PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF DJILALI BOUNAAMA KHEMIS MILIANA

FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

THESIS PRESENTED TO OBTAIN

THE MASTER DEGREE ON « COMPUTER SCIENCE »

OPTION: «SOFTWARE ENGINEERING»

TITLE:

Arabic automatic text summarization: A deep learning approach

The students:
CHERRANI TOUFIK
DILMI ABDELHAKIM

The jury composed of:

Academic year: 2019/2020.

Acknowledgements

First and foremost, we would like to thank God Almighty for giving us the strength,knowledge, ability and opportunity to undertake this research study and to persevereand complete it satisfactorily. Without his blessings, this achievement would not have been possible.

In our journey towards this degree, we have found a teacher, a friend, an inspiration, a role model and a pillar of support in our Guide, Mr. "Djamel Bahloul" . He has been there providing his heartfelt support and guidance at all times and has given as invaluable guidance, inspiration and suggestions in our quest for knowledge. He has given as all the freedom to pursue our research, while silently and non-obtrusively ensuring that I stay on course and do not deviate from the core of our research. Without his able guidance, this thesis would not have been possible and we shall eternally be grateful to him for his assistance.

Finally, We would like to express our sincere thanks to all teachers of mathematics and computer science and everyone who has been involved in this work from near and far.

ملخص

البيانات هي نفط القرن الحادي والعشرين ، والتحليلات هي محرك الاحتراق ، لذلك نحن محاطون بكمية ضخمة من البيانات التي يتم إنتاجها يوميا وستظل غير مفيدة للبشر ما لم يتم معالجتها بأدوات وتقنيات حديثة. حيث اصبح التلخيص الالي وسيلة مهمة للعثور بدقة على المعلومات المهمة من النصوص الكبيرة في وقت قصير وباقل جهد.بحث تصنف مناهج تلخيص النص إلى فئتين: الاستخراجية والتجريدية.في هذا العمل سوف نقوم بتجربة الشبكة العصبية المتكررة على كلا المنهجين وسنناقش مزاياها وعيوبها مع مقارنتها بالنماذج المتاحة.

الكلمات المفتاحية: شبكة العصبية المتكررة، تلخيص آلي للنص ، ملخص تجريدي ، ملخص استخلاصي ، نص عربي.

Abstract

Informations are the oil of the 21st century, and analytics is the combustion engine.Therefore,we are surrounded by the huge amount of data that is produced every day and will remain unuseful for humans unless making it available with new tools and technologies.Automatic text summarization has becomes an important way of finding relevant information precisely in large text in a short time with little efforts. Text summarization approaches are classified into two categories: extractive and abstractive.in this work, we want to experiment the impact of RNN for both approches with Arabic text. We will discuss its advantages and shortcomings in comparison with the available models.

Key words: RNN , Automatic text summarization , Abstractive summary , Extractive summary , Arabic text .

Résumé

L'information est le pétrole du 21e siècle, l'analytique est le moteur à combustion. Nous sommes donc entourés par l'énorme quantité de données qui sont produites chaque jour et qui resteront inutilisables pour l'homme si elles ne sont pas rendues disponibles grâce aux nouveaux outils et aux nouvelles technologies. Le résumé automatique de texte est devenu un moyen important de trouver des informations pertinentes précisément dans un grand texte en peu de temps et avec peu d'efforts. Les approches de résumé de texte sont classées en deux catégories : extractives et abstractives.Dans ce travail, nous voulons expérimenter l'impact de la RNN pour les deux approches avec des textes Arabes. Nous discuterons de ses avantages et de ses inconvénients en comparaison avec les modèles disponibles.

Mots clés:RNN, résumé automatique du texte,résumé abstrait, résumé extractif, texte arabe.

Contents

List of Figures

List of Tables

List of Abbreviations

ATS	Automatic Text Summarization
TF	Term Frequency
IDF	Inverted Document Frequency
HMM	Hidden Markov Models
ANN	Artificial Neural Networks
NLP	Natural language processing
ANLP	Arabic natural language processing
CA	Classical Arabic
MSA	Modern Standard Arabic
ML	Machine learning
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
GEMS	Generative Modelling for Evaluation of Summaries
BLEU	Bilingual evaluation understudy
LCS	Longest Common Subsequence
RNN	Recurrent Neural Networks
AI	Artificial Intelligence
NN	Neural Network
CNN	Convolution Neural Networks
FRNN	Fully recurrent neural network
LSTM	Long Short-Term Memory
CTC	Connectionist Temporal Classification
GRU	Gated Recurrent Units
EASC	Essex Arabic Summaries Corpus
DUC	Document Understanding Conference

TAC	Text Analysis Conference
LCS	Longest Common Subsequence
BERT	Bidirectional Transformer

Introduction

The information age quickly revolutionizing the way transactions are completed and has changed our whole environment. Now We are surrounded by data, not in a metaphoric sense; it is a physical reality. Internet, television and radio signals are circulation in the air and we receive them by our mobile devices everywhere. Data servers in counts of millions have connected all the planet like a spider web. A considerable portion of this stream of new data is textual.it is cover almost most of the available text corpus, like books, journals and newspapers, it is also connected to millions of organizations, companies, universities and research centers, and individual users that are creating content everyday in the websites, blogs, social networks etc. These information is left unusable unless we find a way to make it available for users. Automatic Text Summarization is an attempt to decrease the size of a data while keeping its valuable content. The two primary methods of summarization can be classified into abstractive and extractive summarization. The abstractive summarization uses linguistic methods to examine and interpret the text to generate a new shorter summary that conveys important information from the original document.On the other hand an extractive summary, in contrast, is composed with a selection of sentences and text part, phrases, paragraphs from the source text. In general, topic identification, interpretation, summary generation, and evaluation of the generated summary are the key challenges in text summarization and even more challenging with arabic text.Around 1950's the resources were not sufficient to make an abstractive summarization systems due the difficulties to design and replicate an abstractive algorithm that is close to how a brain receive process and regenerate information .Up until recently neural network based models demonstrated that has the potential to be the ultimate accurate text summarizer if we make it learn from a human style as accurate as possible. as a result we adopted recurrent neural network based model on both the abstractive and extractive approaches for summarization task of Arabic text. This thesis is organized as follow : In the first chapter, we will present the important Components of automatic summarization along with natural language processing ,In the second chapter, we will explore the evolution of sequential deep neural models to the recent RNN models .In the third chapter, we implement extractive RNN model using sentences classification in conjunction with an abstractive RNN model .In the last chapter, we will compare the results with a other models to clearly explain the behavior of the main models on our summarization task

Chapter 1

Automatic text summarization

1.1 Introduction

Currently, we enjoy quick access to enormous amounts of information. However, most of this information is redundant, insignificant, and may not convey the intended meaning. Therefore, using automatic text summarizers capable of extracting useful information that leaves out inessential and insignificant data is becoming vital. Implementing summarization can enhance the readability of documents, reduce the time spent in researching for information, and allow for more information to be fitted in a particular area. In this chapter we present the different types and approaches used in the automatic summarization of texts, then we talk about the different techniques of natural language processing and the methodologies of it ,in the end we discussed how the evaluation of summary works.

1.2 Definition

A summary is a short text which contains the important information of the original text, and which is less than half of the original text and usually too less than that.Automatic text summarization: is the process of shortening a text document with software, in order to create a summary that represents the most important or relevant information within the original content, and it becomes critical when someone needs a quick and accurate summary of very long content if done manually. However, when this is done by means of a computer, we call it Automatic Text Summarization (ATS). Text summarization can broadly be divided into two categories:Extractive Summarization and Abstractive Summarization.

1.2.1 Extractive Summarization

This method relies on extracting several parts, such as phrases and sentences, from a piece of text and stack them together to create a summary, in another way, it identifies the important sentences or phrases from the original text and extract them to the final summary, Their are some techniques to identify the principal sentences and measuring their importance namely Topic Representation, and Indicator Representation [14]. In the Topic Representation technique, assumed that there are a given topic for the text corpus. The summarizer explores the corpus to find the words that are closer to the topic and measures their frequency to score the importance of the sentences.

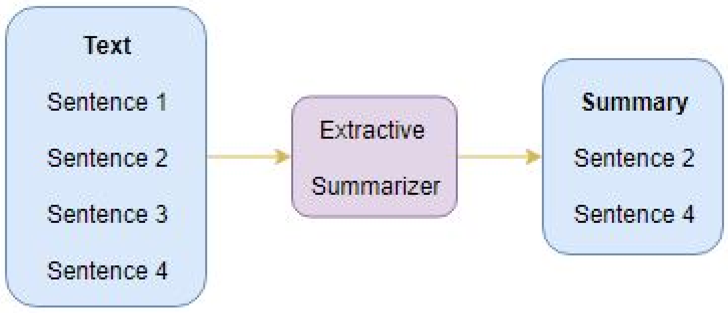


Figure 1.1: Text summarization by extraction.

1.2.2 Abstractive Summarization

Abstractive Summarization is the automatic production of summaries by abstraction, which comes from the field of artificial intelligence, the constitution of a summary by an application must go through the total or partial understanding of the text. For that reason, the abstractive summary is based on understanding and finding the most essential meaning of the original text and rewriting it in fewer words, Abstractive summarization is closer to what a human usually do. He conceives the text, compares it with his memory and related information, and then recreate its core in a brief text. That is why the abstractive summarization is more challenging than the extractive method.

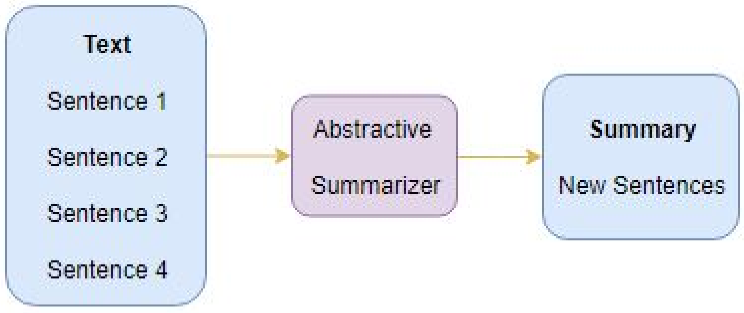


Figure 1.2: Text summarization by abstraction.

1.3 Automatic summarization approaches

Under the two methods cited above, several approaches were proposed where the most common are the following:

1.3.1 Surface level approaches

This approach looks at cue words and phrases like "in conclusion", "important", "in this paper" or complete sentences containing them which are then rearranged to form a coherent summary. It was developed in[15].The words are selected based on their term frequency (important sentences contain frequently appearing words), location (words and phrases in titles and headings are of relevance) and special words found in the original document.

1.3.2 Corpus based approaches

The main idea of the Corpus-Based approach is that instead of looking for term frequency using the original content, a corpus is made from similar contents and relevance of a word is calculated for example by the TF-IDF (Term Frequency -Inverted Document Frequency) formula, where tf is the frequency of the word in the document and idf is the inverted document frequency.

1.3.3 Cohesion based approaches

Surface level and Corpus-based approaches fail to account relations between sentences in a document. For example, in the sentence, "I told him to make the report", the pronoun “him” could be put into the summarized text without even mentioning the person being referred to which making it difficult to understand. Text cohesion identifies relations among of the document terms and those determining text connectivity. Lexical chains are used in this method [16]. They are a grammatically independent sequence of words that express the cohesive structure of the text. For example, a lexical chain can be presented as: [Rome $\rightarrow$ capital $\rightarrow$ city $\rightarrow$ inhabitant]. The lexical chains provide the solution to the problem faced in the previous summarization methods i.e. loss of context after generation of the summary.

1.3.4 Graph based approaches

With graph-based approach, the document is represented in the form of undirected graph. There is a node for every sentence. An edge between two nodes is drawn if there is a relation between these two nodes. A relation can be a cosine similarity above a threshold, or any other type of relationships.

After drawing a graph; it is possible to view the sub-graphs of connected nodes as a cluster of distinct topics covered in the document[17]. This yields two results: For query-specific summaries, sentences may be selected only from the pertinent sub-graph, while for generic summaries sentences would be selected from each sub graph for best coverage.[18]

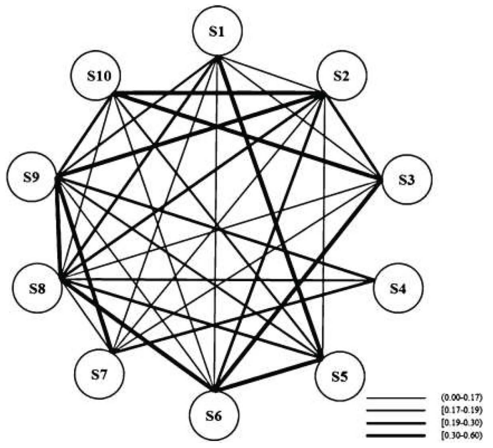


Figure 1.3: Text Summarization Graph-Based representation[1]

1.3.5 Machine learning based approaches

Machine learning approaches are based on machine learning algorithms in order to produce summaries. Machine learning approaches deal with the summarization process as a classification problem, sentences are classified to summary and non-summary sentences based on the features that they have. Hidden Markov Models (HMM)[19][20][21][22], and Bayesian[23] rule are examples on Machine learning summarization approaches, where set of training documents and their extractive summaries are given.

1.3.5.1 Naïve-Bayes method

Naïve-Bayes method was first used in [24] by using Bayesian classifier to determine if a sentence should be extracted or not. The system was able to learn from data. Some features used by their system include the presence of uppercase words, length of sentence, structure of phrase and position of words. The author assumed the following: s = a certain sentence, S = the sentences in the summary, and F1, Fk = the features.

$$P(s \in S|F_1, F_2, ..F_k) = \frac{\prod_{i=1}^{k} P(F_i|s \in S) * p(s \in S)}{\prod_{i=1}^{k} P(F_i)} \tag{1.1}$$

In equation (1.1)[24] , Sentences are scored based on these features and the formula is used to calculate the score, the highest ranking sentences are extracted.
Th naïve-bayes classifier was also used in DimSum [25], which used term frequency (tf) which is the

number of times that a word appears in a sentences and inverse document frequency (idf) which is the number of sentences in which a word occurs, to know words that hold point at the key concepts of a document.

1.3.5.2 Hidden Markov model

A hidden Markov model is a tool for denoting probability distributions over sequences of observations. If we represent the observation at time t by the variable Yt, we assume that the observation are sampled at discrete, equally-spaced time intervals, so t can be an integer-valued time index. The two defining properties of hidden Markov model are: the assumption that the observation at time t was generated by some process whose state St is hidden from the observer and the assumption that the state of the hidden process satisfies the Markov property i.e. given the value of St-1, the current state St is independent of all the states prior to t-1[26].

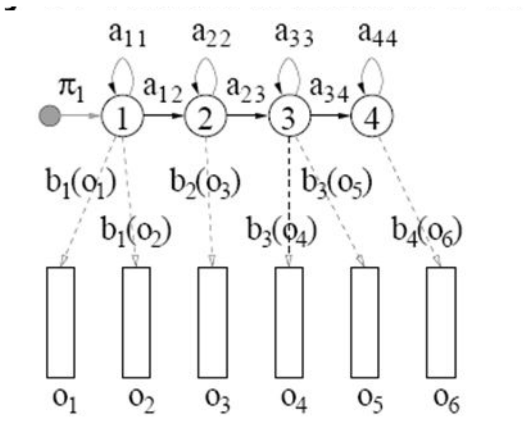


Figure 1.4: Summary of Hidden Markov models[2]

Two sentence reduction algorithms were proposed in [27]. Both were template-translation based which means that they don't need syntactic parser to represent the original sentences for reduction. One was founded on example-based machine-translation which does a good job of in the area of sentence reduction. On the other hand in specific cases, the computational complexity can be exponential. While the second one was an addition to the template-translation algorithm through the application of Hidden Markov model, the model employs the set of template rules that was learned from examples to overcome the problem of computational complexity.

1.3.5.3 Neural Network method

Artificial Neural Networks(ANN) are one of the most popular and powerful classes of machine learning algorithms. ANN is used to generate summaries of arbitrary length news articles. A neural network is trained on a corpus of articles. The neural network is then modified, through fusion

to produce a summary of the most ranked sentences of the article. Through feature fusion, the network discovers the importance of various features used to determine the Summary-worthiness of each sentence[28].
For example, the features are selected according to position of document or position of the sentence. In ANN architecture presented in Figure 1.5, the following seven features are used :

f1 = Paragraph follows title (Paragraph Position)
f2 = Paragraph location in document
f3 = Sentence location paragraph
f4 = First sentence in paragraph
f5 = Sentence Length
f6 = Number of thematic words in sentence
f7 = Number of title words in sentence

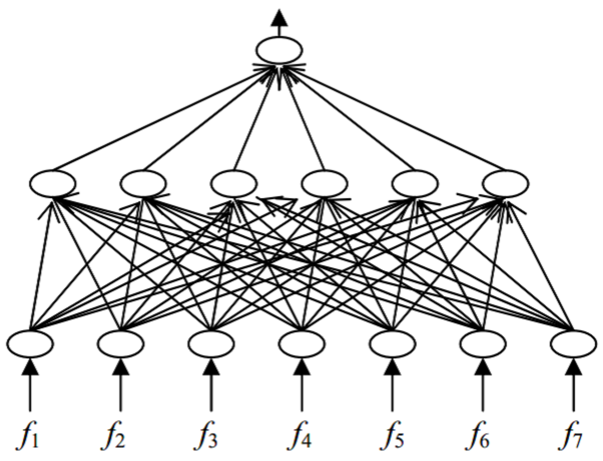


Figure 1.5: The Neural Network after Training[3]

1.4 Natural Language Processing

Natural language processing (NLP) is a field of computer science that seeks to create oncepts, find methods, and construct software that is able to comprehend, study, and produce natural human languages to enable human interaction with computers through writing and speech. In other words, NLP helps computers identify the ways in which humans use language.Over the last decade, Arabic has begun to gain ground in the area of research within NLP. Therefore, much works targeted different aspects related to how this language are processed such as Machine translation ,ATS. And how to present the characteristics of the Arabic language also to classify the works handling it.

1.4.1 Arabic Natural Language Processing (ANLP)

Arabic natural language processing (ANLP) has attracted many researchers after significant research has been carried out on English NLP and that of other languages. Many ANLP laboratories have been established. Recently, ANLP has received more attention, and several applications have been developed including text categorization, web page spam detection, and sentiment analysis [29][30]. However, developing ANLP tools requires additional efforts due to two principal difficulties: the combination of letters in the Arabic language and the removal of diacritics that represent the vowels [31].

1.4.2 Arabic language

Arabic is an Afro-Asiatic language that developed in the Middle East. More than 250 million individuals speak the Arabic language across the world [29]. The Arabic languagewas widely disseminated after the emergence of Islam,although it existed centuries before the religion. As a universalreligion, Islam has delivered the Arabic language to its followers, estimated to be 1.5 billion people [29]. Historically speaking, Arabic is rooted in Classical Arabic (CA), which has been used as the Arab peoples' native language since 600 AD. It is associated with Islam and the Quran. However, over the centuries, the languagehas evolved and been simplied to create what is known as Modern Standard Arabic (MSA). The terminology and the linguistic features of MSA differ from those of CA, but the structure of words and sentences have remained. In addition to CA and MSA, each region has a dialect of Arabic spoken in the community (between friends and family) [32].

1.4.3 ANLP and machine learning

Recently, ANLP tools are developed using machine learning algorithms. Machine learning (ML) falls within the domain of artificial intelligence. The goal of such technologies is to enable computers to learn without explicit programming. ML has been successfully applied in many difficult and complex computing tasks (such as ANLP) without designing and programming explicit algorithms. In addition, the range of ML algorithms that yield satisfactory results has led ML to become vastly involved in ANLP and NLP in general.

1.4.4 NLP processing steps

To extract an information from any content depends and rely on some levels of text extraction, or a full-up NLP techniques.and there are five primary steps involved in the NLP process.

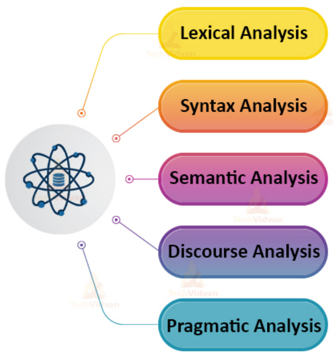


Figure 1.6: NLP processing steps

- **Lexical Analysis :** It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words,It uses some techniques including lemmatization which has the objective of reducing a word to its base form and grouping together different forms of the same word, sentence breaking which places sentence boundaries in large texts and stemming which divides words with inflection in them to root forms.

- **Syntactic Analysis (Parsing) :** It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. Syntax techniques used include parsing (i.e grammatical analysis for a sentence) and morphological segmentation which divides words into groups.

- **Semantic Analysis :** It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain.The techniques that NLP uses with semantics include word sense disambiguation which derives meaning of a word based on context, Named entity recognition which determines words that can be categorized into groups, and natural language generation which will use a database to determine semantics behind words.

- **Discourse Integration :**The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.

- **Pragmatic Analysis :** During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge

Natural Language Processing plays a critical role in supporting machine-human interactions.As more research is being carried in this field, we expect to see more breakthroughs that will make machines smarter at recognizing and understanding the human language. Natural language processing is concerned with the creation of artifacts that accomplish tasks. The operative question in evaluating an NLP algorithm or system is therefore the extent to which it produces the results for which it was designed. In the following section we will discuss some summary evalution methods.

1.5 ATS Evaluation

Since summaries tend to be more and more oriented towards specific needs, it is necessary to tune existing evaluation methods accordingly. However, the Evaluation of a summary is a difficult task because there is no ideal summary for a document or a collection of documents and the definition of a good summary is an open question to large extent [33]. Thus, Human evaluated the summary is expensive way of summarization and a difficult task for him which means the automation of the task is even more challenging and hard to assess. Therefore , we will clearly need a sophisticated evaluation methods which can handles all these difficulties.

1.5.1 Evaluation methods

A lot of diverse approaches for summary evaluation have been proposed in the last two decades. This prevents us from being exhaustive. We will instead focus on the methods that have been widely used during recent evaluation campaigns (especially during the last Text Analysis Conferences organized by NIST): Recall-Oriented Understudy for Gisting Evaluation (ROUGE) which is a fully automatic method, PYRAMID (a mixed method) and a series of indicators resulting from a manual evaluation and we will talk about other popular methods as well.

1.5.2 Manual Methods

The most obvious and simple way to evaluate a summary is to have assessors evaluating its quality. For example, for DUC (Document Understanding Conference), the judges had to evaluate the coverage of the summary, which means they had to give a global score assessing to what extent the candidate summary covers the text given as input. In more recent frameworks, and especially in TAC (Text Analysis Conference), query-oriented summaries have to be produced: judges then have to evaluate uery given in input [33].

1.5.3 Automatic Methods

In order to facilitate the summarization evaluation process, several metrics were proposed. We detail here the most popular measures: Recall-Oriented Understudy for Gisting Evaluation (ROUGE), Pyramid, GEMS and BLEU.

1.5.4 ROUGE

ROUGE is an informatively measure widely used in summarization area[34]. It measures how much of the information in human summaries are reproduced in automatic summaries. Although, it is a very simple measure (which computes the number of common n-grams in the human and automatic summaries), it includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. ROUGE is based on the calculation of Recall, Precision and F-measure. The Recall indicates the percentage of the content in the human summary that is reproduced in the automatic summary and evaluate the completeness, or how much information in the reference summaries are covered by the automatic summary. it can be computed as:

$$Recall = \frac{numberofoverlappingwords}{Totalwordsinreferencesummary} \tag{1.2}$$

The Precision indicates the percentage of the content in the automatic summary that is relevant, and evaluate the correctness, or how much information in the summary are also in the reference summaries. We notice that there is a tradeoff between precision and recall (increasing one tends to decrease the other). Precision can be computed as:

$$Precision = \frac{numberofoverlappingwords}{Totalwordsinreferencesummary} \tag{1.3}$$

F-measure combines both measures, being a unique indication of the system performance. The basic way how to compute the F-score is to count a harmonic average of precision and recall:

$$F = \frac{2 * P * R}{P + R} \tag{1.4}$$

Below is a more complex formula for measuring the F-score:

$$F = \frac{(\beta^2 + 1) * P * R}{\beta^2 * P + R} \tag{1.5}$$

There are Four different ROUGE measures are available: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S

- **ROUGE-N :** is Overlap of N-grams between an automatic summary and a set of manual summaries. Let p be "the number of common n-grams between candidate and reference summary", and q be "the number of n-grams extracted from the reference summary only", ROUGE-N is calculated as:

$$ROUGE - N = \frac{p}{q} \tag{1.6}$$

- **ROUGE-L :**computes the ratio between the length of the two summaries LCS and the length of the reference summary. One advantage of using Longest Common Subsequence (LCS) is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order as n-grams. The other advantage is that it automatically includes longest in-sequence common n-grams, therefore no predefined n-gram length is necessary. ROUGE-L is calculated as:

$$LCS(X,Y) = \frac{length(X) + length(Y) - edit_{di}(X,Y)}{2} \tag{1.7}$$

  where X and Y are representations based on sequences of words or lemmas, LCS(X,Y) is the length of the longest common subsequence between X and Y , length(X) is the length of the string X, and $edit_{di}$(X,Y) is the edit distance of X and Y .

- **ROUGE-W:** The basic LCS also has a problem that it does not differentiate LCSes of different spatial relations within their embedding sequences. To improve the basic LCS method, we introduce another metric called ROUGE-W or weighted longest common subsequence that favors LCS with consecutive matches. Route-W can be computed efficiently using dynamic programming N.

- **ROUGE-S:** measure the overlap ratio of skip-bigrams between a candidate summary and a set of reference summaries. For example, sentence "police killed the gunman" has C(4.2) = 6 skip-bigrams: ("police killed", "police the", "police gunman", "killed the", "killed gunman", "the gunman").

- **ROUGE-SU:** One potential problem for ROUGE-S is that it does not give any credit to a candidate sentence if the sentence does not have any word pair co-occurring with its references. To accommodate this, we extend ROUGE-S with the addition of unigram as counting unit. The extended version is called ROUGE-SU.

1.5.5 Generative Modelling for Evaluation of Summaries (GEMS)

This method [35] suggests the use of signature for analyzing that how they are captured in automatic summaries. The signature terms are calculated on the basis of part-of-speech tags, such as nouns or verbs query terms and terms of reference summaries. The distribution of the signature terms is calculated in the source document and then the possibility of a summary being biased towards such signature-terms is gained.

1.5.6 BLEU

This measure is looking at how much the words (and/or n-grams) in the machine generated summaries appeared in the human reference summaries, or the n-grams overlap between the output and reference translations with a penalty for shorter outputs, is known as BLEU (short for "Bilingual evaluation understudy" which people literally only ever say when explaining the acronym) and was developed by [36]. It's a very popular metric in NLP, particularly for tasks where the output of a system is a text string rather than a classification. This includes machine translation and, increasingly, natural language generation.

1.5.7 Pyramid

As we have seen, ROUGE measures are based on the discovery of perfect matches between some sequences of the candidate summary and some of the reference summaries. These methods are thus inefficient if the candidate summary has been produced using reformulation techniques. PYRAMID [37] is supposed to overcome some of these issues. This evaluation is more precise than ROUGE but requires some manual work to identify summarization content units (SCUs) that are used for comparison of information in summaries, associate linguistic expressions to them and calculate the weights necessary for the evaluation. However, according to [38], this method seems to better correlate with human judgements than ROUGE probably because it takes into account some of the semantics of the text.

1.6 Conclusion

Automatic Text Summarization is one of the most challenging and interesting problems and one of the most important processes in the field of Natural Language Processing (NLP). In this chapter, we presented the different types and approaches used in the automatic summarization of texts, then we talked about how evaluation of summary works and then we explained steps and techniques used in Naturel language processing. The next chapter will focus on Deep neural networks and how they can be used in the NLP tasks. We will more focus on Recurrent Neural Network which are the most appropriate with ATS tasks.

Chapter 2

Deep neural networks

2.1 Introduction

Automatic text summarization has consequently became popular, and until recently text summarization was dominated by unsupervised information retrieval models. In [39], demonstrated that the neural-based continuous models are promising for text summarization This marked the beginning of the widespread use of neural network-based text summarization models, because of their superior performance compared to the traditional techniques. . In this chapter we present the various model of deep learning also the different types of neural network used in the automatic summarization of texts, then we detail the recurrent neural network (RNN) architecture and we explain the work of it .

2.2 Machine learning

Machine learning (ML)is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.ML focuses on the development of computer programs that can access data and use it to learn for themselves.The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.ML algorithms are often categorized as supervised, unsupervised, and reinforcement learning.

2.3 machine learning methods

2.3.1 Supervised Learning

Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data [4]. Supervised learning classified into two categories of algorithms:

- Classification: is a problem that is used to predict which class a data point is part of which is usually a discrete value.
- Regression: is a problem that is used to predict continuous quantity output. A continuous output variable is a real-value, such as an integer or floating point value.

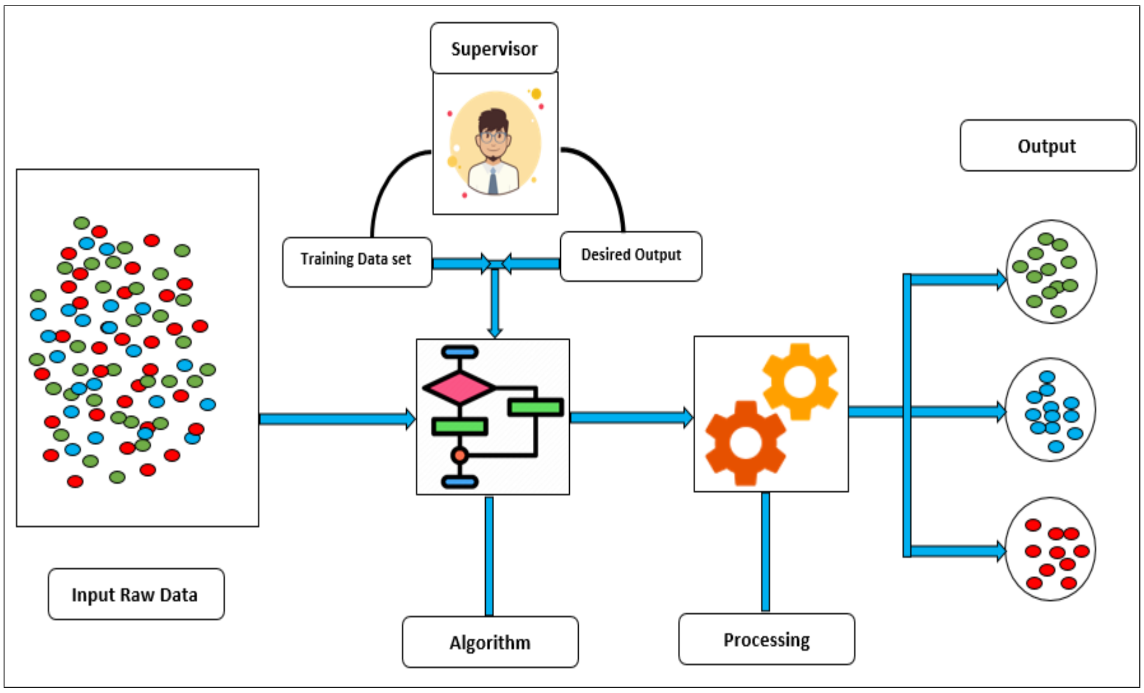


Figure 2.1: Supervised machine learning, the algorithm learns from labeled data

2.3.2 Unsupervised Learning

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Unlike supervised learning, no teacher is provided, that means no training will be given to the machine. Therefore, machine is restricted to find the hidden structure in unlabeled data by our-self[4].

Unsupervised learning classified into two categories of algorithms:

- Clustering: the process of organizing objects into groups whose members are similar in some way.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

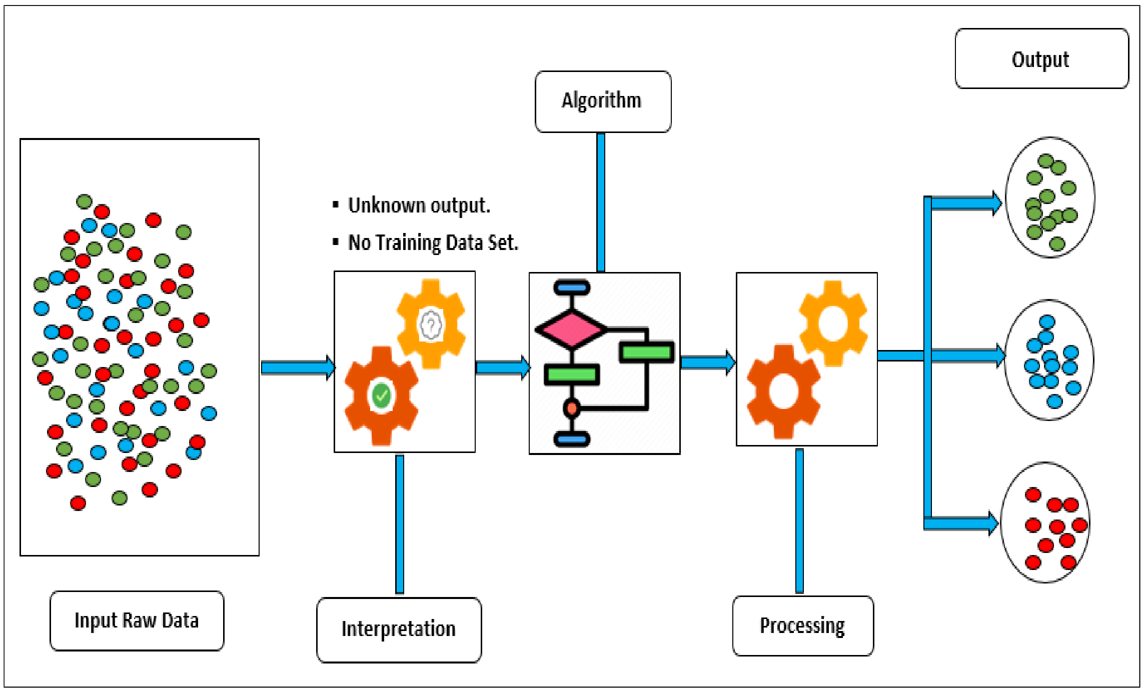


Figure 2.2: Unsupervised machine learning

2.3.3 Reinforcement Learning

Reinforcement learning is an area of Machine Learning introduced by [40] and [41] . It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience.

Types of Reinforcement: There are two types of Reinforcement:

- Positive: Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words it has a positive effect on the behavior.
- Negative: Negative Reinforcement is defined as strengthening of a behavior because a negative condition is stopped or avoided.

2.4 Deep learning

Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. Similarly, to how we learn from experience, the deep learning algorithm would perform a task repeatedly, each time tweaking it a little to improve the outcome [42]. We refer to 'deep learning' because the neural networks have various (deep) layers that enable learning [42]. Just about any problem that requires "thought" to figure out is a problem deep learning can learn to solve. Hence, neural networks are sometimes referred to deep neural networks as well. Usually there are 3-4 hidden layers in a simple neural network while a deep neural network usually can have tens or hundreds of hidden layers. Deep learning models with neural network architectures are trained, so that the models can learn features automatically by analyzing the labeled examples. Hence, The performance of a deep learning system increases with the increase of the amount of data. In traditional machine learning approaches, the performance becomes constant after certain steps, while the performance of the deep learning systems increases with the increase of the amount of data.

2.5 Neural Network

A neural network (NN) is a type of machine learning which models itself after the human brain, creating an artificial neural network that via an algorithm allows the computer to learn by incorporating new data. NN is formed of a set of neurons connected between each other. The number of neurons to construct a neural network can be hundreds or even millions which are arranged in a series of layers. The hidden layers stay in between the input and output layers. The number of hidden layer can be zero or more.There are several types of NN used in machine and deep learning

2.5.1 Artificial Neural Networks (ANN)

An artificial neural network (ANN) is a computational model based on the structure and functions of biological neural networks[43]. Information that flows through the network affects the structure of the ANN because a neural network changes or learns in a sense based on input and utput. In a neural network, there are input and output layers, as well as hidden layers in most of the cases. The hidden layers take the inputs, transform the inputs, and pass them to the output layer Figure 2.3 demonstrates an ANN structure. ANNs are considered nonlinear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found. ANN is also known as a neural network.

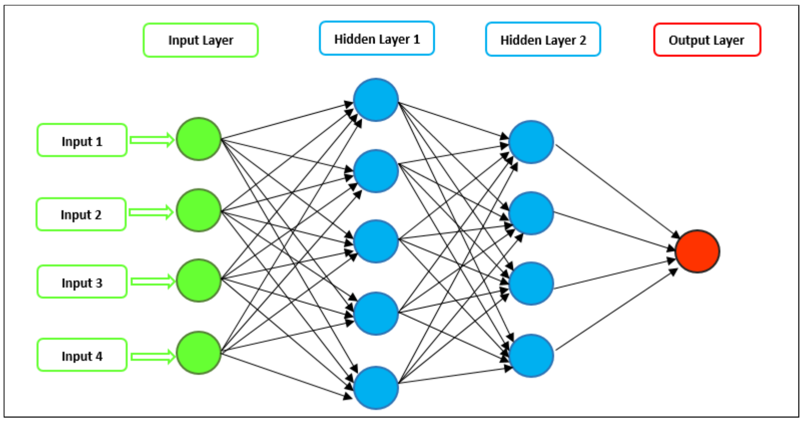


Figure 2.3: A schematic artificial neural network (ANN) with two hidden layers and a single neuron output

2.5.1.1 Global Architecture

Neural networks are organized in various layers:

- **Input layer:** the input layer neurons receive the information supposed to explain the problem to be analyzed;

- **Hidden layer:** the hidden layer is an intermediate layer allowing neural networks to model nonlinear phenomena. This said to be "hidden" because there is no direct contact with the outside world. The outputs of each hidden layer are the inputs of the units of the following layer.

- **Output layer:** the output layer is the last layer of the network; it produces the result, the prediction.

2.5.1.2 Artificial Neural Network Concepts

- **Inputs:** Source data fed into the neural network, with the goal of making a decision or prediction about the data. Inputs to a neural network are typically a set of real values; each value is fed into one of the neurons in the input layer.

- **Training Set:** A set of inputs for which the correct outputs are known, used to train the neural network.

- **Outputs:** Neural networks generate their predictions in the form of a set of real values or boolean decisions. Each output value is generated by one of the neurons in the output layer.

- **Neuron/perceptron:** The basic unit of the neural network. Accepts an input and generates a prediction. Each neuron accepts part of the input and passes it through the activation function. Common activation functions are sigmoid, TanH and ReLu. Activation functions help generate output values within an acceptable range, and their non-linear form is crucial for training the network.

- **Weight Space:** Each neuron is given a numeric weight. The weights, together with the activation function, define each neuron's output. Neural networks are trained by fine-tuning weights, to discover the optimal set of weights that generates the most accurate prediction.

- **Forward Pass:** The forward pass takes the inputs, passes them through the network and allows each neuron to react to a fraction of the input. Neurons generate their outputs and pass them on to the next layer, until eventually the network generates an output.

- **Error Function:** Defines how far the actual output of the current model is from the correct output. When training the model, the objective is to minimize the error function and bring output as close as possible to the correct value.

- **Backpropagation:** In order to discover the optimal weights for the neurons, we perform a backward pass, moving back from the network's prediction to the neurons that generated that prediction. This is called backpropagation. Backpropagation tracks the derivatives of the activation functions in each successive neuron, to find weights that brings the loss function to a minimum, which will generate the best prediction. This is a mathematical process called gradient descent.

2.6 Activation Function in Neural Network

The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

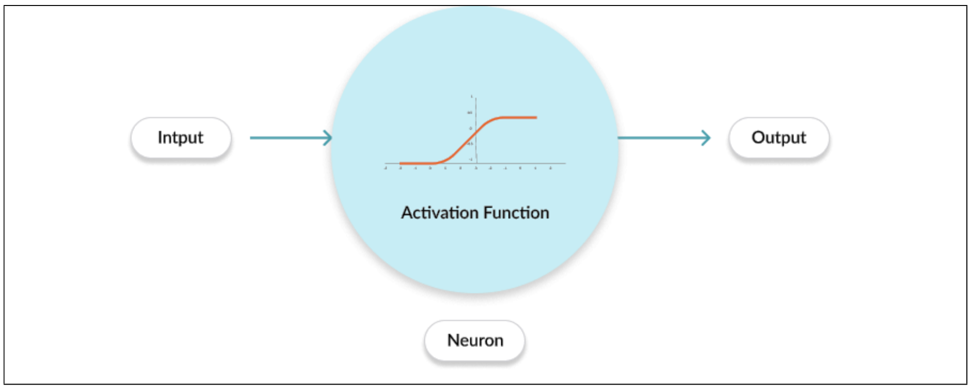


Figure 2.4: A Simple working of Activation Function[4]

2.6.1 Types of Activation Functions

2.6.1.1 Binary Step Function

A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.

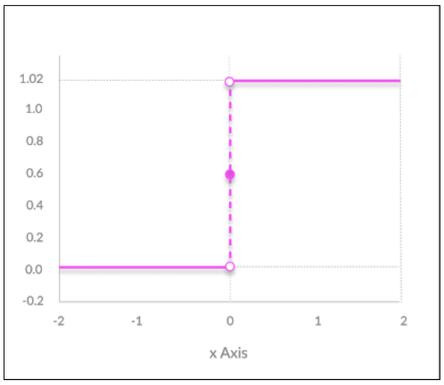


Figure 2.5: Graph of A binary step function[4]

2.6.1.2 Linear Activation Function

A linear activation function takes the form:A = cx .It takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input. In one sense, a linear function is better than a step function because it allows multiple outputs, not just yes and no.

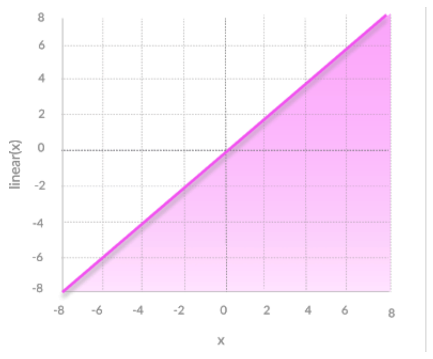


Figure 2.6: Graph of Linear Activation Function[4]

2.6.1.3 Non-Linear Activation Functions

Modern neural network models use non-linear activation functions. They allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or have high dimensionality.

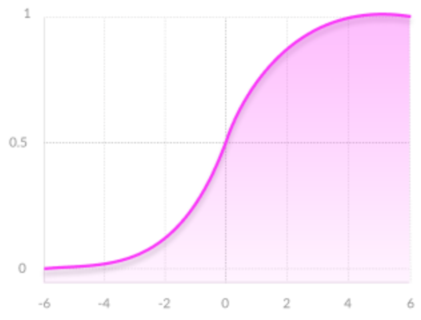


Figure 2.7: Graph of Non-Linear Activation Functions[4]

2.6.2 List of common Activation Functions

2.6.2.1 Sigmoid

A sigmoid function is a type of activation function, and more specifically defined as a squashing function. Squashing functions limit the output to a range between 0 and 1, making these functions useful in the prediction of probabilities.

2.6.2.2 ReLU

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as y = max (0, x). It is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network, ReLU is usually a good first choice.

2.6.2.3 Tanh

Tanh is also like logistic sigmoid but better. The range of the Tanh function is from (-1 to 1). Tanh is also sigmoid (s - shaped).and the advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the Tanh graph.

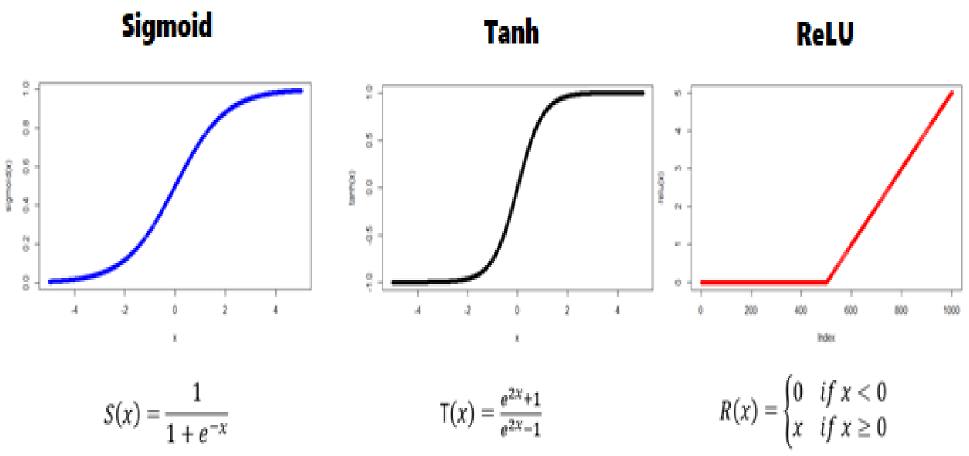


Figure 2.8: Graph of commonly used activation functions[5]

2.7 Types of ANN

2.7.1 neural network (CNN)

A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptions a ML unit algorithm for supervised learning to analyze data. A CNN is also known as a ConvNet Like other kinds of artificial neural networks, it has an input layer, an output layer and various hidden layers. Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. These layers take the input, transform the input, and then feed the transformed input into the next layer. The layers can detect patterns by using filters. Each layer has a large number of filters. Filters are represented by a matrix, which are moved over the original matrix and multiplied with the corresponding index of the original matrix, and sums them up to convolve the required features. This simulates some of the actions in the human visual cortex. CNNs are a fundamental example of deep learning, where a more sophisticated model pushes the evolution of artificial intelligence by offering systems that simulate different types of biological human brain activity. CNNs can be applied to image processing [44], natural language processing and other kinds of cognitive tasks.

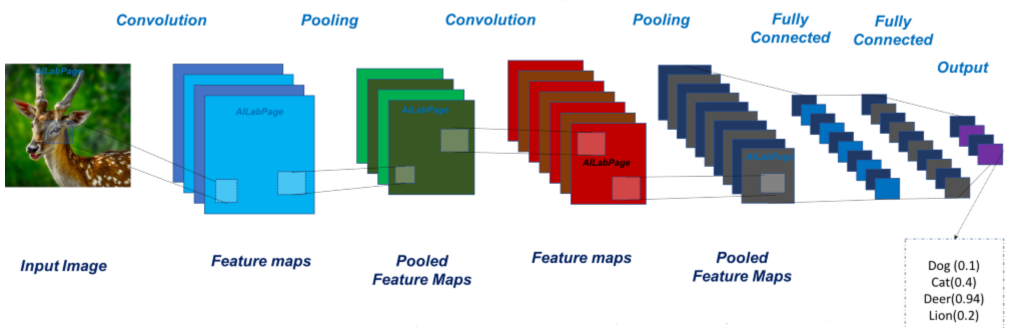


Figure 2.9: CNN structure used for image recognition[6].

2.7.1.1 CNN building technique

This progression is one among the foremost vital strides of the task and It contains 3 sections

- **Convolution:** The essential role of Convolution is to extricate highlights from the image. Convolution saves the spatial connection between pixels by learning image highlights utilizing little squares of information.
- **Pooling:** :Pooling is also known as subsampling or downsampling and it reduce the size of the spatial dimension by sub-sampling their inputs, which is applied after the convolutional layers , Pooling layer is normally inserted periodically between two successive convolutional layers . Pooling is very important to provide a fixed size output matrix and reduce the complexity of the output dimensions.
- **Flattening:** The matrix changed over into a linear array that to enter it into the hubs of our neural system.

2.7.2 Recurrent Neural Networks (RNN)

Recurrent Neural Network (RNN) [45] are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional NNs, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. For example, we take a sequence of two words. We are told to predict the third word of the sequence. It would be a lot easier for the model to predict the third word if it knew the first two words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.RNN have a "memory" which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output[46]. This reduces the complexity of parameters, unlike other neural networks.

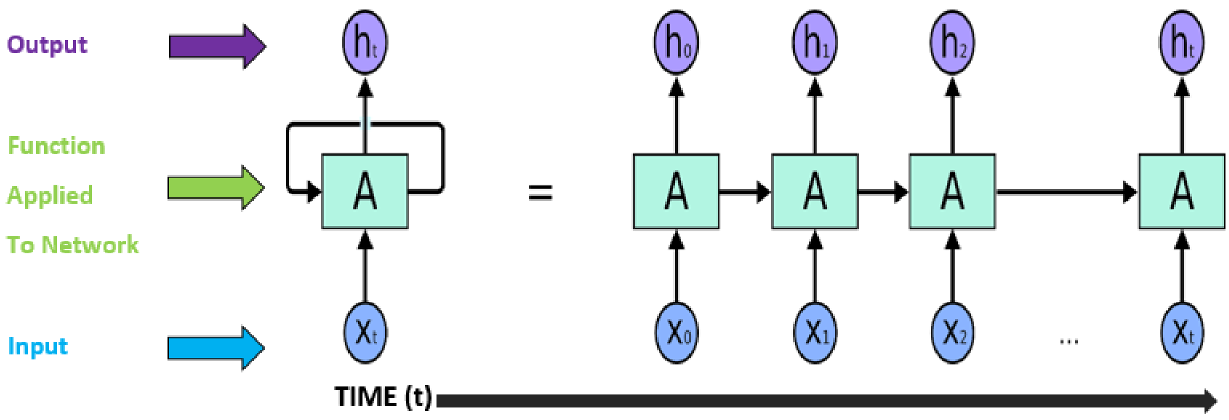


Figure 2.10: An unrolled recurrent neural network[7].

The above figure shows a RNN being unfolded into a full network. By unfolding we simply mean that we are repeating the same layer structure of network for the complete sequence.

- X_t: is the input at time step t. X_t is a vector of any size N.
- A is the hidden state at time step t. It's the "memory" of the network.

It is calculated based on the previous hidden state and the input at the current step. Represented by At= f (W Xt +U At -1). Here W and U are weights for input and previous state value input. And f is the non-linearity applied to the sum to generate final cell state.

2.7.2.1 Training through RNN

- A single time step of the input is provided to the network.
- Then calculate its current state using set of current input and the previous state.
- The current ht becomes ht-1 for the next time step.
- One can go as many time steps according to the problem and join the information from all the previous states.
- Once all the time steps are completed the final current state is used to calculate the output.

- The output is then compared to the actual output i.e the target output and the error is generated.
- The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

2.8 Recurrent neural network architecture types

2.8.1 Fully Recurrent Neural Network

Fully recurrent neural network (FRNN) developed in the [47] , which can learn temporal sequences, either in batch mode or online. FRNN consists of two layers, input and output layer of linear and non-linear units, resp. The units in input layer is fully connected to every units of output layer by adjustable weights. Each unit has a real-valued time-varying activation function. The output units have some knowledge of their prior activations, which feedback the activations to the input layer units. Learning in FRNNs is by mapping input sequences and activations, to another set of output sequences. This continue to feedback to input sequences and finding output sequences over multiple time steps, and over time discover abstract representations.

2.8.2 Recursive Neural network

Network created in differentiable graph like structure by recursively applying same set of weights to network in topological order. Such networks are also trained by automatic differentiation [48] in reverse mode. It corresponds to linear chain structure and are used in natural language processing, processing distributed representation of structure. Variation of recursive neural network is Recursive Neural Tensor Network which uses tensor-based composition function on every network nodes.

2.8.3 Bi-directional RNN

Bi-directional Recurrent Neural Network predicts each element of a finite sequence based on its past/previous and future/next situation. It works in both direction for processing sequence from left-to-right and right-to-left and concatenating their output. This technique is useful when combined with Long Short-Term Memory (LSTM) [47].

2.8.4 Long Short-Term Memory (LSTM)

Long Short Term Memory (LSTM) network which was invented by [49], LSTM is a system which can learn task by using deep learning and avoids vanishing gradient problem [50]. LSTM is normally improved by recurrent gates called "forget" gates and to learn tasks it requires memory of event happened in history,It can be learned by Connectionist Temporal Classification (CTC) which achieves both alignment and recognition for weights. The core concept of LSTM's are the cell state, and it's various gates. The cell state acts as a transport highway that transfers relative information all the way down the sequence chain.

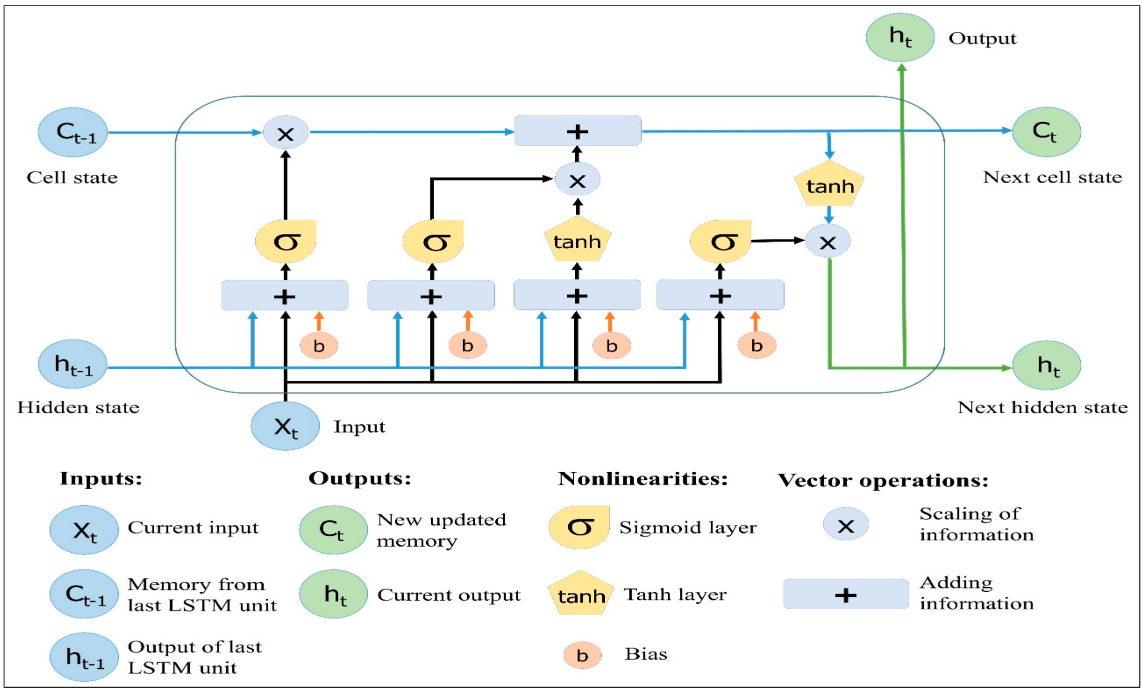


Figure 2.11: LSTM Cell and It's Operations[8].

2.8.4.1 Gates of LSTM

The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training

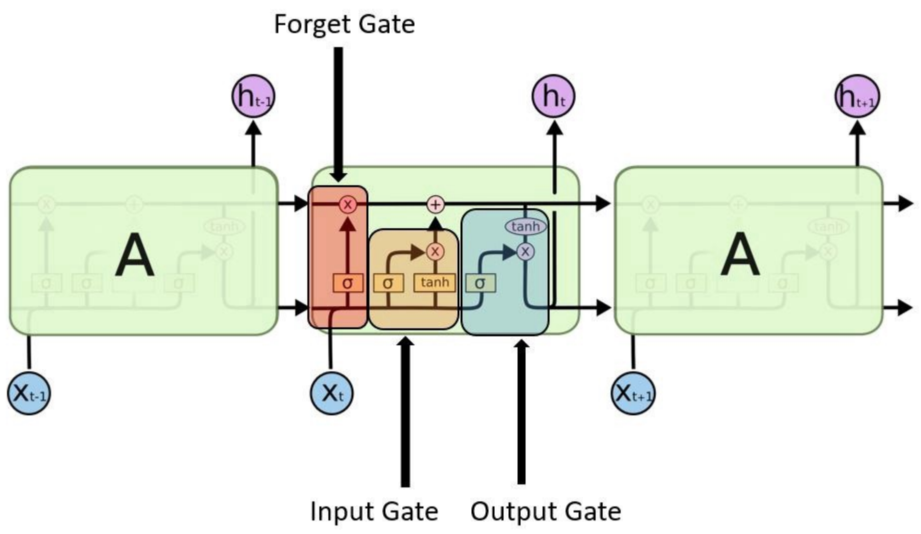


Figure 2.12: The architecture of a LSTM[8].

- **Forget gate** This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep. The information we need to forget is determined by the following equation:
$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_t - 1) \tag{2.1}$$
where x_t is the input in time step t, and h_{t-1} is the activation output of the previous time step. Both of them are multiplied by their corresponding weight ($W^{(f)}$ and $U^{(f)}$) and added together. Then they are passed through a sigmoid activation function.

- **Input Gate** In the input gate pass the previous hidden state and current input into a activation function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. [20] First, to determine which value it needs to update ,it use the following equation:
$$t_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \tag{2.2}$$
The new candidate values, Cttemp is calculated using the following:
$$c_{t_temp} = \tanh(W^{(c)}x_t + U^{(i)}h_{t-1}) \tag{2.3}$$
where tanh is an activation function for this layer which represents the output within -1 and 1. Finally, the memory content in the current time step is determined by the following equation:
$$c_{t_temp} = f_t \odot C_{t-1} + i_t \odot C_{t_temp} \tag{2.4}$$

where $C_t - 1$ is the memory content from the previous time step, and represents the Hadamard product operation.

- **Output Gate** The output gate decides what the next hidden state should be. And the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, passing the previous hidden state and the current input into a sigmoid function. Then passing the newly modified cell state to the tanh function. multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step , To determine what information is going to pass through the output the following two equations will achieve that :

$$o_t = \sigma(W^{(0)}x_t) + U^{(o)}ht - 1 \tag{2.5}$$

$$h_t = o_t \odot tanh(C_t) \tag{2.6}$$

2.8.5 Gated Recurrent Units (GRU)

The GRU was first introduced by[51],it is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU's got rid of the cell state and used the hidden state to transfer information. A GRU consists of an update gate and a reset gate [44]. These two gates can decide which information to carry forward for passing on to the next iteration. The main advantage of using GRU is that they can be used to keep track of long sequences without vanishing it, even if it is not relevant to the prediction. Figure 2.13 shows a basic structure of single GRU unit.

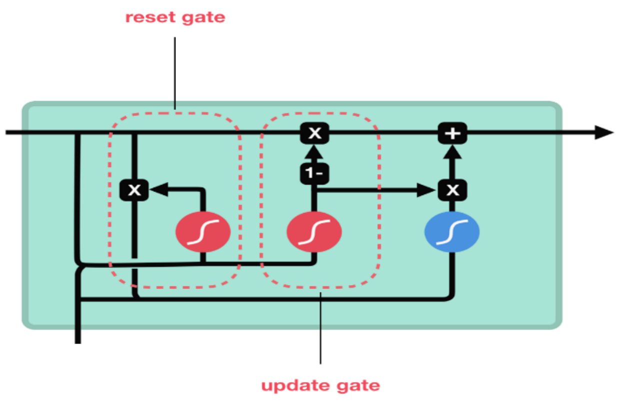


Figure 2.13: The internal structure of a gate recurrent unit (GRU)[9].

- **Update Gate(z):** Update Gate determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit. It update the gate z_t for each time step by the following formula:

$$z_t = \sigma(W^{(z)} + U^{(z)}h_{t-1}) \tag{2.7}$$

where x_t is the input in time step t, and $h_{(t-1)}$ is the activation output of the previous time step. Both of them are multiplied by their corresponding weight ($W^{(z)}$, and $U^{(z)}$) and added together. Then they are passed through a sigmoid activation function [52]).

- **Reset Gate(r):** Reset Gate determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit. It calculate this by the following formula:

$$r_t = \sigma(W^{(r)} + U^{(r)}h_{t-1}) \tag{2.8}$$

where x_t is the input in time step t, and$h_{(t-1)}$ is the activation output of the previous time step. Both of them are multiplied by their corresponding weight ($W^{(z)}$, and $U^{(z)}$), and added together. Then they are passed through a sigmoid activation function.

2.9 Sequence to sequence Architecture

Seq2seq turns one sequence into another sequence. It does so by use of a RNN or more often LSTM or GRU to avoid the problem of vanishing gradient. The context for each item is the output from the previous step[53]. build a Seq2Seq model can be on any problem which involves sequential information. This includes Sentiment classification, Neural Machine Translation, and Named Entity Recognition those are some very common applications of sequential information.

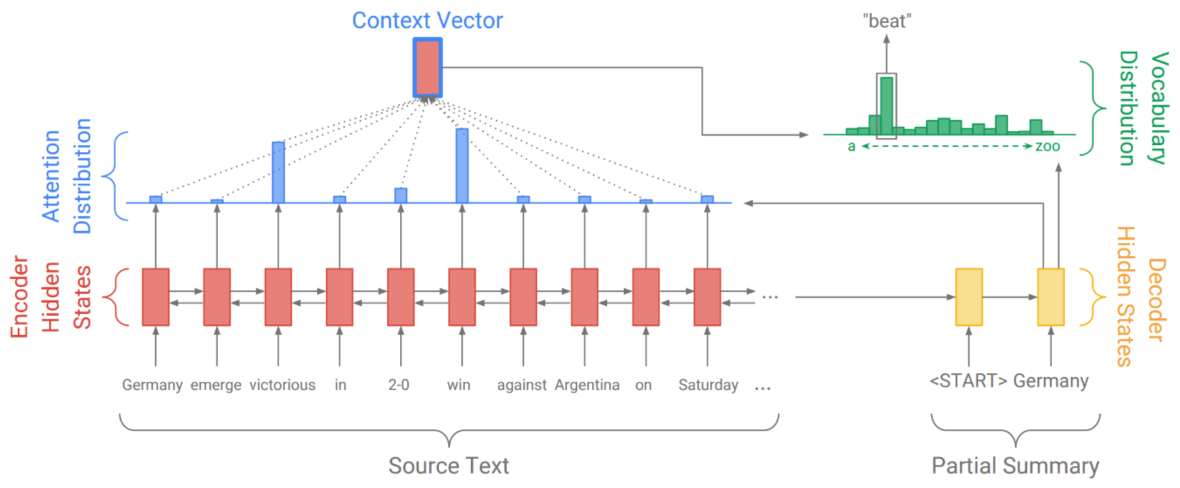


Figure 2.14: Sequence to sequence Architecture[10].

Seq2Seq models are typically implemented with the help of two recurrent neural networks (usually an LSTM or GRU). In the most basic version, the input/source sequence is fed token-by-token to the first RNN (encoder), which computes a vector representation for the whole sequence. This vector representation becomes the starting point for the second RNN (decoder), which generates the output/target sequence, again in a token-by-token manner. Since generation of text summarization is the focus of this thesis work, LSTM based sequence-to-sequence models are used with an encoder LSTM an decoder LSTM.

2.9.1 LSTM Encoder

An Encoder LSTM reads the entire input sequence wherein, at each timestep, one word is fed into the encoder. It then processes the information at every timestep and captures the contextual information present in the input sequence.

2.9.2 LSTM Decoder

The decoder is also an LSTM network which reads the entire target sequence word-by-word and predicts the same sequence offset by one timestep. The decoder is trained to predict the next word in the sequence given the previous word.

2.9.3 Conclusion

In this chapter, we have discussed the various methods of deep learning that are employed in text summarization process and also the techniques that are developed over the years. We have also discussed the activation functions, which have been used to evaluate the results obtained by deep learning approaches. We have focused on LSTM RNN architecture which has been commonly used for the text summarization purpose. we have found that this method gives results either better or at par to the traditional models. It has been found that the encoder-decoder models with LSTM and GRU are the most widely used approaches for the summarization purpose. In the following chapter, we will detail our deep text summarization model.

Chapter 3

Contribution

3.1 Introduction

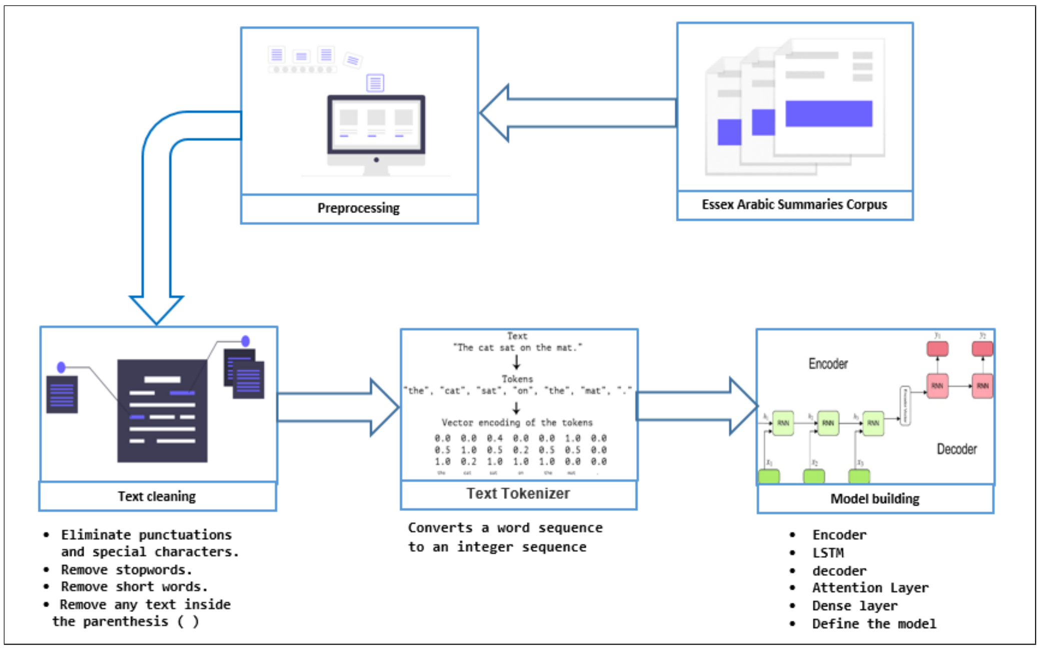
Our contribution consists of adapting, testing and comparing two deep summarization approaches using Arabic text. The two appraoches used LSTM-RNN models. The first one is based on an abstractive model where the second one is based on extractive one. Training and testing are conducted on EASC (Essex Arabic Summaries Corpus)[54] . Summarization process starts with preprocessing, division on traning and testing dataset and then fitting and testing models. The main objective is to compare effectivness of deep abstractive and extractive approaches when dealing with Arabic text. It should be noted that in order to adapt and test the above cited apporaches, several updates are carried on at different levels.

3.2 Abstractive summarization approach

The implemented abtractive appraoch consists of an LSTM-RNN sequence2sequence model. It was originally carried on English text [55]. Abstractive Summarization creates a shorter and informative version of a text document by extracting important information from the text and generating new sentences using that information. Therefore, we adapte summarization model in order to deal with Arabic language. The model use RNN-sequence-2-sequence architecture which basically comprises of Encoder and Decoder models connected sequentially so as to generate an output (Arabic Summary) for a given input (Arabic Text). The decoding algorithm will be able to accept and generate longer sequences than the previous sequence-to-sequences models and fix all problems caused by a such adaptation.

3.3 Abstractive summarizer architecture

In order to build the summarization model, several processing steps are followed (Figure 3.1). The process starts with a dataset preparation step, in which, text preprocessing and cleaning are executed. In order to prepare model input data, a tokenization process is then realised. The process ends with model building. In what follows, we describe step-by-step the different processing levels.



Preprocessing
Essex Arabic Summaries Corpus
Text cleaning
• Eliminate punctuations and special characters.
• Remove stopwords.
• Remove short words.
• Remove any text inside the parenthesis ( )
Text
"The cat sat on the mat."
Tokens
"the", "cat", "sat", "on", "the", "mat", "."
Vector encoding of the tokens
Text Tokenizer
Converts a word sequence to an integer sequence
Encoder
Decoder
RNN
Model building
• Encoder
• LSTM
• decoder
• Attention Layer
• Dense layer
• Define the model

Figure 3.1: System architecture to build the model.

3.3.1 Dataset description

dataset (EASC) is an Arabic natural language resources. It consists of 153 Arabic articles and 765 human-generated summaries of those articles, organized in ten categories Each article is accompanied with five references summaries. These summaries were generated using Mechanical Turk . We take the entire dataset for training our model. Among the major features of EASC are Names and extensions are formatted to be compatible with current evaluation systems such as ROUGE. The dataset is Available in two encoding formats UTF-8 and ISO-8859-6 (Arabic). The following table shows the details of EASC corpus :

Articles	Number of articles	Wikipedia	Alwatan Newspaper	Alrai Newspaper
Art and Music	10	6	3	1
Education	7	4	2	1
Environment	33	32	0	1
Health	17	11	3	3
Finance	17	2	15	0
Politics	21	9	5	7
Religion	8	7	1	0
Science and Technology	16	11	5	0
Sport	10	10	0	0
Tourisms	14	14	0	0
Total	153	106	34	13

Table 3.1: EASC corpus article categories.

3.3.2 Preprocessing

Data preprocessing is a crucial and decisive step in machine learning applications [56] A mere flaw in a dataset is a potentially disastrous move and can ruin hours and days of training procedure.Therefore, Performing basic preprocessing steps is very important before getting to the model building part . the preprocessing strategies are taken for EASC datasets .The articles are long. They are sometimes bigger than the average. Figure 3.2 shows the distribution of the sequences. This way guarantees that almost the instances have equal size and the rest are smaller. In this step all the unwanted symbols, characters are droped from the text that do not affect the objective of the problem .There are many important steps in data preprocessing some of them are as follows:

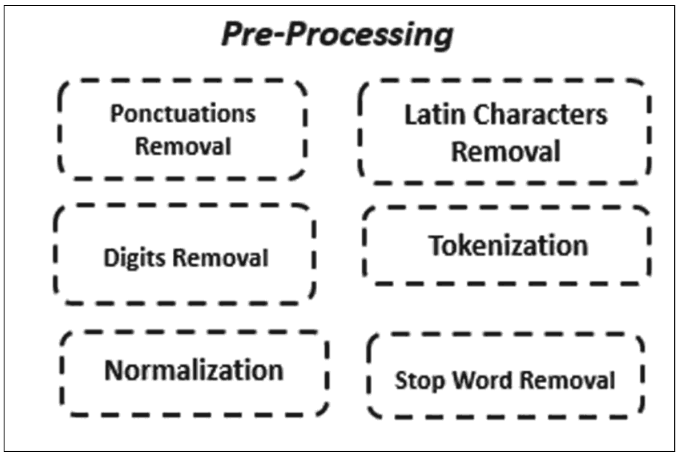


Figure 3.2: Pre-processing steps.

- **Remove any text inside the parenthesis ( ):**because they provide extra information about something else in the sentence.
- **Eliminate punctuations and special characters:** Removing punctuation from the text like I can't use the {$ - _ . , ; : ? !} and also the symbols like
- **Remove stopwords:** Stopword removal is the process of removing stop words. (words which do not convey any information, such as "من ، على" etc. which are insignificant in feature score calculation), they must be removed to simplify the task of the summarizer.

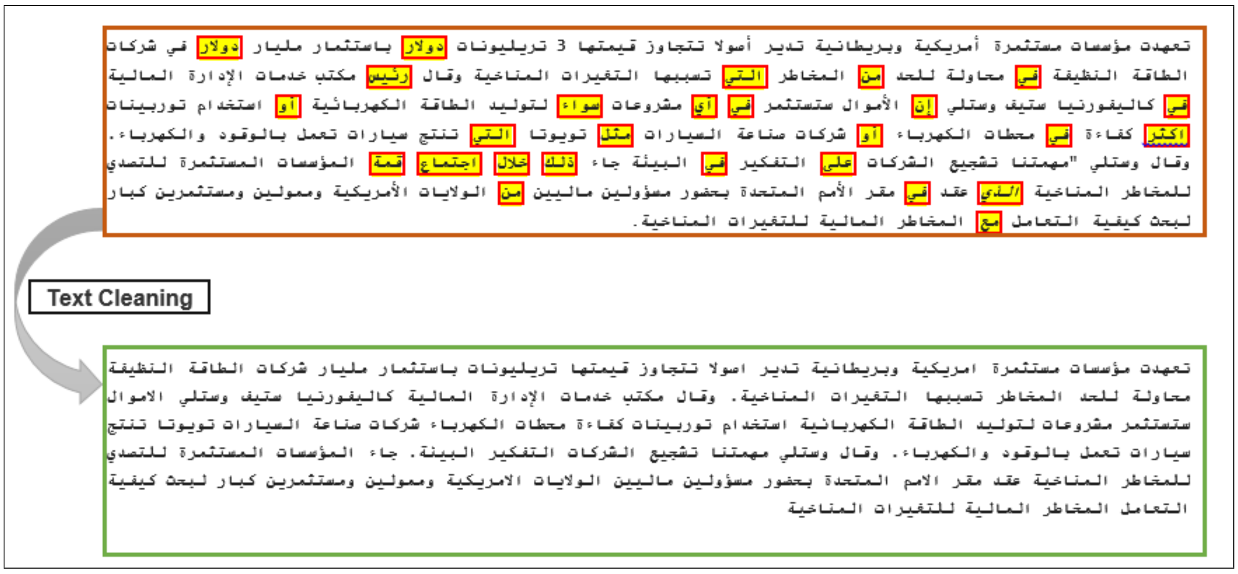
Moving on to the data preprocessing step, data cleaning is implemented along with removal of stopwords in order to make data ready for our model. This phase is implemented separately for both summary and complete text of the training model.

3.3.3 Data cleaning

Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information. We will perform the below cleaning tasks for our data:

3.3.3.1 Text cleaning

It means we will perform the cleaning preprocessing tasks for the articles of our dataset as shown in figure below:



تعهدت مؤسسات مستثمرة أمريكية وبريطانية تدير أصولا تتجاوز قيمتها 3 تريليونات دولار باستثمار مليار دولار في شركات الطاقة النظيفة في محاولة للحد من المخاطر التي تسببها التغيرات المناخية وقال رئيس مكتب خدمات الإدارة المالية في كاليفورنيا ستيف وستلي إن الأموال ستستثمر في أي مشروعات سواء لتوليد الطاقة الكهربائية أو استخدام توربينات أكثر كفاءة في محطات الكهرباء أو شركات صناعة السيارات مثل تويوتا التي تنتج سيارات تعمل بالوقود والكهرباء. وقال وستلي "مهمتنا تشجيع الشركات على التفكير في البيئة جاء ذلك خلال اجتماع قمة المؤسسات المستثمرة للتصدي للمخاطر المناخية الذي عقد في مقر الأمم المتحدة بحضور مسؤولين ماليين من الولايات الأمريكية ومموّلين ومستثمرين كبار لبحث كيفية التعامل مع المخاطر المالية للتغيرات المناخية.

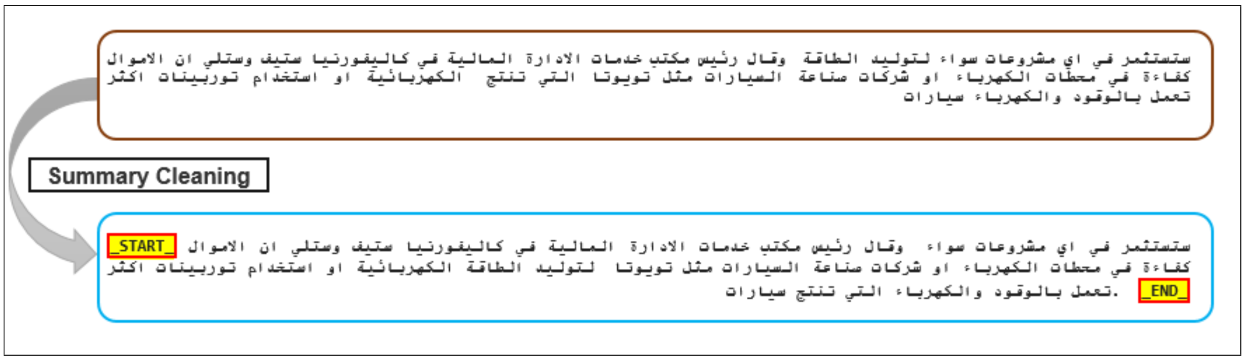
Text Cleaning

تعهدت مؤسسات مستثمرة امريكية وبريطانية تدير اصولا تتجاوز قيمتها تريليونات باستثمار مليار شركات الطاقة النظيفة محاولة للحد المخاطر تسببها التغيرات المناخية. وقال مكتب خدمات الإدارة المالية كاليفورنيا ستيف وستلي الاموال ستستثمر مشروعات لتوليد الطاقة الكهربائية استخدام توربينات كفاءة محطات الكهرباء شركات صناعة السيارات تويوتا تنتج سيارات تعمل بالوقود والكهرباء. وقال وستلي مهمتنا تشجيع الشركات التفكير البيئة. جاء المؤسسات المستثمرة للتصدي للمخاطر المناخية عقد مقر الامم المتحدة بحضور مسؤولين ماليين الولايات الامريكية ومموّلين ومستثمرين كبار لبحث كيفية التعامل المخاطر المالية للتغيرات المناخية

Figure 3.3: An example of Text cleaning.

3.3.3.2 Summary cleaning

It means we will clean the text of every summary we have in our dataset and add the “START” and “END” special tokens at the beginning and end of the summary as shown in figure below:



ستستثمر في اي مشروعات سواء لتوليد الطاقة وقال رئيس مكتب خدمات الادارة المالية في كاليفورنيا ستيف وستلي ان الاموال كفاءة في محطات الكهرباء او شركات صناعة السيارات مثل تويوتا التي تنتج الكهربائية او استخدام توربينات اكثر تعمل بالوقود والكهرباء سيارات

ستستثمر في اي مشروعات سواء وقال رئيس مكتب خدمات الادارة المالية في كاليفورنيا ستيف وستلي ان الاموال _START_ كفاءة في محطات الكهرباء او شركات صناعة السيارات مثل تويوتا لتوليد الطاقة الكهربائية او استخدام توربينات اكثر _END_ تعمل بالوقود والكهرباء التي تنتج سيارات.

Figure 3.4: An example of Summary cleaning.

3.3.4 Sequence distribution analysis

In order to get the correct parameters of our input and output data, we analysed text and summaries length (474919). This will help us fix the maximum length of the model sequences.

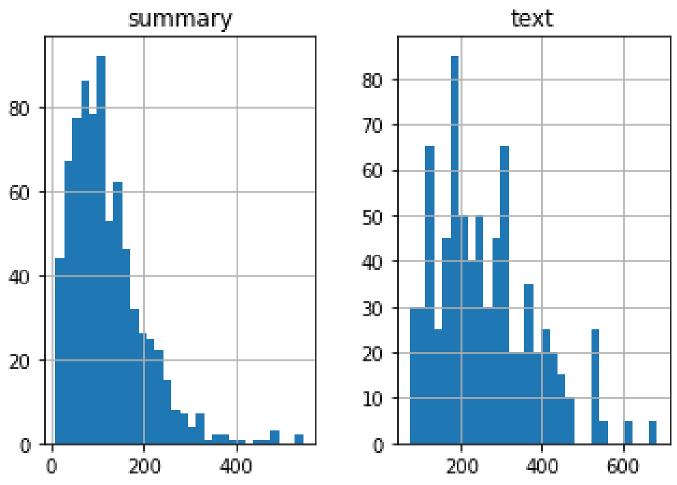


Figure 3.5: Sequences distribution.

According to Figure 3.5 Article maximum length is fixed to 350. This seems to be the average text length. Similarly, summaries maximum length is fixed to 80.

3.3.5 Data Splitting

This step consists of dividing our dataset into two subsets:

- **Training set:**it is a subset to train a model.
- **Test set:**it's a subset to test the trained model

We'll use 90% of the dataset as the training data and evaluate the performance on the remaining 10% (holdout set)

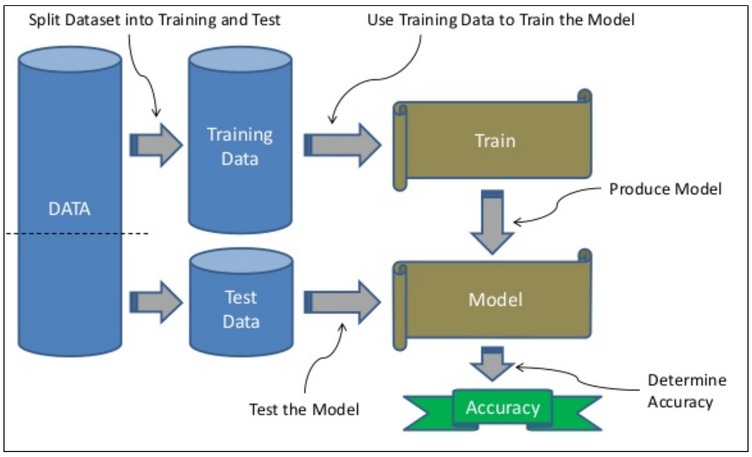
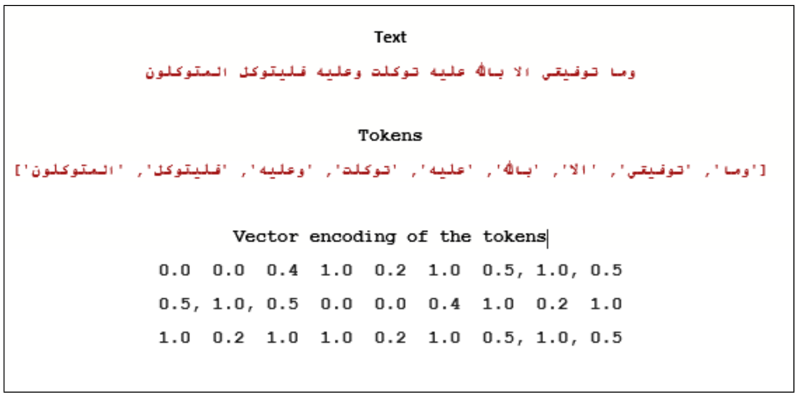


Figure 3.6: Splittig Data for Machine Learning

3.3.6 Tokenization

Word tokenizer tokenizes words into separate entities within a sentence (Figure 3.7). This step is especially important if we want to calculate the feature scores of individual words of a sentence for deducing important sentences in a document.



Text

وما توفيقي الا بالله عليه توكلت وعليه فليتوكل المتوكلون

Tokens

['وما', 'توفيقي', 'الا', 'بالله', 'عليه', 'توكلت', 'وعليه', 'فليتوكل', 'المتوكلون']

```
Vector encoding of the tokens
0.0  0.0  0.4  1.0  0.2  1.0  0.5, 1.0, 0.5
0.5, 1.0, 0.5  0.0  0.0  0.4  1.0  0.2  1.0
1.0  0.2  1.0  1.0  0.2  1.0  0.5, 1.0, 0.5
```

Figure 3.7: An example of word tokenization

Also a tokenizer builds the vocabulary and allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary , and we have to tokenize both the articles and the summaries of the entire dataset.

3.3.7 Model building

In the model building phase, the main objective is to build a text summarizer where the input is a long sequence of words (in a text body), and the output is a short summary (which is a sequence as well).This problem can be modeled as a Many-to-Many Seq2Seq problem, Seq2seq maps 2 sequences to each other that are not necessarily in the same size, in two steps: Compressing the first sequence, and then inferring the output from it. This architecture has two side named encoder and decoder that are both LSTM layers.The LSTM take a sequence of tokens as input and transform them into a sequence of hidden states. the hidden state are calculated for both the readers, and combine the result using the following equations[57] :

$$\vec{h_t} = LSTM(\vec{h_{t-1}}, E(\vec{W_t})) \tag{3.1}$$

$$\vec{h_t} = LSTM(\vec{h_{t-1}}, E(\vec{W_t})) \tag{3.2}$$

$$h_t = [\vec{h_t}, \vec{h_t}] \tag{3.3}$$

Where, $E(w_t)$ denotes the embedding vector of the input word w at the time step t; (w_t) is word w in the reversed sequence. The dimension of the final hidden state is two times of the embedding dimension of the input word in both unidirectional LSTMs. The embedding layer converts the each word in the input sentence to the embedding vector. When processing the i-th word in the input sentence, the input and the output of the layer are the following:

- the one-hot vector which represents i-th word.
- the embedding vector which represents i-th word

Each embedding vector is calculated by the following equation:

$$\vec{x_t} = E^{(s)} x_i \tag{3.4}$$

$E^{(s)} \in R^{D*|V^{(s)}|}$ is the embedding matrix of the encoder.

The Embedding layer is defined as the first hidden layer of a network. It must specify 3 arguments:

- **Input_dim:**This is the size of the vocabulary in the text data.

- **Output_dim:**This is the size of the vector space in which words will be embedded. It defines the size of the output vectors from this layer for each word.
- **Input_length:**This is the length of input sequences, as you would define for any input layer of a Keras model. .

LSTM gates are updated in every time step according to the following equations[**?**]

$$i_t = \sigma(W_{(i)}E_{xt-1} + b_i + W_{hi}h_{t-1}) + b_{hi} \tag{3.5}$$

$$f_t = \sigma(W_{(f)}E_{xt-1} + b_i + W_{ht}h_{t-1}) + b_f \tag{3.6}$$

$$o_t = \sigma(W_{(o)}E_{xt-1} + b_i + W_{ho}h_{t-1}) + b_{ho} \tag{3.7}$$

$$c_{temp} = \tanh(W_{(c)}E_{xt-1} + b_i + W_{hc}h_{t-1}) + b_{hc} \tag{3.8}$$

$$c_t = f_t \odot C_{t-1} + i_t \odot C_{t_{temp}} \tag{3.9}$$

$$h_t = o_t \odot tanh(c_t) \tag{3.10}$$

Where,W represents the weight matrices and b represents bias vectors respectively which are learnable parameters, t represents the current time step, x represents the input tokens to the LSTM, E represents the word embedding of the input token, and c represents the memory content of the LSTM. Both h_t and c_t are initialized as 0.

LSTM are preferred as there components because they are capable of capturing long term dependencies by overcoming the problem of vanishing gradient. Below is a typical Seq2Seq model architecture:

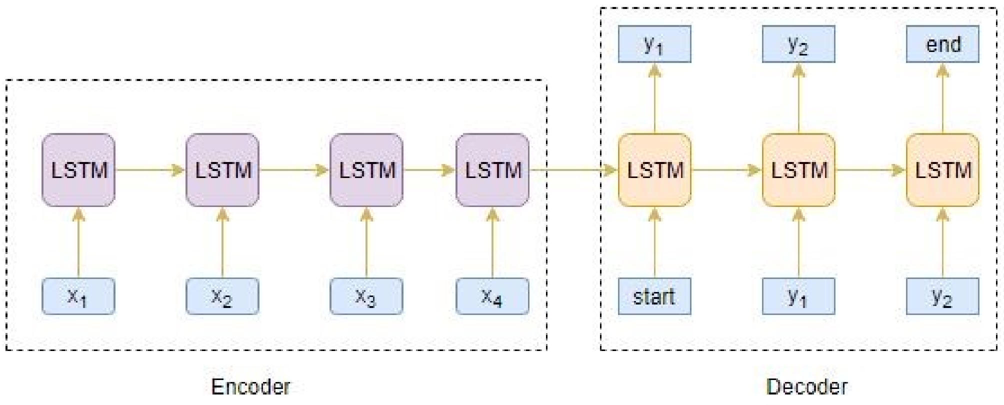


Figure 3.8: Seq2Seq model architecture[11]

The Encoder receives the input data step by step. At each time step the state of the hidden layer is looped back and combined with the input data. At the end of this procedure, the hidden layer of last time step holds a state which has been affected by all the elements in the sequence or has retained the memory of the whole sequence in a single layer. The name of Encoder originates here, because it encodes a long sequence to the state of a hidden layer which is a vector. e.g. a 200-dimensional vector.The hidden state (hi) and cell state (ci) of the last time step are used to initialize the decoder.the below diagram will illustrates this process:

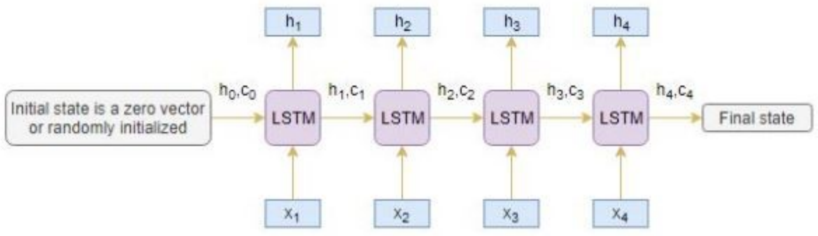


Figure 3.9: The architecture of an encoder[11]

by using the hidden state of last time step as the representation of input sequence, the encoder converts the input to a vector. this vector has been putted in the first time step of the decoder. Having the hidden state of the encoder in the first timestep of decoder, it gives an output, which is a vector, then the closest word vector are found in the dictionary to the output a word vector either. To measure how the output of that timestep matches the target word a score can be calculated from their multiplication. the probability of every output word can be calculated with that score using a normalization technique called softmax and in each the model select the most probable word in the output and put in in the input of the next layer. The algorithm will stop after entering the special token which indicates the end of the sequence is reached.

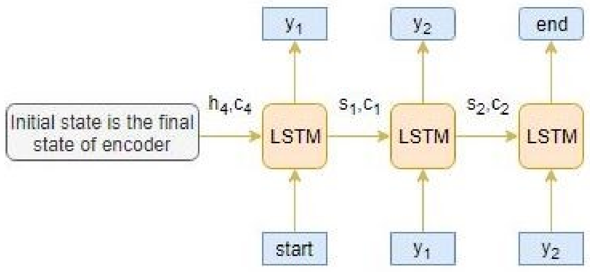


Figure 3.10: The architecture of an decoder[11]

<start> and <end> are the special tokens which are added to the target sequence before feeding it into the decoder. The target sequence is unknown while decoding the test sequence. So, we start predicting the target sequence by passing the first word into the decoder which would be always the <start> token. And the <end> token signals the end of the sentence.

3.3.8 Attention mechanism

As useful as this encoder-decoder architecture is, there are certain limitations that come with it.the encoder works only for short sequences since the decoder is looking at the entire input sequence for the prediction , the problem is in case of long sequences. It is difficult for the encoder to memorize long sequences into a fixed length vector this is where the concept of attention mechanism comes into the picture[58] . It aims to predict a word by looking at a few specific parts of the sequence only, rather than the entire sequence. Let’s consider a simple example [11] to understand how Attention Mechanism works:

- **Source sequence:** ما هي الرياضة التي تحبها انت ؟
- **Target sequence:** انا احب السباحة

The first word انا in the target sequence is connected to the fourth word انت in the source sequence, right? Similarly, the second-word احب in the target sequence is associated with the fifth word تحب in the source sequence.therefore , instead of looking at all the words in the source sequence, we can increase the importance of specific parts of the source sequence that result in the target sequence. This is the basic idea behind the attention mechanism. The Attention layer will be added to selectively choose the relevant information while discarding the non-useful information by cognitively mapping the generated sentences with the inputs of encoder layer

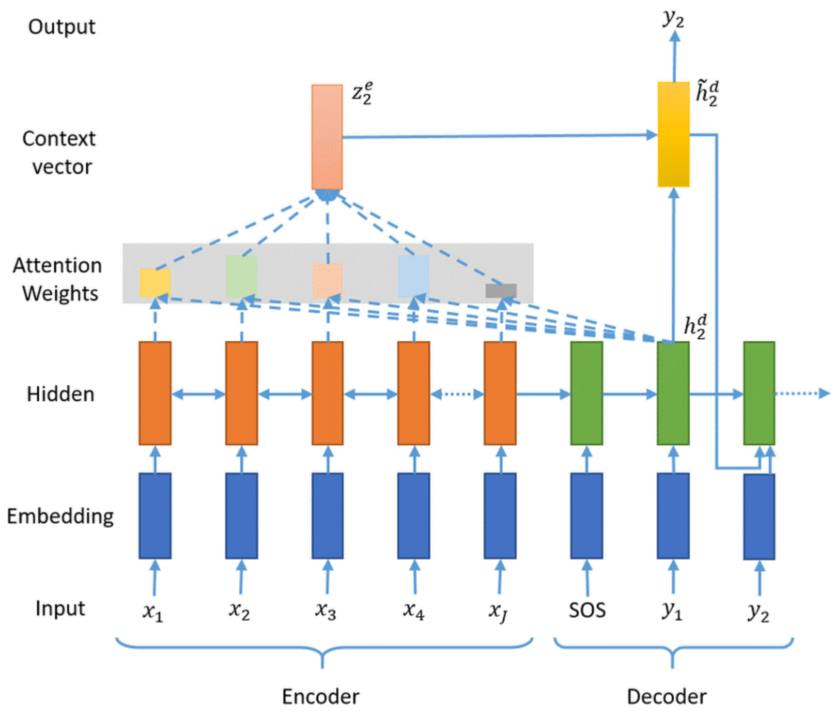


Figure 3.11: An attention-based seq2seq model[12].

The final layer to add is the Dense Layer acompained by softmax activation function whitch is mathematically represents the matrix vector multiplication in neurons and it's used to change the dimensions of the vectors for processing between various layers.

Workflow of Attention Mechanism To truly grasp how attention mechanism works leads to get through this steps:

- The encoder outputs the hidden state (h_j) for every time step j in the source sequence
- Similarly, the decoder outputs the hidden state (s_i) for every time step i in the target sequence
- We compute a score known as an alignment score e_{ij} based on which the source word is aligned with the target word using a score function. The alignment score is computed from the source hidden state hj and target hidden state si using the score function. This is given by[57]:

$$e_i j = score(s_i, h_j) \tag{3.11}$$

where $e_i j$ denotes the alignment score for the target timestep i and source time step j. There are different types of attention mechanisms depending on the type of score function used. We ve mentioned a few popular attention mechanisms below:

Name	Score function
Dot-Product	$score(s_i, h_j) = s_i^t h_j$
Additive	$score(s_i, h_j) = v_a^t \tanh(w_a[s_i, h_i])$where W_a and V_a are the trainable weight matrices
General	$score(s_i, h_j) = s_i^t w_a h_j$ where w_a is a trainble weight matrix

Table 3.2: popular attention mechanisms[11]

- Normalize the alignment scores using softmax function to retrieve the attention weights (a_{ij}) [57]:

$$a_{ij} = e^{e^{ij}} / \sum_{k=1}^{Tx} e^{e^{ik}} \tag{3.12}$$

- Compute the linear sum of products of the attention weights aij and hidden states of the encoder hj to produce the attended context vector (Ci)[57]:

$$C_i = \sum_{j=1}^{Tx} a_{ij} h_j \tag{3.13}$$

- The attended context vector and the target hidden state of the decoder at timestep i are concatenated to produce an attended hidden vector Si :

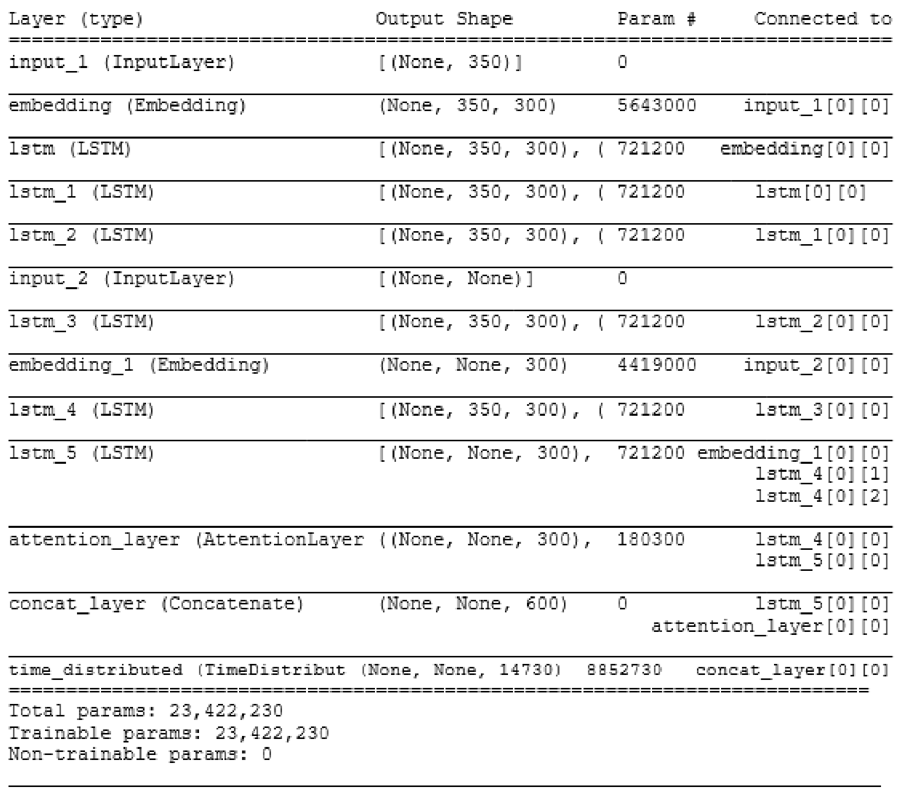
$$Si = concatenate([si; Ci]) \tag{3.14}$$

- The attended hidden vector Si is then fed into the dense layer to produce yi

$$yi = dense(S_i) \tag{3.15}$$

3.3.9 Model hyper-parameters

The parameters defined in the model an Embedding layer with a number of hidden units of 300 and a vocabulary of 350 (e.g. integer encoded words from 0 to 350, inclusive), a vector space of 32 dimensions in which words will be embedded, and 5 LTSM layers with an attention layer to solve the problem of the long sequence and concatenate layer which concatenate attention output and decoder LSTM output and the final layer is Dense layer which is considerate as an output accompained by Softmax activation function . the detaills of Hyper-parameters model are presented in the figure below .



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)            [(None, 350)]        0
__________________________________________________________________________________
embedding (Embedding)           (None, 350, 300)     5643000     input_1[0][0]
__________________________________________________________________________________
lstm (LSTM)                     [(None, 350, 300), ( 721200      embedding[0][0]
__________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 350, 300), ( 721200      lstm[0][0]
__________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 350, 300), ( 721200      lstm_1[0][0]
__________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________
lstm_3 (LSTM)                   [(None, 350, 300), ( 721200      lstm_2[0][0]
__________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 300)    4419000     input_2[0][0]
__________________________________________________________________________________
lstm_4 (LSTM)                   [(None, 350, 300), ( 721200      lstm_3[0][0]
__________________________________________________________________________________
lstm_5 (LSTM)                   [(None, None, 300),  721200      embedding_1[0][0]
                                                                 lstm_4[0][1]
                                                                 lstm_4[0][2]
__________________________________________________________________________________
attention_layer (AttentionLayer ((None, None, 300),  180300      lstm_4[0][0]
                                                                 lstm_5[0][0]
__________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 600)    0           lstm_5[0][0]
                                                                 attention_layer[0][0]
__________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 14730)  8852730     concat_layer[0][0]
==================================================================================
Total params: 23,422,230
Trainable params: 23,422,230
Non-trainable params: 0
__________________________________________________________________________________
```

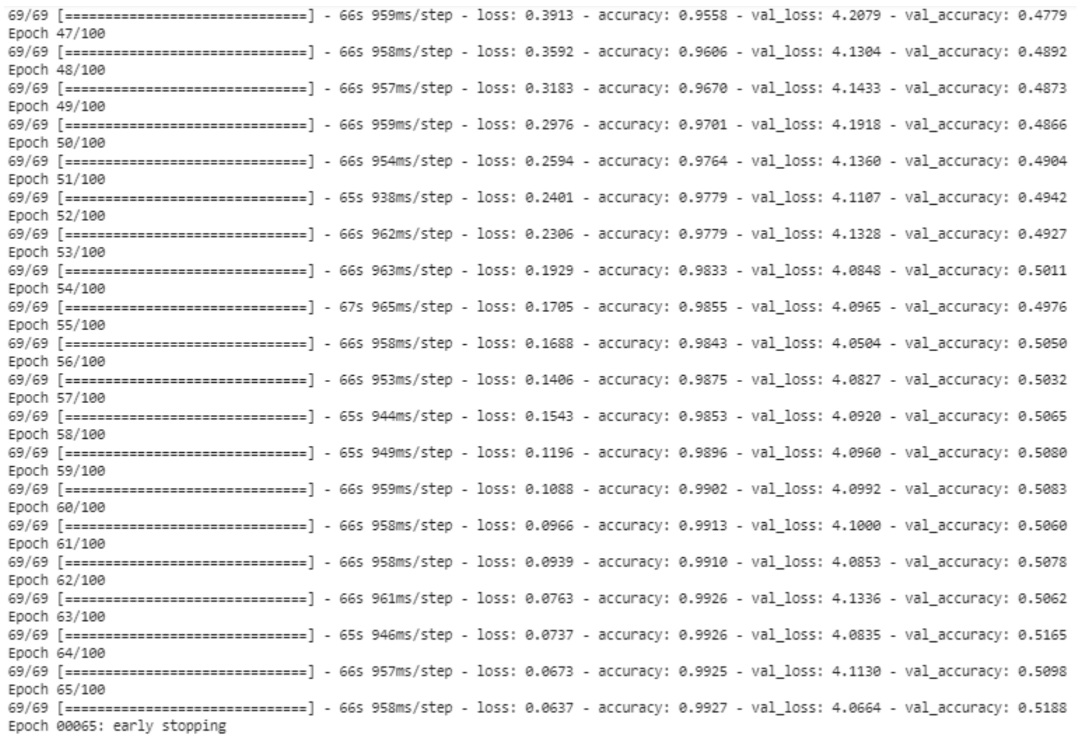
Figure 3.12: Model Hyper-parameters.

3.3.10 Model compiling

In this step and in order to compile the model. Sparse categorical cross-entropy used as the loss function. With this option an integer sequence is converted to a one-hot vector automatically which helps us to overcome memory issues. This also allow us to keep integer sequences as targets (predictions). A one-hot vector is used to distinguish each word in a vocabulary from every other word in the vocabulary.

3.3.11 Model Training

In the training phase.The process of training ML model involves providing an ML algorithm (that is the learning algorithm) with training data to learn from. The term ML model refers to the model artifact that is created by the training process.The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer desired to predict), and it outputs the model that captures these patterns.Therefore, the model are trained with a batch size set to 10 and it will be fit for 100 training epochs and the test dataset are used in the validation set in order to monitor the performance of the model on a holdout set during training, the effect will be more time between weight updates and expect faster and more stable estimates of the gradient, which should result in a more stable performance of the model during training.and When the training is complete, it is always a good idea to save the model since the training can take several hours depending on the size of the data.



```
69/69 [==============================] - 66s 959ms/step - loss: 0.3913 - accuracy: 0.9558 - val_loss: 4.2079 - val_accuracy: 0.4779
Epoch 47/100
69/69 [==============================] - 66s 958ms/step - loss: 0.3592 - accuracy: 0.9606 - val_loss: 4.1304 - val_accuracy: 0.4892
Epoch 48/100
69/69 [==============================] - 66s 957ms/step - loss: 0.3183 - accuracy: 0.9670 - val_loss: 4.1433 - val_accuracy: 0.4873
Epoch 49/100
69/69 [==============================] - 66s 959ms/step - loss: 0.2976 - accuracy: 0.9701 - val_loss: 4.1918 - val_accuracy: 0.4866
Epoch 50/100
69/69 [==============================] - 66s 954ms/step - loss: 0.2594 - accuracy: 0.9764 - val_loss: 4.1360 - val_accuracy: 0.4904
Epoch 51/100
69/69 [==============================] - 65s 938ms/step - loss: 0.2401 - accuracy: 0.9779 - val_loss: 4.1107 - val_accuracy: 0.4942
Epoch 52/100
69/69 [==============================] - 66s 962ms/step - loss: 0.2306 - accuracy: 0.9779 - val_loss: 4.1328 - val_accuracy: 0.4927
Epoch 53/100
69/69 [==============================] - 66s 963ms/step - loss: 0.1929 - accuracy: 0.9833 - val_loss: 4.0848 - val_accuracy: 0.5011
Epoch 54/100
69/69 [==============================] - 67s 965ms/step - loss: 0.1705 - accuracy: 0.9855 - val_loss: 4.0965 - val_accuracy: 0.4976
Epoch 55/100
69/69 [==============================] - 66s 958ms/step - loss: 0.1688 - accuracy: 0.9843 - val_loss: 4.0504 - val_accuracy: 0.5050
Epoch 56/100
69/69 [==============================] - 66s 953ms/step - loss: 0.1406 - accuracy: 0.9875 - val_loss: 4.0827 - val_accuracy: 0.5032
Epoch 57/100
69/69 [==============================] - 65s 944ms/step - loss: 0.1543 - accuracy: 0.9853 - val_loss: 4.0920 - val_accuracy: 0.5065
Epoch 58/100
69/69 [==============================] - 65s 949ms/step - loss: 0.1196 - accuracy: 0.9896 - val_loss: 4.0960 - val_accuracy: 0.5080
Epoch 59/100
69/69 [==============================] - 66s 959ms/step - loss: 0.1088 - accuracy: 0.9902 - val_loss: 4.0992 - val_accuracy: 0.5083
Epoch 60/100
69/69 [==============================] - 66s 958ms/step - loss: 0.0966 - accuracy: 0.9913 - val_loss: 4.1000 - val_accuracy: 0.5060
Epoch 61/100
69/69 [==============================] - 66s 958ms/step - loss: 0.0939 - accuracy: 0.9910 - val_loss: 4.0853 - val_accuracy: 0.5078
Epoch 62/100
69/69 [==============================] - 66s 961ms/step - loss: 0.0763 - accuracy: 0.9926 - val_loss: 4.1336 - val_accuracy: 0.5062
Epoch 63/100
69/69 [==============================] - 65s 946ms/step - loss: 0.0737 - accuracy: 0.9926 - val_loss: 4.0835 - val_accuracy: 0.5165
Epoch 64/100
69/69 [==============================] - 66s 957ms/step - loss: 0.0673 - accuracy: 0.9925 - val_loss: 4.1130 - val_accuracy: 0.5098
Epoch 65/100
69/69 [==============================] - 66s 958ms/step - loss: 0.0637 - accuracy: 0.9927 - val_loss: 4.0664 - val_accuracy: 0.5188
Epoch 00065: early stopping
```

Figure 3.13: Representation of training model process.

3.3.12 Model testing

After training, the model is tested on new source sequences for which the target sequence is unknown. So, we need to set up the inference architecture to decode a test sequence:

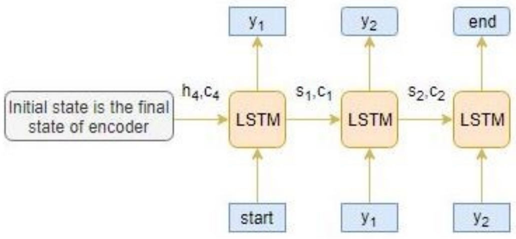


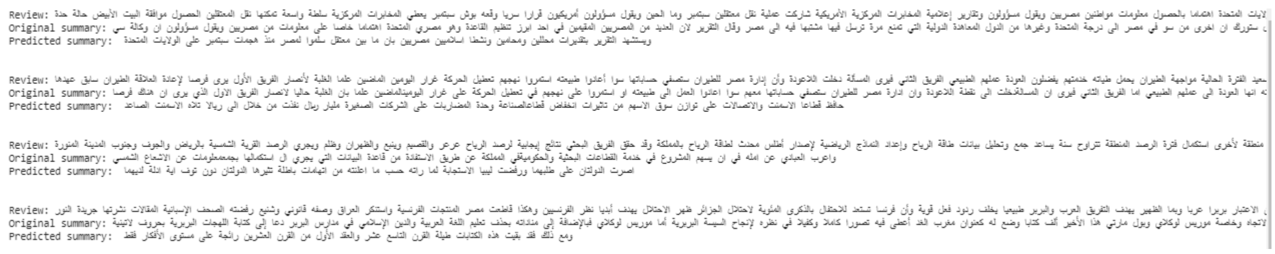
Figure 3.14: the inference architecture to decode a test sequence[11]

In machine learning, model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. The testing set consisting of a set of testing samples should be separated from the both training and validation sets, but it should follow the same probability distribution as the training set.Each testing sample has a known value of the target. Based on the comparison of the model's predicted value, y^i There are multiple steps to decode the test sequence:

- Encode the entire input sequence and initialize the decoder with internal states of the encoder
- Pass <start> token as an input to the decoder
- Run the decoder for one timestep with the internal states
- The output will be the probability for the next word. The word with the maximum probability will be selected
- Pass the sampled word as an input to the decoder in the next timestep and update the internal states with the current time step
- Repeat steps 3 – 5 until we generate <end> token or hit the maximum length of the target sequence

3.3.13 Predicition

After the Fit methods train our dataset, we will evaluate our model We can finally use our model to predict and generate the new summary ,as shown in the figure below:



```
Review: لايات المتحدة اهتماما بالحصول معلومات مواطنين مصريين ويقول مسؤولون وتقارير إعلامية المخابرات المركزية الأمريكية شاركت عملية نقل معتقلين سبتمبر وما الحين ويقول مسؤولون أمريكيون قرارا سريا وقعه بوش سبتمبر يعطي المخابرات المركزية سلطة واسعة تمكنها نقل المعتقلين الحصول موافقة البيت الأبيض حالة حدة
Original summary: ستورك ان اخرى من سو في مصر الى درجة المتحدة وغيرها من الدول المعاهدة الدولية التي تمنع مرة ترسل فيها مشتبها فيه الى مصر وقال التقرير لان العديد من المصريين المقيمين في احد ابرز تنظيم القاعدة وهو مصري المتحدة اهتماما خاصا على معلومات من مصريين ويقول مسؤولون ان وكالة سي
Predicted summary:  ويستشهد التقرير بتقديرات محللين ومحامين ونشطا اسلاميين مصريين بان ما بين معتقل سلموا لمصر منذ هجمات سبتمبر على الولايات المتحدة

Review: سعيد الفترة الحالية مواجهة الطيران يحمل طياته خدمتهم يفضلون العودة عملهم الطبيعي الفريق الثاني فيرى المسألة دخلت اللاعودة وأن إدارة مصر للطيران ستصفي حساباتها سوا أعادوا طبيعته استمروا نهجهم تعطيل الحركة غرار اليومين الماضين علما الغلبة لأنصار الفريق الأول يرى فرصا لإعادة العلاقة الطيران سابق عهدها
Original summary: ه انها العودة الى عملهم الطبيعي اما الفريق الثاني فيرى ان المسالةدخلت الى نقطة اللاعودة وان ادارة مصر للطيران ستصفي حساباتها معهم سوا اعادوا العمل الى طبيعته او استمروا على نهجهم في تعطيل الحركة على غرار اليوميالماضين علما بان الغلبة حاليا لانصار الفريق الاول الذي يرى ان هناك فرصا
Predicted summary:  حافظ قطاعا الاسمنت والاتصالات على توازن سوق الاسهم من تاثيرات انخفاض قطاعالصناعة وحدة المضاربات على الشركات الصغيرة مليار ريال نفذت من خلال الى ريالا تلاه الاسمنت الصاعد

Review: منطقة لأخرى استكمال فترة الرصد المنطقة تتراوح سنة يساعد جمع وتحليل بيانات طاقة الرياح وإعداد النماذج الرياضية لإصدار أطلس محدث لطاقة الرياح بالمملكة وقد حقق الفريق البحثي نتائج إيجابية لرصد الرياح عرعر والقصيم وينبع والظهران وظلم ويجري الرصد القرية الشمسية بالرياض والجوف وجنوب المدينة المنورة
Original summary: واعرب العبادي عن امله في ان يسهم المشروع في خدمة القطاعات البحثية والحكوميةفي المملكة عن طريق الاستفادة من قاعدة البيانات التي يجري ال استكمالها بجمعمعلومات عن الاشعاع الشمسي
Predicted summary:  اصرت الدولتان على طلبهما ورفضت ليبيا الاستجابة لما راته حسب ما اعلنته من اتهامات باطلة تثيرها الدولتان دون توف اية ادلة لديهما

Review: الاعتبار بربرا عربا وبما الظهير يهدف التفريق العرب والبربر طبيعيا يخلف ردود فعل قوية وأن فرنسا تستعد للاحتفال بالذكرى المئوية لاحتلال الجزائر ظهر الاحتلال يهدف أبديا نظر الفرنسيين وهكذا قاطعت مصر المنتجات الفرنسية واستنكر العراق وصفه قانوني وشنيع رفضته الصحف الإسبانية المقالات نشرتها جريدة النور
Original summary: لاتجاه وخاصة موريس لوكلاي وبول مارتي هذا الأخير ألف كتابا وضع له كعنوان مغرب الغد أعطى فيه تصورا كاملا وكفيلا في نظره لإنجاح السيسة البربرية أما موريس لوكلاي فبالإضافة إلى مناداته بحذف تعليم اللغة العربية والدين الإسلامي في مدارس البربر دعا إلى كتابة اللهجات البربرية بحروف لاتينية
Predicted summary:  ومع ذلك فقد بقيت هذه الكتابات طيلة القرن التاسع عشر والعقد الأول من القرن العشرين رائجة على مستوى الأفكار فقط
```

Figure 3.15: representation of prediction model process

3.4 Extractive summarization approach

In this approach, we consider text summarization as a sequence classification problem. the implemented approch is inspired from a sentiment analysis classification [59]. Extractive summarization means identifying important sections of the text and generating them and producing a subset of the sentences from the original text.The approach we adapte for automatically text summarization is supervise extractive text summarization based on classification method using neural network, This depends very much on how our data is formed as an input like each sentence of the articles will be accompanied by class which implies sentence1 is in the original summary and sent2 isn't and we will use LSTM layers to make a prediction whether the sentence should be in the summary or not and will Train it based on this to make an improvement results of the generated summary based on Rouge evaluation .The main steps followed in this classification process are presented in the following figure:

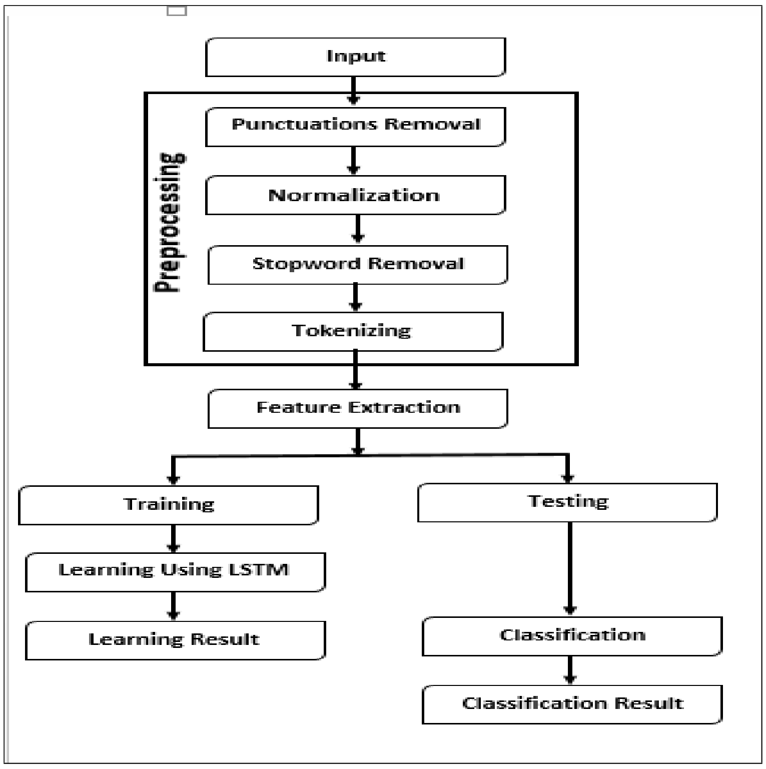
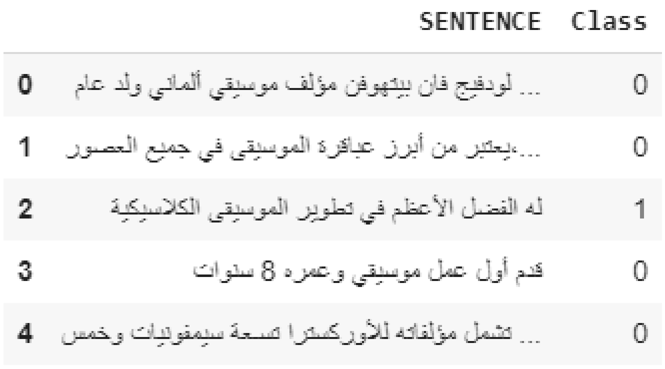


Figure 3.16: System architecture to build the model.

3.4.1 Dataset preparation

We have used the same dataset use in the abstractive approach Preparing the corpus depends very much on how the data is formed as inputs and how it going to be as an output,as an input we split every article into sentences [sent1, sent2,. . .] and each sentence will be associated with a class indicating if the sentence is a summary sentence or not according to its presence or not in the reference summaries the output should be like [0,1,1,0...] where 1 refers to sentences in the summary and 0 if not.This process will be saved in a CSV file as mentioned in the figure 3.16 :



	SENTENCE	Class
0	... لودفيج فان بيتهوفن مؤلف موسيقي ألماني ولد عام	0
1	...يعتبر من أبرز عباقرة الموسيقى في جميع العصور	0
2	له الفضل الأعظم في تطوير الموسيقى الكلاسيكية	1
3	قدم أول عمل موسيقي وعمره 8 سنوات	0
4	... تشمل مؤلفاته للأوركسترا تسعة سيمفونيات وخمس	0

Figure 3.17: Dataset description.

3.4.2 Preprocessing

Sentences pre-processing is the first step in this approach, it converts the Arabic sentences to a form that is suitable for our method . These pre-processing tasks include Punctuations removal, Latin characters removal, Digits removal and normalization. These linguistic are used to reduce the ambiguity of words to increase the accuracy and the effectiveness of our approach.

3.4.3 Tokenization

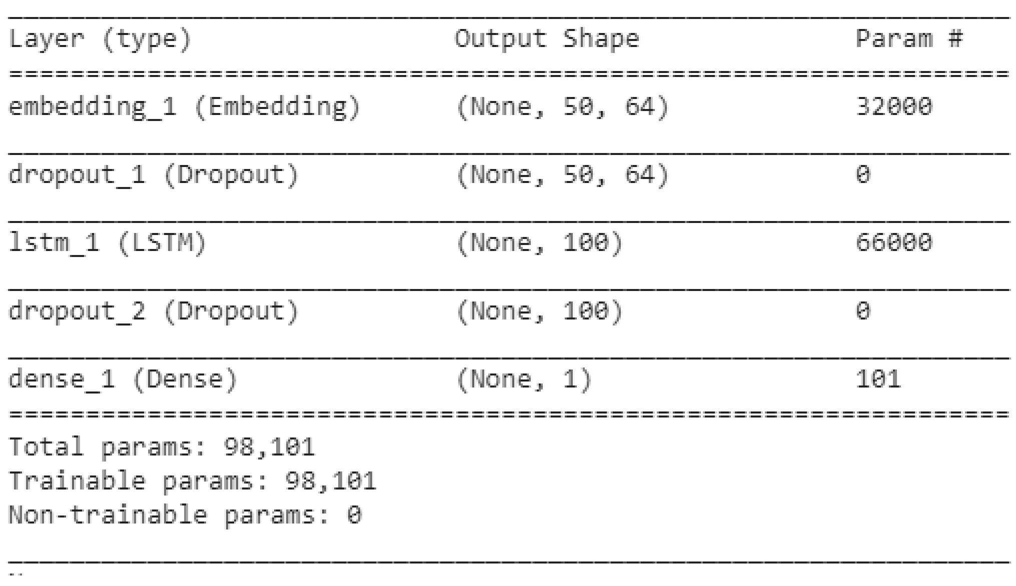
Tokenization is a method for dividing texts into tokens; Words are often separated from each other by blanks (white space, semicolons, commas, quotes, and periods). These tokens could be individual words (noun, verb, pronoun, and article, conjunction, preposition, punctuation, numbers, and alphanumeric) that are converted without understanding their meaning or relationships. The list of tokens becomes input for further processing.

3.4.4 Splitting the corpus

After preparing the EASC corpus, it will be divided into two parts. The first part is made up of 90% of the corpus used as training data. The second part, which is made up of the remaining 10% is reserved for the validation and testing of the learning model.

3.4.5 Model building :

After the dataset preparation ,it's the appropriate time to define, compile and fit the LSTM model. The first layer is the Embedded layer that uses 32 length vectors to represent each word. The next layer is the LSTM layer with 100 memory units (smart neurons). Finally, because this is a classification problem a Dense output layer are used with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (summary and no summary) in the problem. RNNs like LSTM generally have the problem of overfitting. Dropout layer can be applied between layers because It's a powerful technique for combating overfitting in your LSTM models. This can be done easily by adding new Dropout layers between the Embedding and LSTM layers and the LSTM and Dense output layers to have the desired impact on training with a slightly slower trend in convergence and a lower final accuracy. The model could probably use a few more epochs of training and may achieve a higher skill.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 50, 64)            32000     
_________________________________________________________________
dropout_1 (Dropout)          (None, 50, 64)            0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               66000     
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101       
=================================================================
Total params: 98,101
Trainable params: 98,101
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.18: Model Hyper-parameters.

3.4.6 Compile the model:

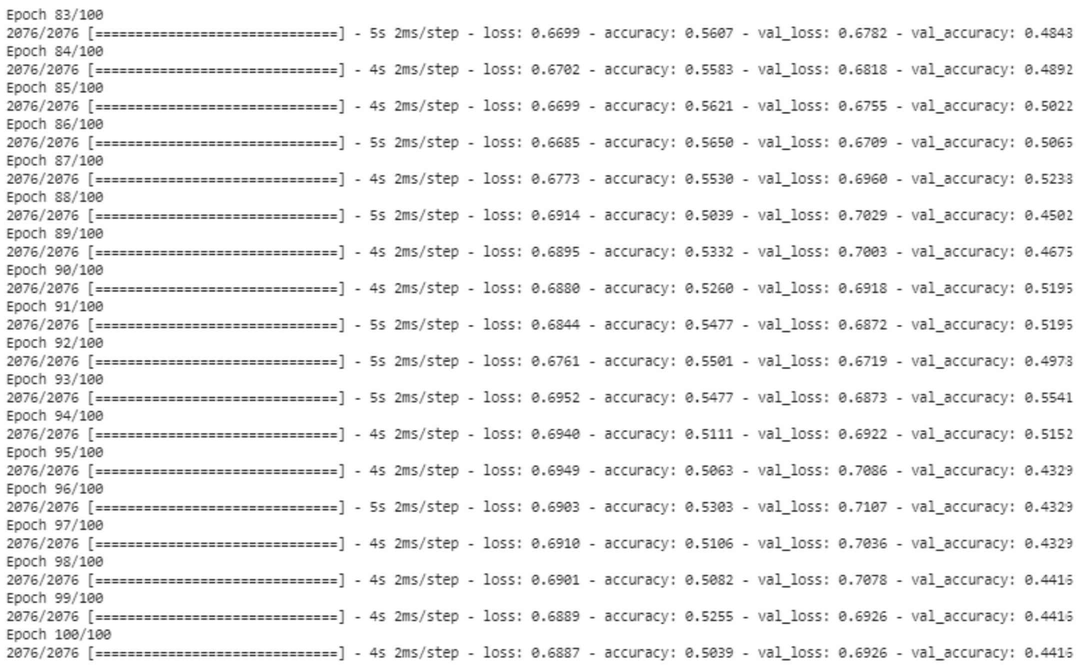
Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

- **Loss function :**This measures how accurate the model is during training. You want to minimize this function to "steer" the model in the right direction.
- **Optimizer:** This is how the model is updated based on the data it sees and its loss function.
- **Metrics:** Used to monitor the training and testing steps.

In order to compile the model and Because it is a binary classification problem, log loss is used as the loss function “binary_crossentropy”. The efficient ADAM optimization algorithm is used and the evaluation metric used is “accuracy” to monitor the training and testing phase .

3.4.7 Training the model:

In this step, the data are used to incrementally improve the model's ability to predict whether a given sentence is going to be a part of the summary or not. This process then repeats for the entire given sentences. Each step of training have an accuracy of the prediction also a loss value to evaluate the progress of learning, The model is trained for 100 epochs with a batch size of 64.Some of the results are shown in the figure below :

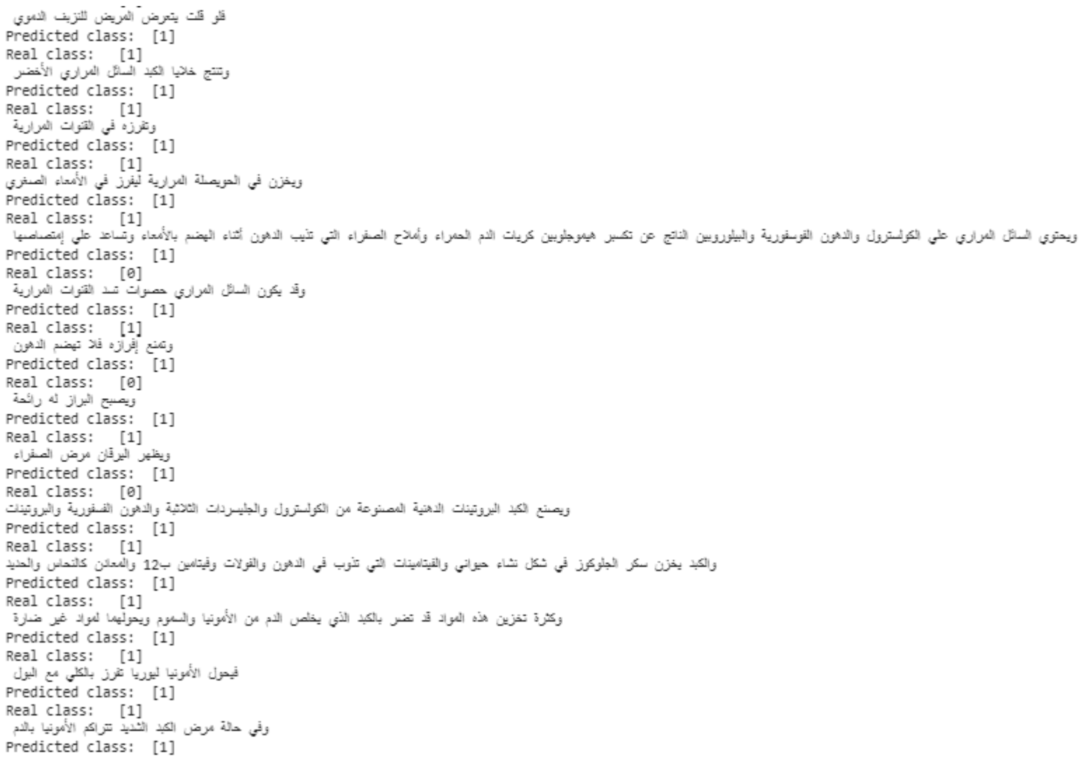


```
Epoch 83/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6699 - accuracy: 0.5607 - val_loss: 0.6782 - val_accuracy: 0.4843
Epoch 84/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6702 - accuracy: 0.5583 - val_loss: 0.6818 - val_accuracy: 0.4892
Epoch 85/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6699 - accuracy: 0.5621 - val_loss: 0.6755 - val_accuracy: 0.5022
Epoch 86/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6685 - accuracy: 0.5650 - val_loss: 0.6709 - val_accuracy: 0.5065
Epoch 87/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6773 - accuracy: 0.5530 - val_loss: 0.6960 - val_accuracy: 0.5233
Epoch 88/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6914 - accuracy: 0.5039 - val_loss: 0.7029 - val_accuracy: 0.4502
Epoch 89/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6895 - accuracy: 0.5332 - val_loss: 0.7003 - val_accuracy: 0.4675
Epoch 90/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6880 - accuracy: 0.5260 - val_loss: 0.6918 - val_accuracy: 0.5195
Epoch 91/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6844 - accuracy: 0.5477 - val_loss: 0.6872 - val_accuracy: 0.5195
Epoch 92/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6761 - accuracy: 0.5501 - val_loss: 0.6719 - val_accuracy: 0.4973
Epoch 93/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6952 - accuracy: 0.5477 - val_loss: 0.6873 - val_accuracy: 0.5541
Epoch 94/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6940 - accuracy: 0.5111 - val_loss: 0.6922 - val_accuracy: 0.5152
Epoch 95/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6949 - accuracy: 0.5063 - val_loss: 0.7086 - val_accuracy: 0.4329
Epoch 96/100
2076/2076 [==============================] - 5s 2ms/step - loss: 0.6903 - accuracy: 0.5303 - val_loss: 0.7107 - val_accuracy: 0.4329
Epoch 97/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6910 - accuracy: 0.5106 - val_loss: 0.7036 - val_accuracy: 0.4329
Epoch 98/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6901 - accuracy: 0.5082 - val_loss: 0.7078 - val_accuracy: 0.4416
Epoch 99/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6889 - accuracy: 0.5255 - val_loss: 0.6926 - val_accuracy: 0.4416
Epoch 100/100
2076/2076 [==============================] - 4s 2ms/step - loss: 0.6887 - accuracy: 0.5039 - val_loss: 0.6926 - val_accuracy: 0.4416
```

Figure 3.19: the results of training the model.

3.4.8 Prediction:

The trained model are evaluated for the test set to check the accuracy, finally the model are used to predict whether a given sentence is in the summary or not ,the result are shown in the figure below:



```
 فلو قلت يتعرض المريض للنزيف الدموي
Predicted class:  [1]
Real class:   [1]
 وتنتج خلايا الكبد السائل المراري الأخضر
Predicted class:  [1]
Real class:   [1]
 وتفرزه في القنوات المرارية
Predicted class:  [1]
Real class:   [1]
ويخزن في الحويصلة المرارية ليفرز في الأمعاء الصغري
Predicted class:  [1]
Real class:   [1]
 ويحتوي السائل المراري علي الكولسترول والدهون الفوسفورية والبيلوروبين الناتج عن تكسير هيموجلوبين كريات الدم الحمراء وأملاح الصفراء التي تذيب الدهون أثناء الهضم بالأمعاء وتساعد علي إمتصاصها
Predicted class:  [1]
Real class:   [0]
 وقد يكون السائل المراري حصوات تسد القنوات المرارية
Predicted class:  [1]
Real class:   [1]
 وتمنع إفرازه فلا تهضم الدهون
Predicted class:  [1]
Real class:   [0]
 ويصبح البراز له رائحة
Predicted class:  [1]
Real class:   [1]
 ويظهر اليرقان مرض الصفراء
Predicted class:  [1]
Real class:   [0]
ويصنع الكبد البروتينات الدهنية المصنوعة من الكولسترول والجليسردات الثلاثية والدهون الفسفورية والبروتينات
Predicted class:  [1]
Real class:   [1]
والكبد يخزن سكر الجلوكوز في شكل نشاء حيواني والفيتامينات التي تذوب في الدهون والفولات وفيتامين ب12 والمعادن كالنحاس والحديد
Predicted class:  [1]
Real class:   [1]
 وكثرة تخزين هذه المواد قد تضر بالكبد الذي يخلص الدم من الأمونيا والسموم ويحولهما لمواد غير ضارة
Predicted class:  [1]
Real class:   [1]
 فيحول الأمونيا ليوريا تفرز بالكلي مع البول
Predicted class:  [1]
Real class:   [1]
 وفي حالة مرض الكبد الشديد تتراكم الأمونيا بالدم
Predicted class:  [1]
```

Figure 3.20: Representation of prediction model process

3.5 Conclusion :

In this chapter, we have presented our contribution for Arabic text summarization. we have implemented two different approaches. The first one concerns a deep abstractive approach. It is based on RNN-LSTM seq2seq model. The second one is a deep extractive approach and in which we have modeled the summarization task as a classification problem.The implementation details and the evaluation will be described in the following section.

Chapter 4

Implementation and Evaluation

4.1 Introduction

Evaluating the quality of automatically produced summaries is subjective (there is no "perfect" summary), rather difficult and remains an open problem.To make a good summary it has to be comparable and evaluated to so called good summary. To do so, there are a variety of possible bases for the comparison of summarization systems performance.Therefore, the system summary can be compared to the source text, to a human-generated summary or to another system summary. In terms of evaluating summaries on sentence level can be done semi-automatically by measuring content overlap with precision, recall, and F1 measure. An extracted sentence is considered acceptable if the same sentence was extracted in a reference summary. This process cannot be automatized because reference summaries are created by human judges. Some semi-automatic evaluation methods used nowadays are: Rouge that will be discussed and used as the method of evaluation in this chapter .In What follows of this section we will show the progress and the results of our text summarization approaches in the test phase, which will help us to consider the possible improvements.We start by presenting and describing the methods used in the evaluation. And then we present the evaluation result for the abstractive approach and the extractive one. In a further experiment, we will compare the obtained result with those obtained by TextRank algorithm and an extractive fine-tuned version of BERT transformer.

4.2 Software Configuration for implimentation

Cloud computing services cover a vast range of options now, from the basics of storage, networking, and processing power through to natural language processing and artificial intelligence as well as standard office applications. Pretty much any service that doesn't require you to be physically close to the computer hardware that you are using can now be delivered via the cloud.In the recent years the only solutions for performing huge computations, was using supercomputers which use arrays of CPUs to do many computations in parallel. Therefore in this work we use Google Colab. To make the executions faster , it has free Jupyter notebook environment that runs completely on Google Cloud and uses Google Compute Engine backend for all the computations. It is very easy to write and execute the code as it does not require any installation on your local machine. All we need is a web browser to access the colaboratory. Colab offers both GPU and TPU hardware accelarators for free with our Google free tier account up to a certain computational power.That is, we can change the runtime type while executing our code. The free GPU offered is 1xTesla K80, 2496 CUDA cores and free TPU includes TPU v2, 8 cores, approx 12 GB RAM with a maximum RAM limit up to 36 GB, Consequently for our implementation we have used the GPU accelarator.

4.3 Experimental Results and Evaluation

Experiments are conducted on EASC [54]. The dataset is formed of 153 Arabic articles and 765 human-generated extractive summaries of those articles.. The articles were collected from Arabic Wikipedia and news paper. The selected articles cover different topics: art and music, environment, politics, sports, health, finance and insurance, science and technology, tourism, religion and education. We evaluated summarization quality automatically using ROUGE [34]. We report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

4.3.1 Rouge

The Rouge measure [34] is based on the Bleu metrics used in machine translation tasks. The idea is to compare the differences between the distribution of words in the candidate summary and the distribution of words in the reference summaries. Given h reference summaries and a candidate summary they are split into n-grams to calculate the intersection of n-grams between the references and the candidate. This process is illustrated in figure 4.1 .

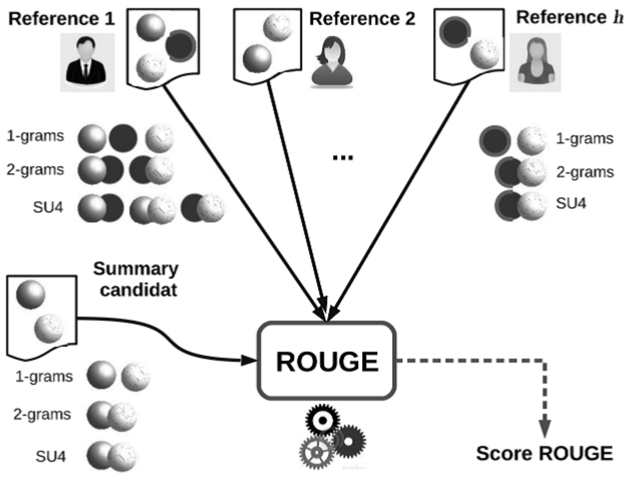


Figure 4.1: The basic idea of ROUGE for evaluation of summaries [13].

4.3.1.1 ROUGE-N

Formally, ROUGE-N metric is calculated from the co-occurrences of n-grams between the candidate and reference summaries as shown by formula 5.1 in [13] where the numerator is the maximum number of co-occurrences of ngrams in both reference and candidate summary and the denominator is the total sum of the number of n-grams present in the reference summaries.

- **ROUGE-1:** refers to the overlap of 1-gram (each word) between the system and reference summaries.
- **ROUGE-2:** refers to the overlap of bigrams between the system and reference summaries.

4.3.1.2 ROUGE-L

measures the longest sequence of words that matches between the system generated summary and the reference summary. LCS (Longest Common Subsequence) is used to find the longest sequence, which allows in-sequence matches rather than consecutive matches.ROUGE-L can be computed in following way :

$$R_{lcs} = \frac{\sum_{i=1}^{u} LCS_u(r_i, C)}{m} \tag{4.1}$$

$$P_{lcs} = \frac{\sum_{i=1}^{u} LCS_u(r_i, C)}{n} \tag{4.2}$$

$$F_{lcs} = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{4.3}$$

Where,
R_{lcs} :Recall score
P_{lcs} :Precision score
F_{lcs} :F-Measure
r_i :reference summary sentence
C :candidate summary
m :number of words in reference summary
n :number of words in candidate summary
β :it is the factore that controls relative importance between precision and recall scores

4.4 Result of abstractive summarization model

We successfully managed to implement the attentive encoder-decoder RNN abstractive summarizer, we trained the Model on a small data for most of its training time, and for 100 epochs on 100,000 sentence and size batch set to 10 with 22,448,217 parameter. We used 300-dimensional encoder and decoder cells, 32-dimensional vectors, and a vocabulary size of 350. The loss decreased from 7.6503 to 0.0637.and validation accuracy increased from 0.1946 to 0.5188 this results means that the model was very well trained during almost 65% of the entire training time and it is capable of predecting and generating new summary with 51% chance of taking the right choices to predect new summaries.the progress of training are presented in the figure 4.2:

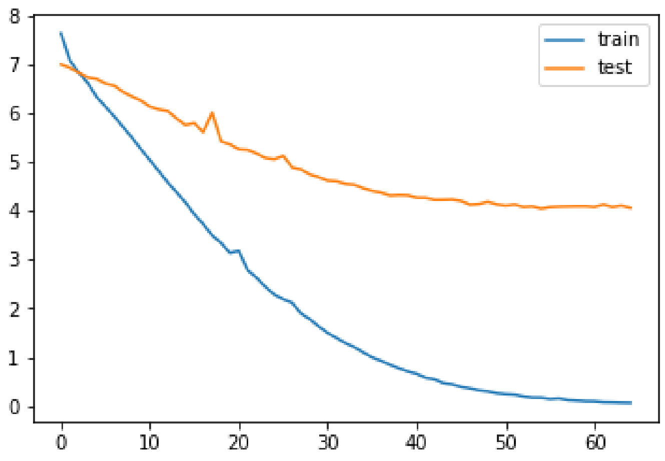


Figure 4.2: Representation of the behavior of the model over time .

4.5 Evaluation experiments on abstractive summarization model

In this section we present the result for supervised models for summarization. For this puproses we trained our models on 688 human extracted summaries and evaluated on 77 human extracted summaries. Figure 4.3 compiles the performance result for abstractive summarization strategies. Because of the lack of our dataset content number the results of abstractive LSTM-RNN's was a bit low in all metrics. For example, Rouge- 1 ,Rouge- 2 and Rouge- L are given for Recall 0.21 , 0.14 and 0.19 respectively, and for Precision the results was 0.24,0.16 and 0.21 respectively.We observe that the precision ratio is higher than recall in all metrics which means that the system do take the best choices every time for the generate summaries but the ratio is lower than precision ratio which also means that produced summaries are less informative but it capture a good amount of important information in the articles. In term of fluency Rouge-L results has proven that the generate summaries are written in well formed for Arabic langage .

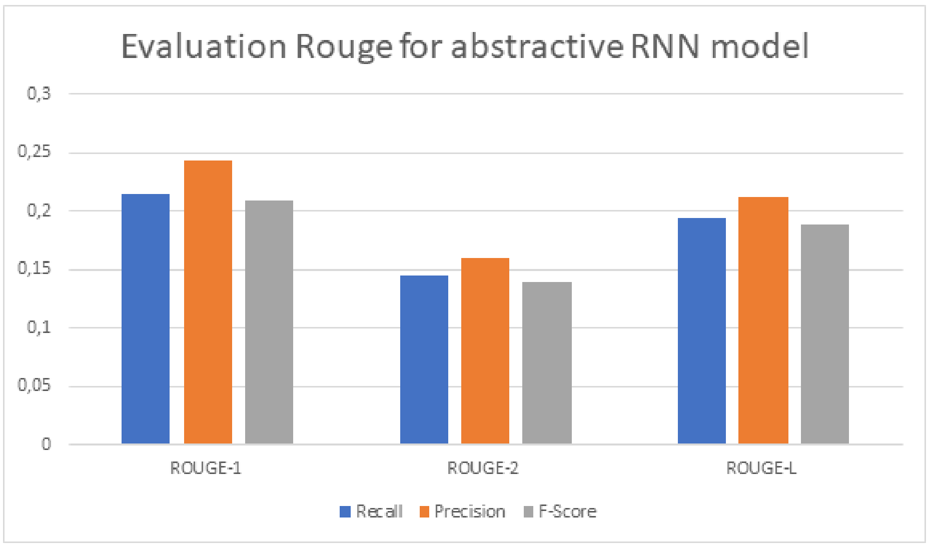
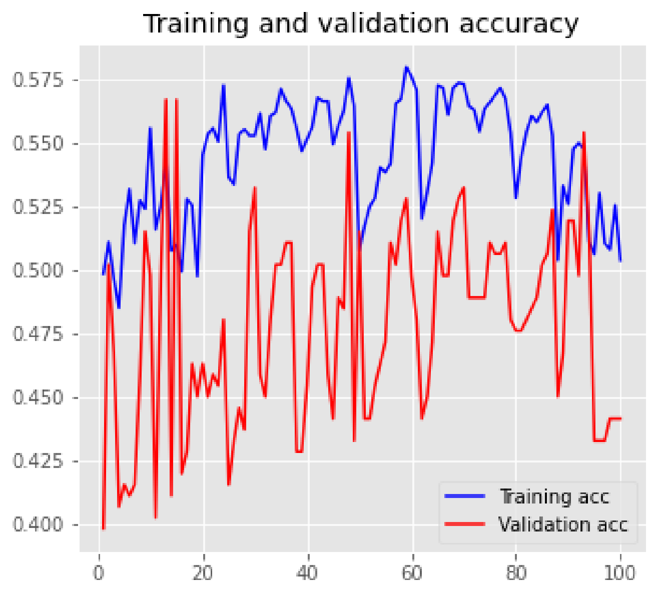


Figure 4.3: ROUGE evaluation for abstractive summrization model.

4.6 Result of extractive summarization model

For the recurrent neural network component in the sentence classification extractor The embedding layer size in the model was 500 with vocabulary size of 50 and an embedding vector size of 64 we have also used a single-layered LSTM network with size 100.. We performed binary cross-entropy training with a batch size of 64 for 100 training epochs. It took around half an hour on google collab GPU to train. After each epoch, we evaluated our model on the validation set and chose the best performing model for the test set. As shown in the Figure below the graph curve gets significant dips in both the training and validation accuracy and loss during the entire training time.the loss decreased from 0.6938 to 0.6887.and validation accuracy increased from 0.3983 to 0.4416. this results means that the model was well trained and it is capable of predecting and generating new summary with 44.16% chance of taking the right choices to predect whether given sentence is in the summary or not.



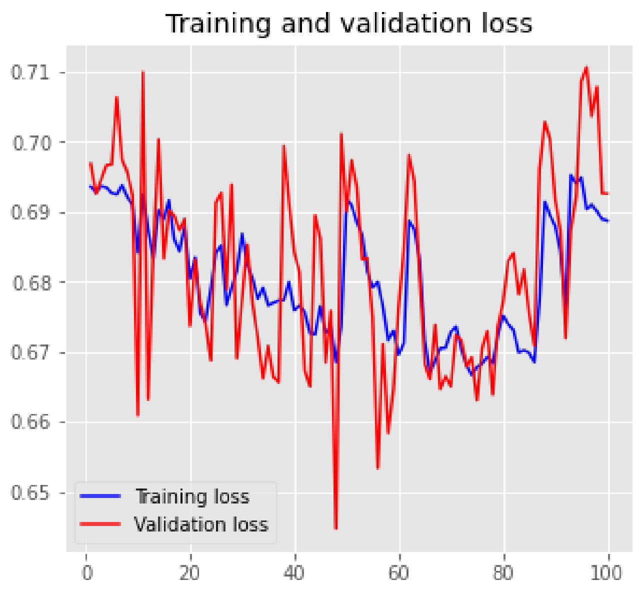


Figure 4.4: diagnostic plots to understand the behavior of the model over time

4.7 Evaluation experiments on extractive summarization model

As presented in the Figur 4.5 we evaluate extractive summarization model based on classification RNN method. The model are doing a good job of separating summary-sentences from non-summary sentences.but for supervised RNN's extractive model should outperform other techniques and it is obviously this model are getting behind on Rouge metric because the process of sentence labeling wasn't perfoming very well for classification the articles and the main reason why is some special characteres in our data was interrupting the classification process for example "أ" (hamza) .As result the Rouge-1 , Rouge-2 and Rouge-L are given at Recall 0.36 , 0.23 and 0.33 respectively, and at Precision 0.35,0.23 and 0.32 respectively. We notice that recall ratio is higher than precision ratio in all metrics which means that produced summaries are more informative and capable of extracting the valuable informations from the original text and less precise in making important decisions and In term of fluency Rouge-L results has proven that the generate summaries are written in well formed for Arabic langage .

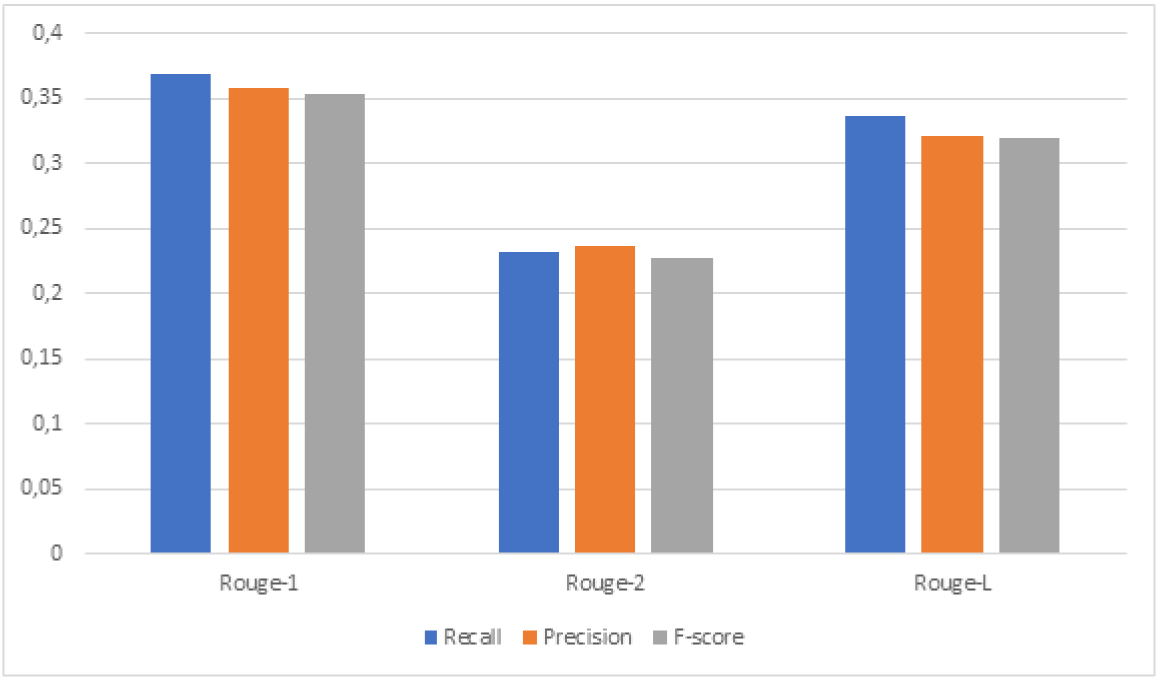


Figure 4.5: ROUGE evaluation for extractive classifier RNN model

4.8 Discussion

In this section we compare the result of testing abstractive and extractive summarization model on ROUGE metrics .According to the results that was mentioned in Figure 4.3 and 4.5 .Extractive Model outperform the abstractive model with significant difference in all metrics .the extractive summaries are more informative and much related to the main topics and very well labeled because whereas the abstractive summaries were written in a good form but less precise at predicting the right words in the right places and introducing unknown words out of nowhere , this is very likely due to the huge lack of data in our corpus that was needed from the RNN model but in the end it was a good results for an abstractive model to be able to do such a performance .

4.9 Further experiment

In a further experiment and in order to compare the obtaned result with other systems and algorithms, we have implemented two algorithms. the first one is the TextRank and the second one is a fine-tuned model based on Bert transformer. TextRank is a text summarization technique which is used in Natural Language Processing to generate Document Summaries.TextRank uses an extractive approach and is an unsupervised graph-based text summarization technique [60]. BERT (Bidirectional transformer) is a transformer used to overcome the limitations of RNN and other neural networks as Long term dependencies. It is a pre-trained model that is naturally bidirectional. This pre-trained model can be tuned to easily to perform the NLP tasks [61].

4.9.1 Experiment with Text-Rank

In this experiment we measured the generated summaries from text-Rank summarizer with human summaries from the angle of the studied evaluators. The correlation results can be found in Figure 4.6 , We can observe that when comparing summaries with human abstracts, ROUGE measures demonstrate the best performance. The measure showing the best correlation was ROUGE-1 with no significant differences in results with ROUGE-2 and ROUGE-L and the reason is the abstractors usually put in the abstract some words not contained in the original text and this can make the main topics of the abstract and an extractive summary different. Another reason is that the abstracts were sometimes not long enough to find the main topics and therefore to use all terms in evaluation, as ROUGE does.

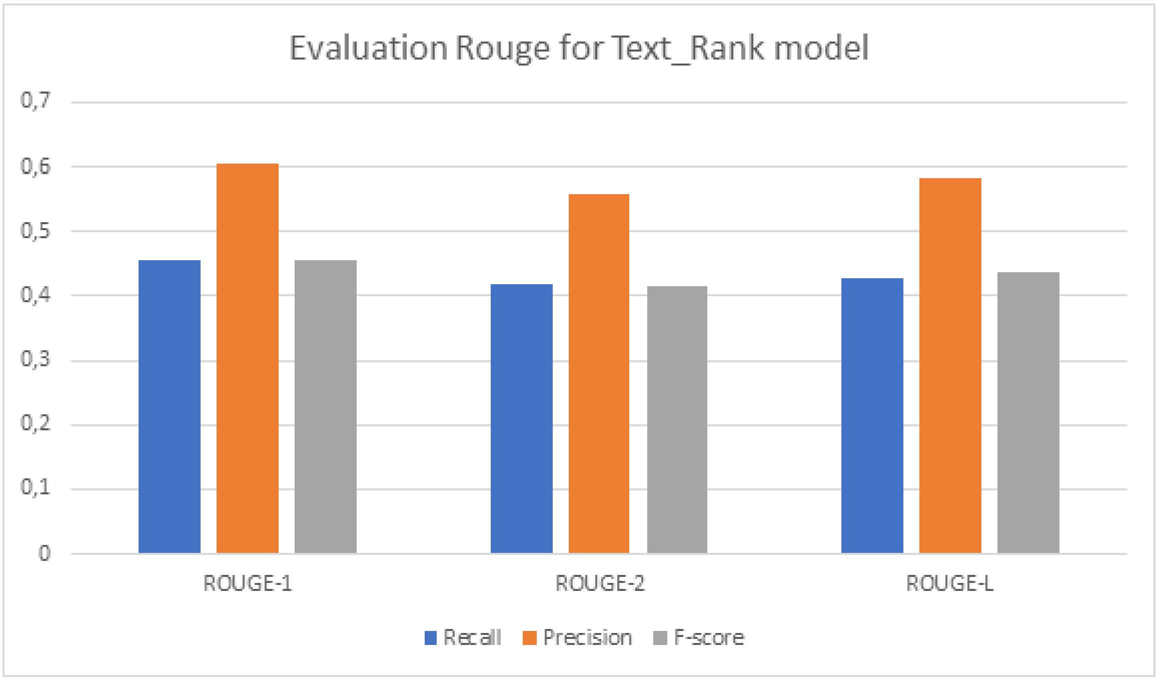


Figure 4.6: ROUGE evaluatione for Text_Rank model

4.9.2 Experiment with BertSumm-RNN :

The experimental results on EASC corpus are shown in Figure 4.7 . The implemention was based BERT architecture with small parameters randomly initialized and only trained on the summarization task. Bert model architecture has 6 layers, the hidden size is 512 and the feed-forward filter size is 2048. The model is trained with same settings following [62]. as illustrated in the Figure BERTSUM with LSTM achieved a good performance on all three metrics. The BERTSUM model does not have an obvious influence on the summarization performance. The experiment proved that the result of ROUGE-1 and ROUGE-L are simulaire and better than ROUGE 2 and the result was logical because from the definition of ROUGE-N and ROUGE-L. In other part, we observe that precision ratio is higher than recall ratio in all metrics which means that produced summaries are less informative (low recall) but the precision ratio is higher than recall which means that the system make the good choices every time.

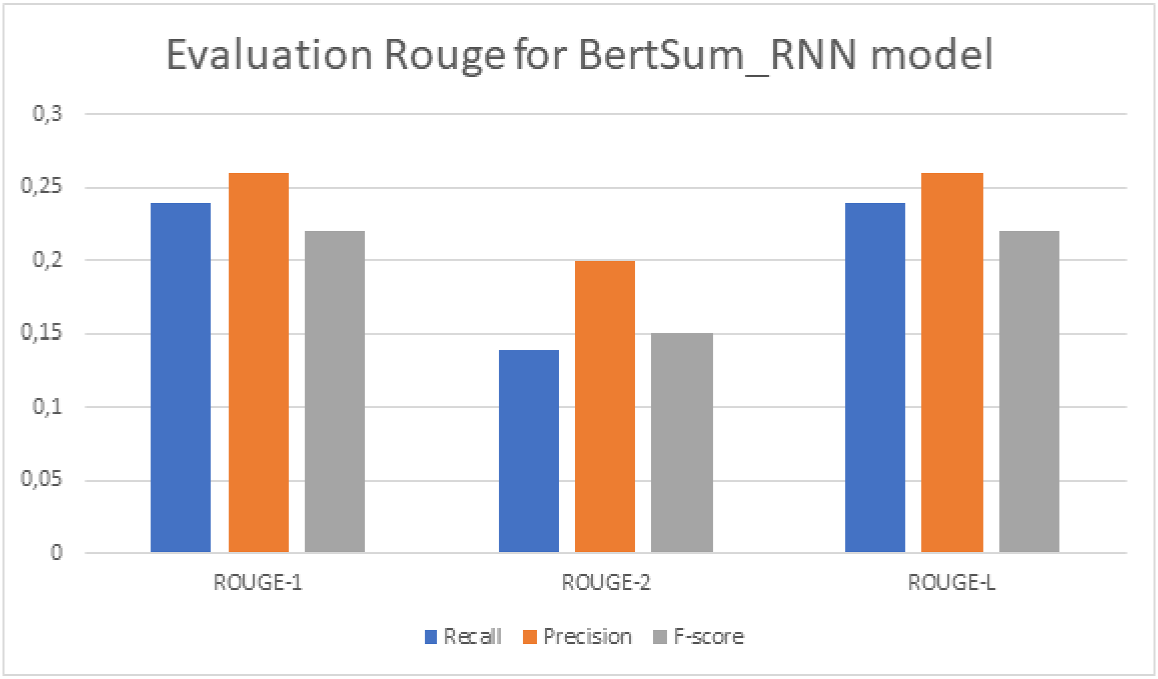


Figure 4.7: ROUGE evaluation for BertSum-RNN model

4.10 Overall discussion :

The experiments were done using EASC data set with a various models such as BERT-RNN,Text-Rank,extractive, abstractive models. As mentioned before ROUGE was used to evaluate the summaries. ROUGE-1, ROUGE-2, ROUGE-L were used as the evaluation metric under ROUGE. The results are shown in Appendix A and the average results for the participants models is shown in table 4.1. As seen from the table, for comparing the ROUGE results of our systems with those of the BERT summarization based on RNN and Text-Rank systems , we found that our systems results are very close to them. The results obtained by the Text-rank summarizer are comparable and slightly better than the extractive summaries based on classification RNN this is very logical because the human summaries are based on extractive method ,Even thought the classification extractive summaries outperform both bert-RNN and abstractive summaries in all terms of redundancy and fluency and informativeness .For the abstractive summries that were shown as the less performing results Out of all systems summaries due the lack of the amount data that needed from the model to be well trained and much performing . Regarding the other systems , they all performed good and These results should only be taken to be indicative.However,Text-Rank appears to be a better method for extractive summaries as we are comparing summaries on the sentence level. It is worth noting that the main results obtained from abstractive summarizer are in line with results

Model	Rouge-1	Rouge-2	Rouge-L
Abstractive-RNN	0.22	0.15	0.19
Extractive classification	0.36	0.23	0.33
Text-Rank	0.51	0.46	0.48
BERT-RNN	0.24	0.16	0.24

Table 4.1: ROUGE evaluation for various model.

4.11 Conclusion :

This chapter illustrates briefly the process of implementing the adopted systems by specifying the software configurations , platforms , the environment features used, and the approaches we took to achieve our purposes. In the second part ,we presented the experimental results where we notice that performance with the classification method on extractive approach showed a higher quality than the abstractive approach and we realize that the lack of resources can be a serious problem particularly for evaluation purposes,. As a future work, we consider to enhance the informative scoring technique by implementing Bi-Directional LSTM which is capable of capturing the context from both the directions and results in a better context vector, increase the training dataset size and build the model and Implement pointer-generator networks and coverage mechanisms.

conclusion

Automatic text summarization is the data science problem of creating a short, accurate, and fluent summary from a longer document. Humans are generally good at this type of task as it involves first understanding the meaning of the source document and then distilling the meaning and capturing salient details in the new description and the main goal of this thesis was automatically creating abstractive and extractive summaries of arabic text and have the resulting summaries as good as those written by humans with The most recent researches raised in this field through the last years. At first, different approaches of automatic text summarization are described including the main appoaches, as well as the basics techniques and methodologies of natural language processing. After that the various methods of deep learning also the different types of neural network used in ATS are presented along with the recurrent neural network (RNN) architecture as the adopted model architecture for the main approaches. Finally a contribution based on adapting,testing and comparing both deep abstractive and extractive approaches when dealing with Arabic text are managed. As a conclusion, Techniques and methodologies for Arabic text summarization are still immature due to the inherent complexity of the Arabic language in terms of both structure and morphology and it's still in its initial stage compared to the work done in English and other languages . In this thesis ,We adapted two approaches of automatic summarization conducted on an arabic text ,both of them used LSTM-RNN models. The first one is based on an abstractive model where the second one is based on extractive model using sentences classification. Summarization process starts with preprocessing, division on traning and testing dataset and then fitting and testing models for both the approaches.then the resulting summaries are evaluated with a human summaries throught Rouge metrics.the results obtained are compared to other recent and modern works aiming to improve there quality. This work has been very helpfull for us due the rich information and knowledge gained throughout our researchs and studies . we've emerged a lot of computer science field such as :Artificial Intelligence with software development .It gave us the opportunity to implement python programming language to explore the large field of ATS with deep learnig and made us realise what we need to improve our work for Arabic language in the near future .As future work, we intend to improve and test the built models on other corpora. The following oppotunity is the design of Transformer-based approches for Arabic text summarization.

Appendix A

Scores for the models participating in our experiments

ROUGE-Type	Recall	Precision	F-Score
ROUGE-1	0.215339	0.243158	0.209191
ROUGE-2	0.144716	0.16052	0.139661
ROUGE-L	0.194564	0.212636	0.188108

Table A.1: ROUGE measure scores of abstractive model.

ROUGE-Type	Recall	Precision	F-Score
ROUGE-1	0.368118	0.357599	0.353338
ROUGE-2	0.232441	0.236253	0.227331
ROUGE-L	0.335687	0.320738	0.319749

Table A.2: ROUGE measure scores of extractive model.

ROUGE-Type	Recall	Precision	F-Score
ROUGE-1	0.455264	0.605173	0.455129
ROUGE-2	0.417513	0.557828	0.413817
ROUGE-L	0.428564	0.58214	0.437193

Table A.3: ROUGE measure scores of Text-Rank model.

ROUGE-Type	Recall	Precision	F-Score
ROUGE-1	0.24738	0.26239	0.22693
ROUGE-2	0.14691	0.20263	0.153232
ROUGE-L	0.24738	0.26239	0.22693

Table A.4: ROUGE measure scores of Bert-RNN model.

Bibliography

[1] N. Alami, M. Meknassi, S. A. Ouatik, and N. Ennahnahi, “Arabic text summarization based on graph theory,” pp. 1–8, 2015.

[2] A. Thomas, “Speech recognition and hidden markov models.” [Online]. Available: https://www.slideserve.com/amber/speech-recognition-and-hidden-markov-models

[3] K. Kaikhah, “Text summarization using neural networks,” 2004.

[4] aimadhu. (2015) Supervised and unsupervised learning. [Online]. Available: https://dataaspirant.wordpress.com/2014/09/19/supervised-and-unsupervised-learning/

[5] V. Alto. (2019-07-06) Neural networks: parameters, hyperparameters and optimization strategies. Accessed 2020-03-25. [Online]. Available: https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5

[6] How does convolutional neural network work. Accessed 2020-03-25. [Online]. Available: https://mc.ai/how-does-convolutional-neural-network-work/

[7] (2018) Sentiment analysis with supervisioned and unsupervisioned learning. Accessed 2020-03-26. [Online]. Available: https://mc.ai/sentiment-analysis-with-supervisioned-and-unsupervisioned-learning/

[8] S. Yan. (2017-11-15) Understanding lstm and its diagrams. Accessed 2020-03-26. [Online]. Available: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

[9] M. Phi. (2018) Illustrated guide to lstm’s and gru’s: A step by step explanation. Accessed 2020-03-27. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[10] S. Esmaeilzadeh, G. X. Peh, and A. Xu, “Neural abstractive text summarization and fake news detection,” *arXiv preprint arXiv:1904.00788*, 2019.

[11] A. PAI. (2019-06-10) Comprehensive guide to text summarization using deep learning in python. Accessed 2020-04-20. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/

[12] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *arXiv preprint arXiv:1812.02303*, 2018.

[13] J.-M. Torres-Moreno, *Automatic text summarization.* John Wiley & Sons, 2014.

[14] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, "Text summarization techniques: a brief survey," 2017.

[15] H. P. Luhn, "The automatic creation of literature abstracts," 1958.

[16] M. E. R. Barzilay, *Using lexical chains for text summarization*, 1999.

[17] Y. Jaya Kumar, O. S. Goh, H. Basiron, H. C. Ngo, Suppiah, and P.C, "A review on automatic text summarization approaches," vol. 12, pp. 178–190, 01 2016.

[18] S. l. Nabiha.Azizi, Mohammed Redjimi, "Automatic arabic text summarization approaches," 2017.

[19] D. P. O. J. M. Conroy, "Text summarization via hidden markov models," 2001.

[20] L. Rabiner and B. Juang, "An introduction to hidden markov models," 1986.

[21] R. Nag, K. Wong, and F. Fallside, "Script recognition using hidden markov models," 1986.

[22] S. L. Cailan Zhou, "Research of information extraction algorithm based on hidden markov model," 2010.

[23] T. Nomoto, "Bayesian learning in text summarization," 2005.

[24] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," 1995.

[25] B. Larsen, "A trainable summarizer with knowledge acquired from robust nlp techniques," 1999.

[26] Z. Ghahramani, *AN introduction to hidden markov models and bayesian networks*, 2001.

[27] M. L. Nguyen, S. Horiguchi, A. Shimazu, and B. T. Ho, "Example-based sentence reduction using the hidden markov model." 2004.

[28] K. Kaikhah, "Automatic text summarization with neural networks," 2004.

[29] A. A. Al-Ajlan, H. S. Al-Khalifa, and A. S. Al-Salman, "Towards the development of an automatic readability measurements for arabic language," 2008.

[30] R. V. Sumit Sahu, Bharti Dongre, "Web spam detection using differentfeatures," 2011.

[31] N. Boukhatem, "The arabic natural language processing: Introduction and challenges," 2014.

[32] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," 2009.

[33] H. Saggion and T. Poibeau, "Automatic text summarization: Past, present and future," in *Multi-source, multilingual information extraction and summarization.* Springer, 2013, pp. 3–21.

[34] C.-Y. Lin and F. Och, “Looking for a few good metrics: Rouge and its evaluation,” in *Ntcir Workshop*, 2004.

[35] R. Katragadda, “Gems: generative modeling for evaluation of summaries,” 2010.

[36] K. P. S. R. T. Ward and J. H. F. Reeder, “Corpus-based comprehensive and diagnostic mt evaluation: Initial arabic, chinese, french, and spanish results,” 2002.

[37] A. Nenkova, R. Passonneau, and K. McKeown, “The pyramid method: Incorporating human content selection variation in summarization evaluation,” 2007.

[38] D. Gillick, B. Favre, and D. Hakkani-Tür, “The icsi summarization system at tac 2008.” in *Tac*, 2008.

[39] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi, “Extractive summarization using continuous vector space models,” 2014.

[40] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*, 1996.

[41] R. S. Sutton, A. G. Barto *et al.*, “Introduction to reinforcement learning.” 1998.

[42] J. Schmidhuber, “Deep learning in neural networks: An overview,” 2015.

[43] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung, I. Yen *et al.*, “Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes,” 2019.

[44] Y. LeCun, Y. Bengio *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[45] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, *Recurrent neural networks for language understanding.*, 2013, 2013.

[46] W. Lalouani, “Neural information retrieval.”

[47] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural computation*, vol. 1, no. 2, pp. pages:270–280, 1989.

[48] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Siam, 2008, vol. 105.

[49] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with lstm recurrent networks,” *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[51] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[52] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 3, pp. 1264–1268, 2013.

[53] M. Wadhwa. (2018-11-16) seq2seq model in machine learning). Retrieved 2019-12-17. [Online]. Available: https://www.geeksforgeeks.org/seq2seq-model-in-machine-learning/

[54] M. El-Haj. "essex arabic summaries corpus. [Online]. Available: http://www.lancaster.ac.uk/staff/elhaj/corpora.htm

[55] A. PAI. (2020-03-23) Comprehensive guide to text summarization using deep learning in python. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/

[56] V. Srividhya and R. Anitha, "Evaluating preprocessing techniques in text categorization," *International journal of computer science and application*, vol. 47, no. 11, pp. page:94–51, 2010.

[57] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[58] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[59] J. Brownlee. (2016-07-26) Sequence classification with lstm recurrent neural networks in python with keras. Accessed 2020-05-02. [Online]. Available: https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/?fbclid=IwAR1jUIuV6eIe2skU0C6gzUly32RdTTagwCpvwof5u-Up3uhQEgwiY8CytJQ

[60] P. JOSHI. (2018-11-01) An introduction to text summarization using the textrank algorithm (with python implementation). Accessed 2020-05-15. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

[61] Y. Liu, "Fine-tune bert for extractive summarization," *arXiv preprint arXiv:1903.10318*, 2019.

[62] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.