective that's why we chose to update that further robust resources by upgrading to a professional version of Google Collaboratory. This upgrade guarantees priority access to highly powerful GPUs such as Tesla T4 and Tesla P100 and provides additional disk space and RAM capacity this cost 10 £ for every month (7 months), which we have been purchasing each time we wanted to test the model or do another evaluation.

Utility Libraries: Another bright spot for Google Collaboratory is the availability of all the necessary python libraries used for audio processing and deep learning experiments[88]. These libraries do not require any installation or configuration. The following are a few of the most relevant libraries that we have used in our work.

LIBROSA: LIBROSA is a Python package for audio and music signal analysis and processing[89]. It provides implementations of a variety of common functions that fall into four categories that are audio and time-series operations, spectrogram calculation, time and frequency conversion, and pitch operations. These functions are heavily used throughout our experiments.

PYTORCH: PYTORCH is an open-source library developed by Facebook that performs instantaneous dynamic tensor computations with automatic differentiation

and GPU acceleration, while maintaining performance comparable to the fastest modern libraries for deep learning[90].

Sentence Transformers: Sentence Transformers is a Python framework for state-of-the-art sentence, text and image embeddings. The initial work is described in chapter 3 Sentence-BERT Sentence Embeddings using Siamese BERT-Networks. this framework can be used to compute sentences or text embeddings for more than 100 languages.

In addition to the above mentioned deep learning libraries we have made use of the NUMPY library to perform manipulation operations on our data, the MATPLOTLIB library for plotting and graphical representations, the PICKLE module for serialization of python objects for storing purposes and the PYTHON package to store our trained models for testing. Table 12 provides additional information about the libraries we have used in our work.

UTILITY LIBRARY	VERSION
PYTHON	**3.9**
LIBROSA	**0.7.1**
LOGURU	**0.3.2**
PYYAML	**5.4.0**
PYTORCH	**1.3.1**
MATPLOTLIB	**2.0.2**
NUMPY	**1.17.4**
PICKLE	**5 0.0.12**

Table 12- Utility libraries used for deep learning

4. Data acquisition procedure

4.1. Audio Dataset for Audio Captioning

Clotho is a freely available audio captioning dataset, is an extension of the original Clotho dataset v1 and consists of audio samples of 15 to 30 seconds duration, each audio sample having five captions of eight to 20 words length. There is a total of 6974 (4981 from version 1 and 1993 from v2) audio samples in Clotho, with 34 870 captions (6974 audio samples * 5 captions per each sample). All audio samples are from the Free sound platform, and captions are crowdsourced using Amazon Mechanical Turk and annotators from English speaking countries. Unique words, named entities, and speech transcription are removed with post-processing[3].

Clotho v2 has a total of around 4500 words and is divided in four splits: development, validation, evaluation, and testing. Audio samples are publicly available for all four splits, but captions are publicly available only for the development, validation, and evaluation splits. There are no overlapping audio samples between the four different splits and there is no word that appears in the evaluation, validation, or testing splits, and not appearing in the development split. Also, there is no word that appears in the development split and not appearing at least in one of the other three splits. All words appear proportionally between splits (the word distribution is kept similar across splits) 55% in the development, 15% in the and validation, 15% in the evaluation, and 15% in the testing split.

4.2. Audio samples in Clotho

They have durations ranging from 10s to 300s, no spelling errors in the first sentence of the description on Free sound, good quality (44.1kHz and 16-bit), and no tags on Free sound indicating sound effects, music or speech. Before extraction, all 12k files were normalized and the preceding and trailing silences were trimmed. The content of audio samples in Clotho greatly varies, ranging from ambiance in a forest (animal sounds, and crowd yelling to machines and engines operating or revving[3].

In the following figure 50 is the distribution of the duration of audio files in Clotho and similar distribution is expected in Clotho v2.

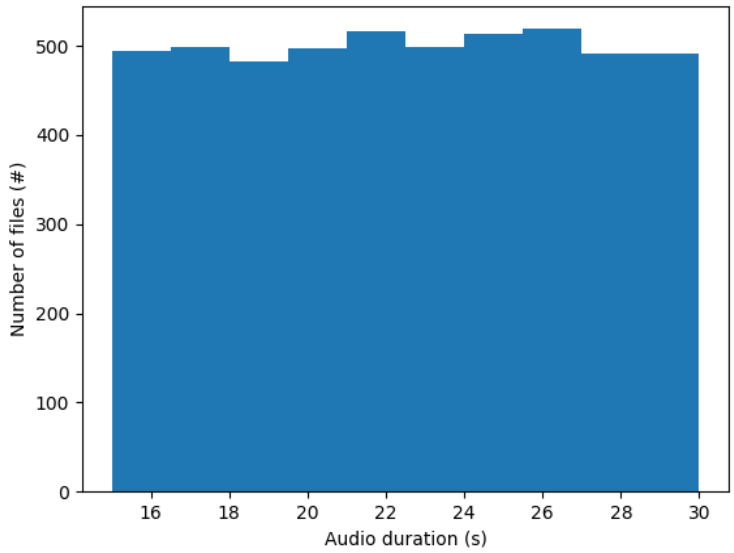

Figure 50:Audio duration distribution for Clotho dataset

4.3.Captions in Clotho

The captions in the Clotho dataset range from 8 to 20 words in length, and were gathered by a three-step framework. The three steps are:

a. Audio description,
b. Description Editing,
c. Description scoring.

In step 1, five initial captions were gathered for each audio clip from distinct annotators. In step 2, these initial captions were edited to fix grammatical errors. Grammatically correct captions were instead rephrased, in order to acquire diverse captions for the same audio clip. In step 3, the initial and edited captions were scored based on accuracy how well the caption describes the audio clip. The initial and edited captions were scored by three distinct annotators. The scores were then summed together and the captions were sorted by the total accuracy score first, total fluency score second. The top five captions, after sorting, were selected as the final captions of the audio clip[3].

Then manually sanitized the final captions of the dataset by removing apostrophes, making compound words consistent, removing phrases describing the content of speech, and replacing named entities. We used in-house annotators to

replace transcribed speech in the captions. If the resulting caption were under 8 words, we attempt to find captions in the lower-scored captions. The same in-house annotators were used to also replace unique words that only appeared in the captions of one audio clip. Since audio clips are not shared between splits, if there are words that appear only in the captions of one audio clip, then these words will appear only in one split.

In the next section we have a better representation of how the data was split.

4.4.Data splits of Clotho dataset

Clotho was divided into a development split of 2893 audio clips with 14465 captions, an evaluation split of 1045 audio clips with 5225 captions, and a testing split of 1043 audio clips with 5215 captions. These splits are created by first constructing the sets of unique words of the captions of each audio clip. These sets of words are combined to form the bag of words of the whole dataset, from which we can derive the frequency of a given word. With the unique words of audio files as classes.

```
Clotho.v2.1
├─clotho_captions_development.csv
├─clotho_captions_validation.csv
├─clotho_captions_evaluation.csv
├─development
│ └─... (3839 wavs)
├─validation
│ └─... (1045 wavs)
└─evaluation
└─... (1045 wavs)
```

Table 13- Clotho data split

Table 13 shows how the Clotho data set was split to be used in our ACC task.

5. Evaluation Metrics

In this section, we present the standard evaluation metrics used to evaluate the quality of captions generated by AAC models. Many of these metrics were introduced and are also used to evaluate other tasks such as Machine Translation, Summarization, and Image Captioning.

The proposed method was evaluated according to the following metrics used in the machine translation and image captioning fields. These metrics were calculated

for 10 different training and testing runs for each run, the parameters of the neural network were re-initialized according to the initialization functions and the neural network was re-trained on the training split. Caption evaluation is performed using the tools provided by the organizer of this challenge. This table14 sum up the metrics used.

Metrics	**What's measuring**
BLEUN	Measures a modified n-gram precision.
ROUGEL	Measures a score based on the longest common subsequence.
METEOR	Measures a harmonic mean of weighted unigram precision and recall.
CIDER	measures a weighted cosine similarity of n-grams.
SPICE	Measures the F-score of semantic propositions extracted from caption and reference.
SPIDER	Measures the arithmetic mean between the SPICE score and the CIDER score.

Table 14-metrics used in the evaluation of the model

BLEU*:* BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text in many NLP tasks and was originally used to evaluate machine translation [91]. BLEU uses a modified form of precision to compare a candidate text against multiple reference texts. The metric calculates the precision for n-grams. To calculate precision, the matching words in the actual sentence and the predicted sentence are calculated. BLEU does not consider the context of the word in the sentence. The metric range is between [0,1]. If the actual sentence and the predicted sentence are totally the same, then the score is 1. BLEU-1 (B-1) represents 1-gram, whereas BLEU-4 (B-4) represents 4-grams.

METEOR*:* METEOR [92] unlike BLEU incorporates both precision and recall in the evaluation score. The algorithm has two stages. First given a ground truth and a predicted sentence METEOR creates an alignment between them i.e., a mapping between unigrams, such that every unigram in each string maps to zero or one unigram in the other string. If there are two alignments with the same number of mappings, the alignment is chosen with the fewest crosses, that is, with fewer intersections of two mappings. Then METEOR calculates unigram recall and unigram precision together and takes a harmonic mean score. Finally, the harmonic mean score is multiplied with a penalty calculated as follows: The first fewest possible number of chunks is calculated such that the unigrams in each chunk are in adjacent positions in the system translation, and are also mapped to unigrams that are in adjacent positions in the reference translation. The penalty is then computed by the following.

$$Penalty = 0.5 \times \frac{\#chunks}{\#unigrams\_matched}$$

The penalty increases as the number of chunks increases to a maximum of 0.5. As the number of chunks goes to 1, penalty decreases, and its lower bound is decided by the number of unigrams matched. Finally, the METEOR Score for the given alignment is computed as follows:

$$Score = Fmean \times (1 - Penalty)$$

ROUGEL*:* ROUGE-L measures the longest common subsequence (LCS) between the ground truth and reference sentence. The idea is that a longer shared sequence would indicate more similarity between the two sequences. Then recall and precision calculations are applied as follows:

$$R_{LCS} = \frac{LCS(X,Y)}{m}$$

$$P_{LCS} = \frac{LCS(X,Y)}{n}$$

Where m is the length of the reference sentence and n is the length of the ground truth sentence. Then finally *ROUGEL* is calculated as:

$$F_{ROUGEl} = \frac{(1+b^2)R_{LCS}P_{LCS}}{R_{LCS} + b^2 P_{LCS}}$$

Where b = *PLCS/RLCS*. ROUGE-L is 1 when the sentences are the same, while ROUGE-L is zero when LCS (X, Y) = 0, i.e., there is no common sub-sequence in the sentences.

CIDER: Consensus-based Image Description Evaluation (**CIDER**) is a new paradigm for the evaluation of image captions that are based on human consensus [93]. It aims to capture sentence similarity, grammatically, importance, saliency and accuracy. To evaluate how well a generated caption ci matches the consensus of a set of captions Si = si1, ..., sim, all words are first mapped to their stem forms and each caption is represented using the set of n-grams ωk, that are present in it. Then, a Term Frequency Inverse Document Frequency (TF-IDF) weighting is performed for each n-gram to encode how often n-grams in the generated caption are present in the reference ones, and how often n-grams not present in the reference captions are not in the generated captions. Additionally, frequent n-grams are given low weight. The TF-IDF gk(sij) for each n-gram ωk is:

$$g_k(s_{i,j}) = \frac{h_k(s_y)}{\Sigma_{\omega \epsilon \Omega} h_l(s_y)} \log\left(\frac{|S|}{\Sigma_{S_p \epsilon S} \min\left(1, \Sigma_q h_k(S_{qp})\right)}\right)$$

where h_k (c) is the frequency that an n-gram k occurs in the caption c, Ω is the vocabulary of n-grams and S is the set of all samples. The $CIDEr_n$ score for n-length n-grams is the average cosine similarity between the generated caption and the reference captions and is given by:

$$CIDEr_n(ci.Si) = \frac{1}{m} \sum_j \frac{\mathbf{g}^n(ci).\mathbf{g}^n(Sy)}{||\mathbf{g}^n(ci)||||\mathbf{g}^n(Sy)||}$$

where $\mathbf{g}^n$ (*ci)* is a vector with elements *gk*(*ci*) that correspond to all n-length n-grams. The final CIDER score combines the scores of variable length n-grams as follows:

$$CIDEr_n(ci, Si) = \sum_{n=1}^{N} w_n CIDEr_n(ci, Si)$$

SPICE: All the evaluation metrics mentioned above are primarily sensitive to n-gram overlap. However, n-gram overlap is neither necessary nor sufficient for two

sentences to convey the same meaning. If we take for consideration the following example, we observe that it produces a high similarity score to all of the above metrics:

• A dog is standing on top of a chair

• A woman is standing on top of a field

These two sentences describe very different events but would get a fairly good similarity score with the all of the above metrics due to the phrase **is standing on top of a** which is common in both sentences.

Spice addresses this issue with the following procedure. At first, the generated caption c and the reference captions $S = s1, ..., sm$ are transformed to the scene graphs $G(c)$ and $G(S)$ respectively, where $G(S)$ is the union of scene graphs $G(si)$ for $si \in S$. The semantic relations in a scene graph are considered to be a conjunction of logical propositions or tuples and the function T returns these tuples from a scene graph as:

$$T(G(c)) \triangleq O(c) \cup E(c) \cup K(c)$$

where O(c) is the set of object mentions in c, $E(c)$ is the set of hyper-edges that represent relations between objects and $K(c)$ is the set of attributes associated with objects. The precision P, recall R and SPICE score are defined as:

$$P(c,S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(c))|}$$

$$R(c,S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(S))|}$$

$$SPICE(c,S) = F_1(c,S) = \frac{2.P(c,S).R(c,S)}{P(c,S) + R(c,S)}$$

where the binary matching operator $\otimes$ returns the matching tuples in two scene graphs. Since SPICE is an F-score, it is easily interpretable and its range is between 0 and 1. Moreover, it can be applied equally to both small and large datasets.

SPIDER*:* SPIDER [94] is used as the official ranking metric in the automatic audio captioning task in DCASE Challenge. SPIDER is the average of SPICE and CIDER. The SPICE score ensures captions are semantically faithful to the content, while CIDER score ensures captions are syntactically fluent.

6. Experiments parameters

This section describes and discusses the experimental parameters and findings that we have obtained during our experiments. The main goal of our case study is to compare the performances of several AAC models, varying the mechanism for extracting features and the hyperparameters used for learning the models.

Parameter	Value
Batch size	32
Optimizer	Adam
Learning Rate	10^{-4}
Weight Decay	10^{-4}
Epochs	300

Table 15-Training hyperparameters for all experiments

The table 15 shows the configuration we used for the training of our models. We manage to use a batch size 4 times the mini-batch size by performing gradient accumulation. We use gradient witch solves the exploding gradient problem and smoothens the gradient landscape. The learning rate is linearly increased to 10^{-4} in the first five epochs using warm-up, which is then multiplied by 0.1 every 5 epochs. In order to conduct our experiments fast and more efficiently we use early stopping policy with a patience of 5 epochs. If spider on the validation set hasn't been unproved beyond a threshold $\tau = 0.05$ in the last 5 epochs we stop training.

7. Evaluation and result

This section describes and discusses the experimental results and findings that we have obtained during our experiments. The main goal of our case study is to compare the performances of our AAC systems, varying the mechanism for extracting features and the hyperparameters used for learning the models. Most importantly, Specifically, we have conducted our experiments on CLOTHO v2 dataset. We have trained our systems on Log-Mel Spectrogram features.

For evaluating these systems, we have used a version of the caption evaluation tools used for the MS COCO challenge. This version of the code has been updated in

order to be compliant with Python 3.6 and above and with the needs of the automated audio captioning task.

Furthermore, we have based our discussions on various statistical tests, our case study consists of 2 experiments:

Experiment 1 investigates the impact of the LOG_MEL_SPECTOGRAME as audio feature method and it effect on the evaluation results this figure below shows the result, we optioned for all our 6 metrics with using the pre-trained wights there for not doing any training on the model just creating the datasets splits and running the evaluation steps. The table 16 show the result we got from the 1rst experiment.

Metrics	**Score**
BLEU _1	0.4141
BLEU _2	0.0714
BLEU _3	0.0171
BLEU_4	0.0000
METEOR	0.0717
ROUGE_L	0.2635
CIDER	0.0238
SPICE	0.0055
SPIDER	0.146

Table 16- pre-trained wights evaluation results

2nd Experiment we examines the impact of the new hyperparameters and compares several metrics of evaluation that we obtained which gave us a better score after a 300 epoch of training wish lasted for approximately 12 hours this figure below demonstrates the result we obtained. This next table 17 shows the best result we obtained after several experiments of parameters changes.

Metrics	**Score**
BLEU _1	0.3966
BLEU _2	0.1402
BLEU _3	0.0662
BLEU_4	0.0000

Metrics	Score
METEOR	0.0821
ROUGE_L	0.2779
CIDER	0.02791
SPICE	0.0326
SPIDER	0.0559

Table 17- Post full training evaluation results

From this Table you can notice that all metrics got approximately 20 to 55% better result than with using the pre trained wight, all metrics got better result except the 1rst one blue_1 wish got slightly worst result.

For more understanding of the results we obtained, we have compared them to previse proposed architecture that was showed in chapter 3, this next table x is a comparison between our results and the previse architectures results.

Model	CNN + RNN	LSTM + RNN	RNN + RNN	RNN+ TRANSFORMER	First Model	Second Model
BLEU1	0.614	0.641	0.655	0.610	0.3966	0.4141
BLEU2	0.446	0.479	0.476	0.461	0.1402	0.0714
BLEU3	0.317	0.335	0.344	0.334	0.0662	0.0171
BLEU4	0.219	0.236	0.231	0.237	0.0000	0.0000
ROUGE_L	0.475	0.467	0.469	0.455	0.0821	0.0717
METEOR	0.203	0.221	0.229	0.206	0.2779	0.2635
CIDER	0.593	0.660	0.693	0.629	0.02791	0.0238
SPICE	0.114	0.195	0.168	0.144	0.0326	0.0055
SPIDER	0.369	0.414	0.426	0.386	0.0559	0.146

Table 18- Comparison between the result we obtained and related works results

For the results of our second model, we could not reach them because of the big amount of data that is involved on the training process. We could not finish the training of the model because we did not have enough hardware capabilities for such a task, therefore we could not achieve good scores that we can discuss or compare with related work scores.

In order to study these results and reveal significant differences, we have first created a desktop application that allows us to do faster experiments with our audio test files.

For further analysis of these results, we have compared also used google collab pro to make sure that we are getting the best result possible from our training. Because this model is expensive on the hardware collab pro allows us to get more Ram and a dedicated GPU to run our code with easy access to our data throw google drive.

This figure 51 below shows our chosen interface that we created to perform our experiments.

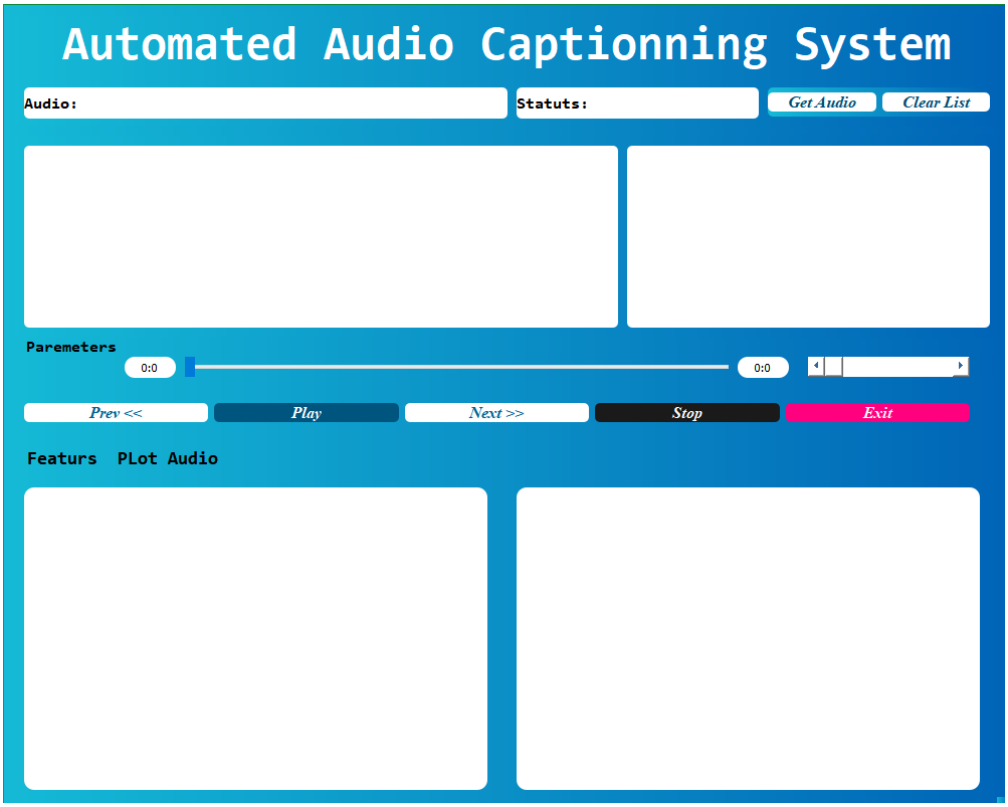

Figure 51: Desktop application interface

This application allows us to read any audio file slows us to play the audio, it also shows us the original caption of the file and after this file is processed and classified it shows us the predicted caption as shown in the figure 52 below.

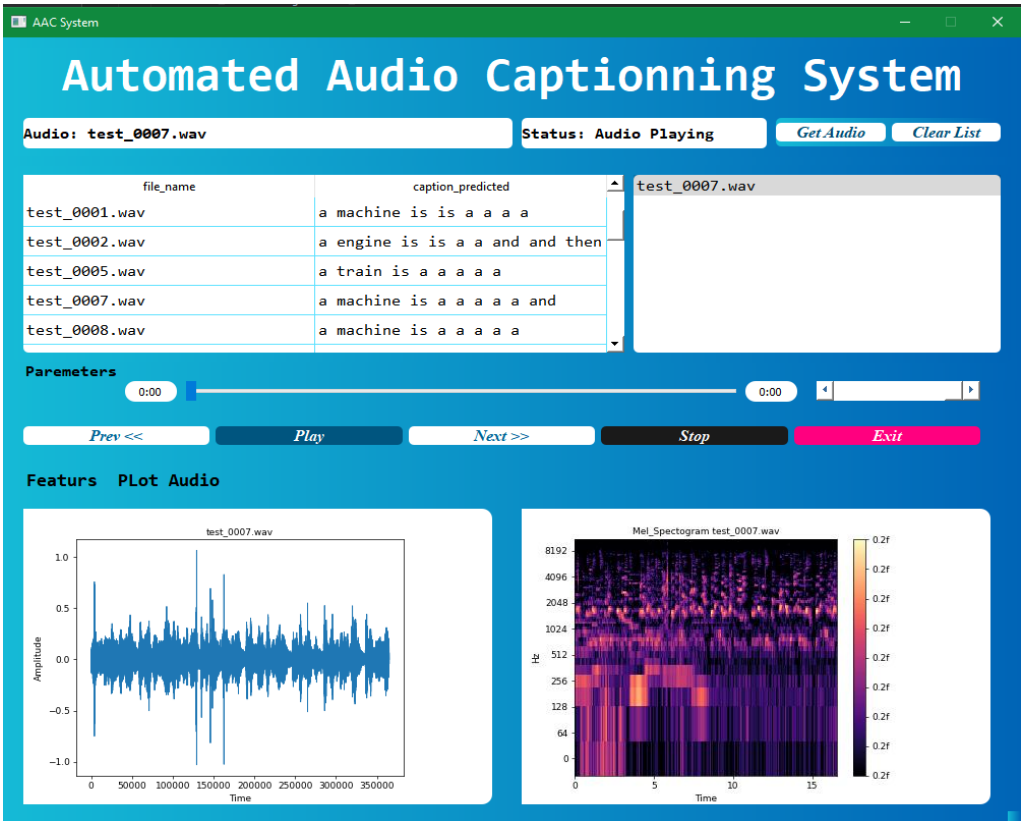

Figure 52:Desktop application interface example of execution

This next table 19 represent the results we obtained after running the test data part of Clotho with saved model wights, we did not obtain the most optimal results possible, but this is related to the dataset the bigger the dataset the better the results.

Test audio	Caption predicted
test_0087.wav	birds are chirping chirping birds birds and in
test_0940.wav	a person is a a a a a a
test_0646.wav	a person is walking a a a
test_0276.wav	a train is and a and and a a

Table 19-Post testing results

This table 17 represent the results we obtained after running the test data part of Clotho with saved model wights, we did not obtain the most optimal results possible, but this is related to the dataset the bigger the dataset the better the results.

It is also related the approach we selected to conducted this AAC system. From the result we obtained we can say that the AAC system works it gives a far description of the audio event that the audio describes but it does not give the perfect description. For example, if we have an audio test_0276.wav that presents a “sound of a train in a station” the AAC system recognizes the sound of the train but not the station or the people so it gives as a result “a train is and a and and a a “

This result would be more coherent in the desktop application that we have developed around our model, it takes an input an audio file from the testing part of CLOTHO dataset and it gives a result at what the audio event is, this figure below shows an example of an experiment we tried with audio file that has bird sound.

8. Conclusion

In this chapter, we presented the experimental results of the AAC task. At the beginning, we made a presentation on the development software and the programming languages used. Subsequently, we presented the dataset used in our experiments. We have presented the different steps necessary for the implementation of the proposed methods.

Conclusion

The main purpose of this project is to design an AAC system, where the system accepts as an input an audio signal and outputs the textual description of that signal. we have performed multiple experiments using a Clotho dataset. We implemented two deferent methods to construct our model the 1rst involved around word embedding using one-hot-encoding method witch we obtained good result with it.

The second method we changed the word embedding and replaced with sentence embedding specifically sentence BERT. In this method it proved rather hard to finish the training and get a proper wight to do evaluation with the model and get some proper result, the difficulties we faced was not having good hardware to run the entire code, the amount of ram we have purchased with google pro collab wasn't what we needed to do the full training this is due to the huge amount of data that we had to feed to the model at each epoch of the training data.

We also used a combination of feature representations based on audio processing techniques like Mel Spectrogram and text processing techniques used in text decoding from word embeddings like one-hot-encoding and BERT.

Automated audio captioning is a new and challenging task that involves different modalities. The main goal of our work was to design an audio captioning system Clotho dataset. The proper use of such information can considerably improve the captioning performance. our second goal was to test the impact of various embeddings methods and to perform a comparative study with using encoder-decoder architectures.

Based on the insights gained from the experimental findings in this AAC task we have learn new neural networks architectures, we learn new concepts on the natural league processing, audio feature handling which exited us to research more and try new experiments.

Our approach proves its interest by providing good solutions in a fairly reasonable time.

Future work

For Future work we would like to change the decoder to a CNN based architecture and measure the new performance of the model, integrate this model into a new platform for example a website, we would like to improve on our results and try new architectures that allow real time automated audio captioning.

Over all we would like to make this model as efficient as possible and make it less dependent on time and the type of hard ware a possible user might have, by implementing new methods for audio and text processing.

REFERENCES

[1] F. Lo, F. Su, S. Chen, J. Qiu, and J. Du, “Artificial Intelligence Aided Innovation Education Based on Multiple Intelligence,” in *2021 IEEE International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, 2021, pp. 12–15. doi: 10.1109/ICAIRC52191.2021.9544874.

[2] K. Drossos, S. Adavanne, and T. Virtanen, “Automated audio captioning with recurrent neural networks,” in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 374–378. doi: 10.1109/WASPAA.2017.8170058.

[3] K. Drossos, S. Lipping, and T. Virtanen, “Clotho: an Audio Captioning Dataset,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 736–740. doi: 10.1109/ICASSP40776.2020.9052990.

[4] K. Capek, *Rur rossum’s univ ersal robo ts*. 1920.

[5] L. von Rueden *et al.*, “Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems,” *IEEE Trans Knowl Data Eng*, p. 1, 2021, doi: 10.1109/TKDE.2021.3079836.

[6] H. I. Bulbul and Ö. Unsal, “Comparison of Classification Techniques used in Machine Learning as Applied on Vocational Guidance Data,” in *2011 10th International Conference on Machine Learning and Applications and Workshops*, 2011, vol. 2, pp. 298–301. doi: 10.1109/ICMLA.2011.49.

[7] R. & T. F. Bunker, “A Machine Learning Framework for Sport Result Prediction,” *Aplied Computing and Informatics*, 2017.

[8] M. Schwab, Ed., “Supervised Learning,” in *Encyclopedia of Cancer*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, p. 4398. doi: 10.1007/978-3-662-46875-3_102213.

[9] T. Chauhan, S. Rawat, S. Malik, and P. Singh, “Supervised and Unsupervised Machine Learning based Review on Diabetes Care,” in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2021, vol. 1, pp. 581–585. doi: 10.1109/ICACCS51430.2021.9442021.

[10] “https://towardsdatascience.com/supervised-vs-unsupervised-learning-in-2-minutes-72dad148f242”.

[11] “https://towardsdatascience.com/machine-learning-for-beginners-d247a9420dab”.

[12] “https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d”.

[13] “https://learnai1.home.blog/2019/12/13/classification-in-machine-learning/”.

[14] “https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d”.

[15] “Unsupervised Learning,” in *Encyclopedic Reference of Genomics and Proteomics in Molecular Medicine*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 1972. doi: 10.1007/3-540-29623-9_9085.

[16] A. Y. Al-Omary and M. S. Jamil, “A new approach of clustering based machine-learning algorithm,” *Knowl Based Syst*, vol. 19, no. 4, pp. 248–258, 2006, doi: https://doi.org/10.1016/j.knosys.2005.10.011.

[17] “https://www.powerbitraining.com.au/clustering-using-tables-in-power-bi/”.

[18] Z. & C. X. & Z. Y. & Q. L. & Y. R. & P. G. & Z. H. Zhou, “Brain Network Construction and Classification Toolbox ,” 2019.

[19] jaden wu, *Reinforcement learning Machine Learning.* 1998.

[20] Y. Guo, X. Wu, and X. Shu, “Data Acquisition and Preparation for Dual-Reference Deep Learning of Image Super-Resolution,” *IEEE Transactions on Image Processing*, vol. 31, pp. 4393–4404, 2022, doi: 10.1109/TIP.2022.3184819.

[21] D. Goularas and S. Kamis, “Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data,” in *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, 2019, pp. 12–17. doi: 10.1109/Deep-ML.2019.00011.

[22] D. E. Zomahoun, “A Semantic Collaborative Clustering Approach Based on Confusion Matrix,” in *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 2019, pp. 688–692. doi: 10.1109/SITIS.2019.00112.

[23] V. N. Mandhala, D. Bhattacharyya, and D. Midhunchakkaravarthy, “Need of Mitigating Bias in the Datasets using Machine Learning Algorithms,” in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, 2022, pp. 1–7. doi: 10.1109/ACCAI53970.2022.9752643.

[24] S. Sehra, D. Flores, and G. D. Montañez, “Undecidability of Underfitting in Learning Algorithms,” in *2021 2nd International Conference on Computing and Data Science (CDS)*, 2021, pp. 591–594. doi: 10.1109/CDS52072.2021.00107.

[25] M. P. Ranjit, G. Ganapathy, K. Sridhar, and V. Arumugham, “Efficient Deep Learning Hyperparameter Tuning Using Cloud Infrastructure: Intelligent Distributed Hyperparameter Tuning with Bayesian Optimization in the Cloud,” in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 2019, pp. 520–522. doi: 10.1109/CLOUD.2019.00097.

[26] Warren S McCulloch and Walter Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull Math Biophys*, pp. 115–133, 1943.

[27] “https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc”.

[28] Z. Hong, “A preliminary study on artificial neural network,” in *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference*, 2011, vol. 2, pp. 336–338. doi: 10.1109/ITAIC.2011.6030344.

[29] P. de Chazal, J. Tapson, and A. van Schaik, “A comparison of extreme learning machines and back-propagation trained feed-forward networks processing the mnist database,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 2165–2168. doi: 10.1109/ICASSP.2015.7178354.

[30] A. A. Karcioğlu and H. Bulut, “Performance Evaluation of Classification Algorithms Using Hyperparameter Optimization,” in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, 2021, pp. 354–358. doi: 10.1109/UBMK52708.2021.9559003.

[31] J. Xie, Z. Ma, G. Zhang, J.-H. Xue, Z.-H. Tan, and J. Guo, “Soft Dropout And Its Variational Bayes Approximation,” in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019, pp. 1–6. doi: 10.1109/MLSP.2019.8918818.

[32] A. Kumar, V. Bali, and S. Pandey, “Improving the efficiency of Plant-Leaf Disease detection using Convolutional Neural Network optimizer-Adam Algorithm,” in *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2022, pp. 362–367. doi: 10.1109/Confluence52989.2022.9734132.

[33] J. Konar, P. Khandelwal, and R. Tripathi, “Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network,” in *2020 IEEE International Students’ Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–5. doi: 10.1109/SCEECS48394.2020.94.

[34] P. Samudre, P. Shende, and V. Jaiswal, “Optimizing Performance of Convolutional Neural Network Using Computing Technique,” in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 2019, pp. 1–4. doi: 10.1109/I2CT45611.2019.9033876.

[35] “https://www.upgrad.com/blog/basic-cnn-architecture/”.

[36] W. Yang, D. Zhang, and Y. Fu, “Research of a Diagonal Recurrent Neural Network and Artificial Neural Networks,” in *2016 International Symposium on Computer, Consumer and Control (IS3C)*, 2016, pp. 374–377. doi: 10.1109/IS3C.2016.103.

[37] “https://towardsdatascience.com/growing-your-own-rnn-cell-simplified-b68ba2c0f082”.

[38] Md. E. Karim and S. Ahmed, “A Deep Learning-Based Approach for Stock Price Prediction Using Bidirectional Gated Recurrent Unit and Bidirectional Long Short Term Memory Model,” in *2021 2nd Global Conference for Advancement in Technology (GCAT)*, 2021, pp. 1–8. doi: 10.1109/GCAT52182.2021.9587895.

[39] “https://ikyathvarmadantuluri.gitbook.io/rnns/gated-recurrent-unit/architecture-of-gru”.

[40] “https://d2l.ai/chapter_recurrent-modern/gru.html”.

[41] S. Wu and Y. Wang, “Attention-based Encoder-Decoder Recurrent Neural Networks for HTTP Payload Anomaly Detection,” in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, 2021, pp. 1452–1459. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00196.

[42] P. Xiaosheng, W. Bo, Y. Fan, F. Gaofeng, W. Zheng, and C. Kai, “A Deep Learning Approach for Wind Power Prediction Based on Stacked Denoising Auto Encoders Optimized by Bat Algorithm,” in *2018 China International Conference on Electricity Distribution (CICED)*, 2018, pp. 945–948. doi: 10.1109/CICED.2018.8592384.

[43] E. Kavvousanos and V. Paliouras, “Optimizing Deep Learning Decoders for FPGA Implementation,” in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 2021, pp. 271–272. doi: 10.1109/FPL53798.2021.00053.

[44] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[45] “https://ledatascientist.com/a-la-decouverte-du-transformer/”.

[46] “http://keeganhin.es/blog/attn.html”.

[47] M. Abe, K. Fujii, Y. Nagata, T. Sone, and K. Kido, “Estimation of the waveform of a sound source by using an iterative technique with many sensors,” *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 1, pp. 24–35, 1998, doi: 10.1109/89.650307.

[48] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, “Deep learning for audio signal processing,” *IEEE J Sel Top Signal Process*, vol. 13, no. 2, pp. 206–219, 2019.

[49] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U. K. Laine, and J. Huopaniemi, “Frequency-warped signal processing for audio applications,” *Journal of the audio engineering society*, vol. 48, no. 11, pp. 1011–1031, 2000.

[50] U. Illindala, R. A. Grimm, A. Morehead, S. Chandra, N. L. Greenberg, and J. D. Thomas, “Automated analysis of transmitral and aortic Doppler velocity profiles using audio signal processing,” in *Computers in Cardiology 1996*, 1996, pp. 189–192.

[51] D.-J. Liu and C.-T. Lin, “Fundamental frequency estimation based on the joint time-frequency analysis of harmonic spectral structure,” *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 6, pp. 609–621, 2001.

[52] “https://www.teachmeaudio.com/recording/sound-reproduction”.

[53] U. Zölzer, *Digital audio signal processing*. John Wiley & Sons, 2022.

[54] “https://processing.org/tutorials/sound”.

[55] K. Nguyen, K. Drossos, and T. Virtanen, “Temporal sub-sampling of audio feature sequences for automated audio captioning,” *arXiv preprint arXiv:2007.02676*, 2020.

[56] I. N. Sneddon, *Fourier transforms*. Courier Corporation, 1995.

[57] U. Goswami *et al.*, “Amplitude envelope onsets and developmental dyslexia: A new hypothesis,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 16, pp. 10911–10916, 2002.

[58] R. Javaid, R. Besar, and F. S. Abas, “Performance evaluation of percent root mean square difference for ecg signals compression,” *Signal Processing: An International Journal (SPIJ)*, vol. 48, pp. 1–9, 2008.

[59] R. G. Bachu, S. Kopparthi, B. Adapa, and B. D. Barkana, “Voiced/unvoiced decision for speech signals based on zero-crossing rate and energy,” in *Advanced techniques in computing sciences and software engineering*, Springer, 2010, pp. 279–282.

[60] L. Wyse, “Audio spectrogram representations for processing with convolutional neural networks,” *arXiv preprint arXiv:1706.09559*, 2017.

[61] S. Umesh, L. Cohen, and D. Nelson, “Frequency warping and the Mel scale,” *IEEE Signal Process Lett*, vol. 9, no. 3, pp. 104–107, 2002.

[62] H. Meng, T. Yan, F. Yuan, and H. Wei, “Speech emotion recognition from 3D log-mel spectrograms with deep learning network,” *IEEE access*, vol. 7, pp. 125868–125881, 2019.

[63] B. Logan, “Mel frequency cepstral coefficients for music modeling,” 2000.

[64] Y. Kang, Z. Cai, C.-W. Tan, Q. Huang, and H. Liu, “Natural language processing (NLP) in management research: A literature review,” *Journal of Management Analytics*, vol. 7, no. 2, pp. 139–172, 2020.

[65] E. D. Liddy, “Natural language processing,” 2001.

[66] J. Levine, *Flex & Bison: Text Processing Tools*. “ O’Reilly Media, Inc.,” 2009.

[67] H. Liu, T. Christiansen, W. A. Baumgartner, and K. Verspoor, “BioLemmatizer: a lemmatization tool for morphological processing of biomedical text,” *J Biomed Semantics*, vol. 3, no. 1, pp. 1–29, 2012.

[68] C. Silva and B. Ribeiro, “The importance of stop word removal on recall values in text categorization,” in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, 2003, vol. 3, pp. 1661–1666.

[69] Y. Li and T. Yang, “Word embedding for understanding natural language: a survey,” in *Guide to big data applications*, Springer, 2018, pp. 83–104.

[70] P. Rodr\’\iguez, M. A. Bautista, J. Gonzalez, and S. Escalera, “Beyond one-hot encoding: Lower dimensional target embedding,” *Image Vis Comput*, vol. 75, pp. 21–31, 2018.

[71] E. M. Voorhees, “Natural language processing and information retrieval,” in *International summer school on information extraction*, 1999, pp. 32–48.

[72] K. W. Church, “Word2Vec,” *Nat Lang Eng*, vol. 23, no. 1, pp. 155–162, 2017.

[73] “https://openclassrooms.com/fr/courses/4470541-analysez-vos-donnees-textuelles/4855006-effectuez-des-plongements-de-mots-word-embeddings”.

[74] “Belkacem, Thiziri & Dkaki, Taoufiq & Moreno, Jose & Mohand, —. (2017). Apprentissage de représentations de documents et leur exploitation en recherche d’information.”.

[75] M. v Koroteev, “BERT: A review of applications in natural language processing and understanding,” *arXiv preprint arXiv:2103.11943*, 2021.

[76] “https://towardsml.wordpress.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/”.

[77] F. Sun *et al.*, “BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer,” in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.

[78] “https://medium.com/analytics-vidhya/how-to-fine-tune-bert-on-text-classification-task-723f82786f61”.

[79] X. Xu, M. Wu, and K. Yu, “A Comprehensive Survey of Automated Audio Captioning,” *arXiv preprint arXiv:2205.05357*, 2022.

[80] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli, “State-of-the-art in heterogeneous computing,” *Sci Program*, vol. 18, no. 1, pp. 1–33, 2010.

[81] B. S. H. Michel, “General-purpose gpu computing: practice and experience,” in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006, pp. 233–es.

[82] T. Carneiro, R. V. M. da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. de Albuquerque, and P. P. Reboucas Filho, “Performance analysis of google colaboratory as a tool for accelerating deep learning applications,” *IEEE Access*, vol. 6, pp. 61677–61685, 2018.

[83] Python, “Python,” *https://www.python.org/downloads/release/python-394/*, Apr. 04, 2021.

[84] Fonctionnalités de PyCharm, “https://www.jetbrains.com/help/pycharm/quick-start-guide.html.”

[85] Qt Designer Manual, “https://doc.qt.io/qt-5/qtdesigner-manual.html ,” 2022.

[86] Henri Michel, “https://ledatascientist.com/google-colab-le,” *Le guide Ultime*.

[87] T. Carneiro, R. V. M. da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. de Albuquerque, and P. P. Reboucas Filho, “Performance analysis of google colaboratory as a tool for accelerating deep learning applications,” *IEEE Access*, vol. 6, pp. 61677–61685, 2018.

[88] T. Carneiro, R. V. M. da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. de Albuquerque, and P. P. Reboucas Filho, “Performance analysis of google colaboratory as a tool for accelerating deep learning applications,” *IEEE Access*, vol. 6, pp. 61677–61685, 2018.

[89] B. Mcfee et al, “http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf,” *Librosa - audio processing Python library*, 2015.

[90] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.

[91] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[92] S. Banerjee and A. Lavie, “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments,” in *Proceedings of the acl workshop on*

intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65–72.

[93] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[94] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, “Optimization of image description metrics using policy gradient methods,” *arXiv preprint arXiv:1612.00370*, vol. 5, 2016.