

**Faculty of Computer Studies  
Information Technology & Computing**

**SummarizeIt**

**(text summarizer with ML&DL)  
  
Submitted to  
Arab Open University  
  
Supervised by  
Dr. /** **Mustafa Abdul-Salam**Presented by **Awad Safwat**

**ID: 1851710330**

**Cairo  
2021/2022**

|  |  |
| --- | --- |
|  | **Declaration of No Plagiarism**  I hereby declare that this submitted report work is a result of my own efforts and I have not plagiarized any other person's work. I have provided all references of information that I have used and quoted in my work.  Name of Student: **Awad Safwat Rady**  Signature: **Awad Safwat**  Student ID: **1851710330**  **Abstract.**  Text Summarization is the task of extracting salient information from the original text document, and it's one of the most important tasks of NLP (Natural Language Processing). In this process, the extracted information is generated as a condensed report and presented as a concise summary to the user. It is very difficult for humans to understand and interpret the content of the text. In this project we will work on the problem of text Summarization , using ML & DL algorithms. the data we will use to train the model in this project is "Amazon Fine Food reviews " from kaggle.  Acknowledgements  I needed a lot of help and asked a lot of questions to finish this project, and my professor, Dr. Mustafa Abdul-Salam, always responded, even if the time was not appropriate, so he has my gratitude and respect for everything he did for me in terms of assistance and encouragement to complete this project. I'd want to express my gratitude to Dr. Eid Emary for his assistance. and i will never forget the help from the ENG ( "Mohamed Anwar" and "karem Eldepaisy"). and I'd want to thank the AOU for everything it has done to help me get to the last course, which is graduation  project.    Table of Contents  [**Chapter One:** Introduction**.** 11](#_Toc102927262)  [**1.1** **Overview** 12](#_Toc102927263)  [**1.2** **Motivations of the project** 13](#_Toc102927264)  [**1.3** **Problems of the project** 14](#_Toc102927265)  [**1.4** **Aims and Objectives** 15](#_Toc102927266)  [**1.5** **Scope and Constraints of the project** 16](#_Toc102927267)  [**1.6** **Suggested Solution** 17](#_Toc102927268)  [Fig1. 1 Solution process. 17](#_Toc102927269)  [**1.7** **Project Plan** 18](#_Toc102927270)  [Fig1. 2 Project plan. 18](#_Toc102927271)  [Fig1. 3 Gantt Chart. 18](#_Toc102927272)  [**Chapter Two:** Literature Review 19](#_Toc102927273)  [**2.0** **Background Information** 20](#_Toc102927274)  [**2.1** **Similar applications** 23](#_Toc102927275)  [**2.2** **Disadvantages of similar apps** 25](#_Toc102927276)  [**Chapter Three**: Requirements and Analysis 26](#_Toc102927277)  [3.1 Functional Requirements / use case 27](#_Toc102927278)  [3.2 Non-Functional Requirements 27](#_Toc102927279)  [1- Usability: 27](#_Toc102927280)  [3.3 Software Tools & Hardware Specifications 28](#_Toc102927281)  [3.4 Analysis Diagrams 29](#_Toc102927282)  [**Chapter Four:** Design , Implementation and testing 34](#_Toc102927283)  [References 75](#_Toc102927284) |

|  |
| --- |
| **Declaration of No Plagiarism**  I hereby declare that this submitted report work is a result of my own efforts and I have not plagiarized any other person's work. I have provided all references of information that I have used and quoted in my work.  Name of Student: **Awad Safwat Rady**  Signature: **Awad Safwat**  Student ID: **1851710330**  **Abstract.**  Text Summarization is the task of extracting salient information from the original text document, and it's one of the most important tasks of NLP (Natural Language Processing) . In this process, the extracted information is generated as a condensed report and presented as a concise summary to the user. It is very difficult for humans to understand and interpret the content of the text. In this project we will work on the problem of text Summarization , using ML & DL algorithms. the data we will use to train the model in this project is "Amazon Fine Food reviews " from kaggle.  Acknowledgements  I needed a lot of help and asked a lot of questions to finish this project, and my professor, Dr. Mustafa Abdul-Salam, always responded, even if the time was not appropriate, so he has my gratitude and respect for everything he did for me in terms of assistance and encouragement to complete this project. I'd want to express my gratitude to Dr. Eid Emary for his assistance. and i will never forget the help from the ENG ( "Mohamed Anwar" and "karem Eldepaisy"). and I'd want to thank the AOU for everything it has done to help me get to the last course, which is graduation  project.  Table of content.  ***Contents***  [1. Chapter One: Introduction. 11](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111065)  [1.1 Overview 12](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111066)  [1.2 Motivations of the project 13](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111067)  [1.3 Aims and Objectives](file:///F:\\AOU\\summarizer\\report%20Pa\\Project%2520Sample.docx" \l "_Toc40111069) **[Error! Bookmark not defined.](file:///F:\\AOU\\summarizer\\report%20Pa\\Project%2520Sample.docx" \l "_Toc40111069)**  [1. 4Scope and Constraints of the project 16](file:///F:\\AOU\\summarizer\\report%20Pa\\Project%2520Sample.docx" \l "_Toc40111070)  [1.5Project Plan **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111072)  [Project plan. 18](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111073)  [Gantt Chart. 18](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111074)  [2. Chapter Two: Literature Review 19](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111075)  [2.1 Background Information 20](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111076)  [2.2 Similar applications 20](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111077)  [3. Chapter Three: Requirements and Analysis 26](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111078)  [3.1 Functional Requirements 27](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111079)  [3.2 Non-Functional Requirements 27](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111080)  [3.3 Software Tools & Hardware Specifications 28](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111081)  [3.4 Analysis Diagrams 29](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40111082)  ***List of figures:***  [Fig1. 1 Solution process.. 18](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112437)  [Fig1. 2 Project Plan. 18](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112438)  [Fig2. 1 Text Summarizer **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112446)  [Fig2. 2 Text Summarizer\_AI **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112447)  [Fig2. 3 Sumlt! Text Summarization **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112448)  [Fig2. 4 Summarizer **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112449)  [Fig3. 1 Use Cases. **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112453)  [Fig3. 2 Use case Diagram. **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112454)  [Fig3. 3 Sequence Diagram. **Error! Bookmark not defined.**](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112456)  [Fig3. 4 Block Diagram 31](file:///F:\AOU\summarizer\report%20Pa\Project%2520Sample.docx#_Toc40112460) |

# 

# **Chapter One:** Introduction**.**

## **Overview**

**“SummarizIt”** (Text Summarizer using DL) is a flutter application that does the task of summarizing a text entered by the user by scanning an image and extracting the text from it, or uploading a file , using one of the ML algorithms and then display the summarized text and can pronounce it .

The automatic text summarizing system creates a summary, condensed version of the text that only includes a few key sentences. Text summarization began in the late 1950s, and the discipline has progressed significantly since then. In this field of study, a large number of techniques and approaches have been developed.

An automatic text summarizer should produce a summary that contains the most important information in a document while also taking up less space than the original document. Automatic summary production, on the other hand, is a difficult undertaking. When summarizing a large number of documents, various concerns such as duplication, temporal dimension, sentence sequencing, and so on require special attention, making the process more difficult. (Halifax, 2017)

For example:

The original text is : “Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".”

The summary is : “Not as Advertised”

## **Motivations of the project**

These days we feel that time is passing quickly and there is not enough time to spend our daily need completely, because of the many daily requirements of each of us, so each person should try to create enough time for himself and not waste his time on things that are supposed to take less time than usual .

Such as reading the daily news, or reading an entire article to reach specific information and not needing the rest of the article,

Or that the person is supposed to see important texts and there is not enough time to do so.

All these reasons and more made my project something that has become a constant presence in the life of each of us, and these reasons are considered a great incentive to complete this project.

## **Problems of the project**

Nowadays anyone can be a publisher, and this is with the help of the Internet and social media. It has also become very easy to publish different books and articles daily in large numbers and in different languages, as well as the large number of different sites in very similar specialties and topics, all of this has led to an increase in the volume of data There are very large daily on the Internet, these data are difficult to read all in order to extract some of the information from it, also sometimes we need to analyze this data to know whether it meets the ethical standards or not.

Therefore, we must try to find automatic solutions to this problem.

## **Aims and Objectives**

In this project, we will seek to create an application that works using deep learning algorithms, to summarize texts in a way similar to the way humans summarize texts.

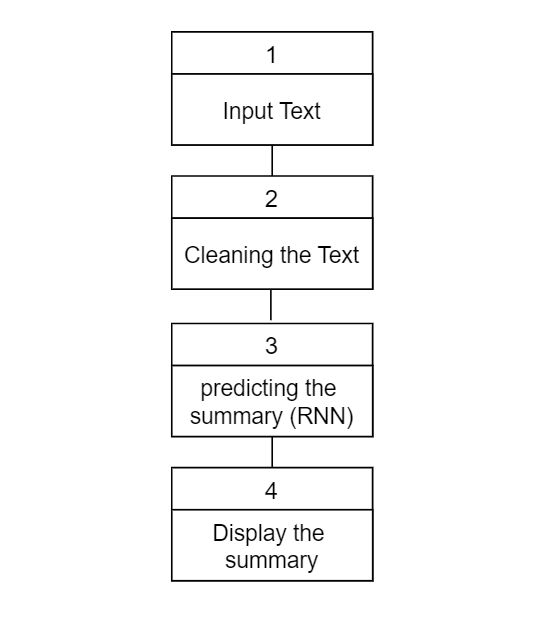
We also strive to make this application work accurately and quickly.

## **Scope and Constraints of the project**

In this project, we target readers in general and academics in particular, and we will seek to develop the algorithm so well that we can rely on the program to summarize scientific and research articles.

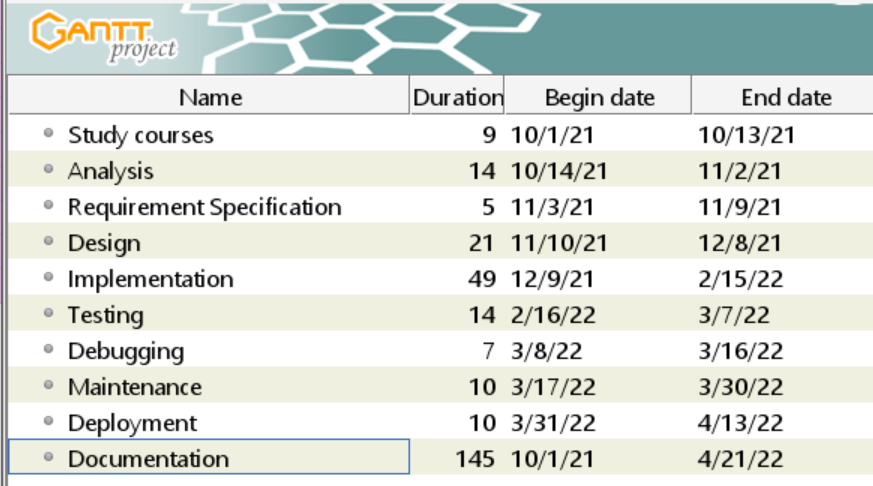
## **Suggested Solution**

The targeted solution to solve this problem, is that the target text is received from the user first, then that text is cleaned to be ready to work on, after that the process of tokenization and transformation into matrices for that text, after which the matrices are inserted into a set of recurrent neural networks ( RNN) that has previously been trained on data similar to this text, in turn predicts the summary text of that text



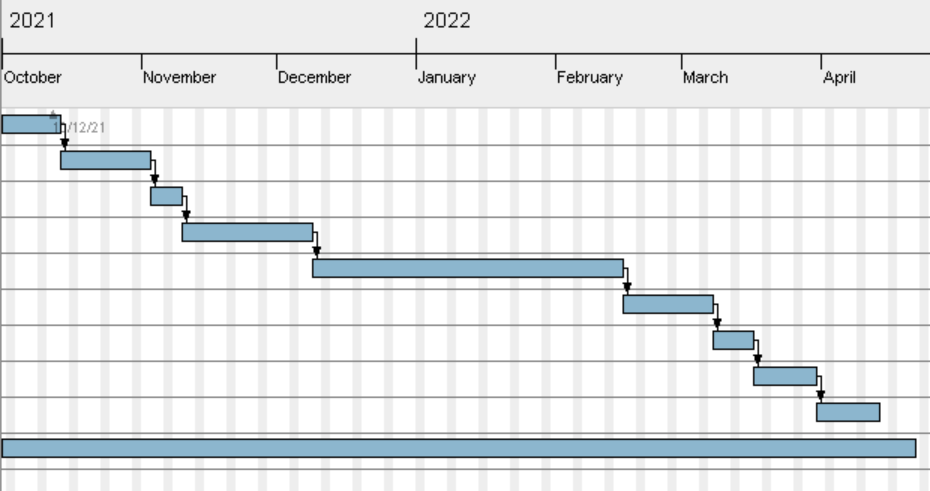
## Fig1. 1 Solution process.

## **Project Plan**



## Fig1. 2 Project plan.

The process sequence of this system is depicted in the diagram above. Starting with the analysis phase, when all components of the project are investigated in order to define the requirements for the design, implementation, and testing stages to be implemented. The user interface and navigation for the application are introduced during the design phase. The test types are applied according to software testing principles in the code for the back-end application and the AI model. All faults and errors detected during the test phase will be fixed after this debug and maintenance phase.



## Fig1. 3 Gantt Chart.

# **Chapter Two:** Literature Review

## **Background Information**

People are getting overwhelmed by data as more and more digitalized text becomes available, especially as the Internet grows. It becomes critical to figure out how to assist individuals in successfully and efficiently extracting information from data. Text summarization is one of several strategies that have been developed to achieve this aim.

Since the 1950s, text summary has existed in some form. two major factors have dominated study in this field. A emphasis on methods for creating extracts from scientific articles has emerged as a result of work in library science, office automation, and information retrieval, including the use of "shallow" language analysis and term statistics.

The other factor has been artificial intelligence research, which has looked at "shallow" knowledge-based ways for condensing data. While there are still some issues to be resolved, extraction-based approaches have made significant development in the recent decade. (devopedia, 2020)

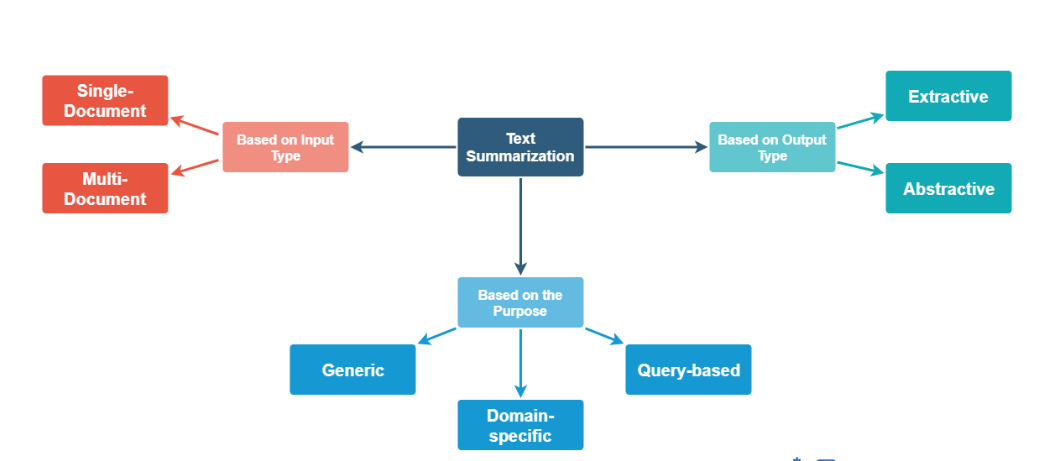


Fig2. 1 Types of text summarization.

Since extraction is frequently utilized, such as the snippets provided by search engines, and it is considerably easier than abstracting, most contemporary text summarizers are extractive. As a result, this post will concentrate on extraction. Extractive summarizers (whether for single-document or multi-document summarizing) often have to overcome three problems:(i) Content selection, or what should be selected from the text to form the summaries, which are typically in the form of sentences or phrases; (ii) Information ordering, or how to order the extracted sentences or phrases; (iii) Sentence realization, or how to clean up the extracted sentences or phrases so that they form a coherent summary. The first issue is clearly the most important for extractive summarizers. (Shen, 2009)

**Abstractive Summarizers:**

Abstractive summarizers are so-called because they do not select sentences from the originally given text passage to create the summary. Instead, they produce a paraphrasing of the main contents of the given text, using a vocabulary set different from the original document. This is very similar to what we as humans do, to summarize.In our heads, we generate a semantic representation of the document. Then we choose terms from our general vocabulary (words we frequently use) that meet the semantics to construct a brief summary that summarizes the entire content. As you can see, creating this type of summarizer may be tough because it would require Natural Language Generation.

Let's have a look at the most common solution to the problem.

Application of sequence-to-sequence RNNs

The approach was proposed in a paper by Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, Bing Xiang from IBM.

The term “sequence to sequence models” is used because the models are designed to create an output sequence of words from an input sequence of words. The input sequence in the considered case is the actual text document and the output sequence is the shortened summary. (Vashisht, 2019)

In this project the model used will be attentional Recurrent Neural Network encoder-decoder model which was first proposed for machine translation by Dzmitry Bahdanau, Jacob’s University, Germany.

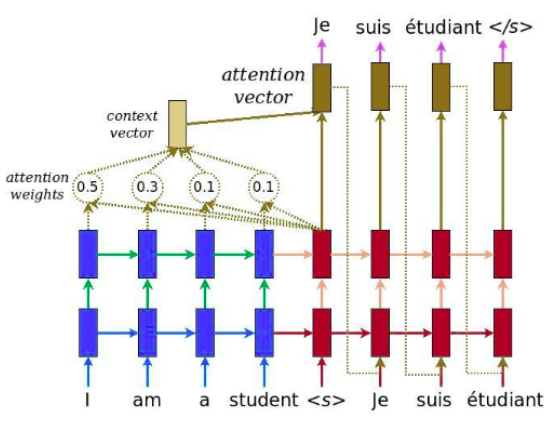


Fig2. 2 Attentional-encoder-decoder

## **Similar applications**

1. (Text Summarizer)

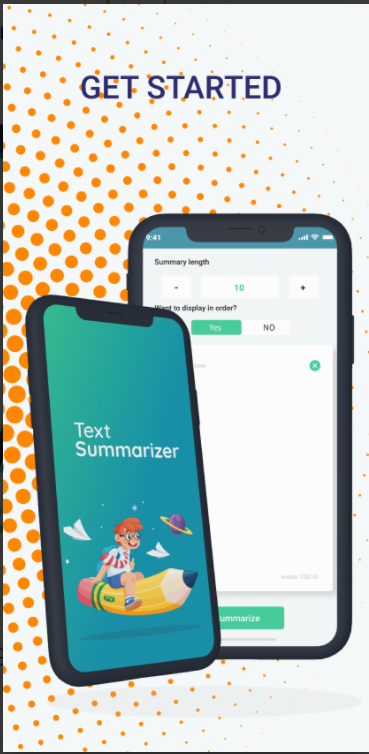


Fig2. 3 Text Summarizer

Text summarizer is an android app that helps you to summarize the educational books ,

You can use such app to summarize a text for four times free then you need to pay for it to use ,

1. (Text Summarizer\_AI)

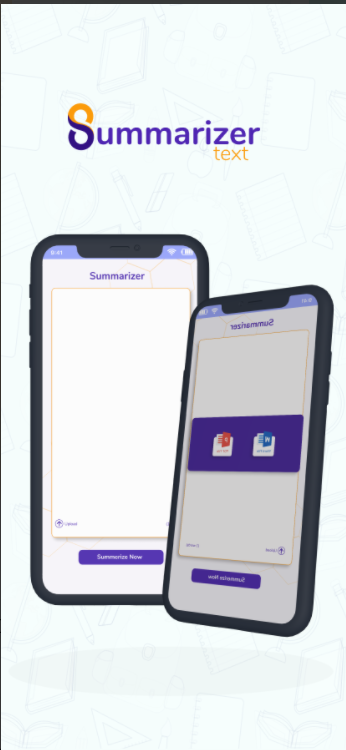


Fig2. 4 Text Summarizer\_AI

Text Summarizer\_AI is an IOS app that helps you to summarize the wiki articles , you can copy past the summary of the text and you can upload a file from your devise , the app is based on extracting model .

## **Disadvantages of similar apps**

1. The Accuracy: the accuracy of such apps is not good at all

You will fined a lot of problems on the summary

Like : repeated words , wrong summary

1. Privacy : such apps asks you about sum of privet information

Put the mechanism of the app not need it

1. Mechanism : there a lot of errors that happen suddenly while using such apps like , disconnect to the model.
2. Need a lot of money to use that apps many times

Our app (SummarizeIt ) will solve these problems with the future versions

# **Chapter Three**: Requirements and Analysis

## Functional Requirements / use case

* User must be able to open the camera and scan a text from image
* User must be able to Upload file from the memory
* User must be able to listen to the summarized text
* The application must be connected to the internet
* The application must be able to connect to the Summarizer model
* The application must be able to send a text and receive the summarized text from the Summarizer model
* The application must be able to pronounce (text to speech) the summarized text
* The application must be able to run on multiple OS

Fig3. *1* Use Cases

## Non-Functional Requirements

## Usability:

the system should be easy to use and simple to understand:

* Ease of use: Ease of Entering a text with any way of the allowed.
* Easy to work with the result text.

1. Security :

* Using different kind of firewall systems.
* Encrypting the network in high level of encryption.
* The system should be secure and saving people privacy.

1. Maintainability :

* Continuous enhancement, improvement, and development of the system
* There must be periodic maintenance on the servers and devices from time to time (ranging from one to 3 months)
* Ease and speed of handling errors.

## Software Tools & Hardware Specifications

* Mobile device specification:

Supporting Android Platform and IOS

(Android 4.0.4 (Ice Cream Sandwich), 4.3 (Jelly Bean)).

Ram: 1 GB

CPU Quad-core 1.4 GHz Cortex-A9

* Software Requirements:

Operating System : Windows 10 Home Premium x64

• IDE : Visual Studio Code

• Browser : Google Chrome

* Technologies and Programing Languages:

• Dart

• Flutter

• Google Colab

• TensorFlow

• Keras

• NLP

• Recurrent Neural Network (RNN)

• Machine Learning

• Deep Learning

## Analysis Diagrams

* + 1. Use case Diagram**:**

A use case diagram is a graphic representation of the interactions between users and the various aspects of a system. A use case is a strategy for identifying, clarifying, and organizing system needs in system analysis. The use cases are the roles that the actors (User, Admin, and mobile application) play within and around the system.

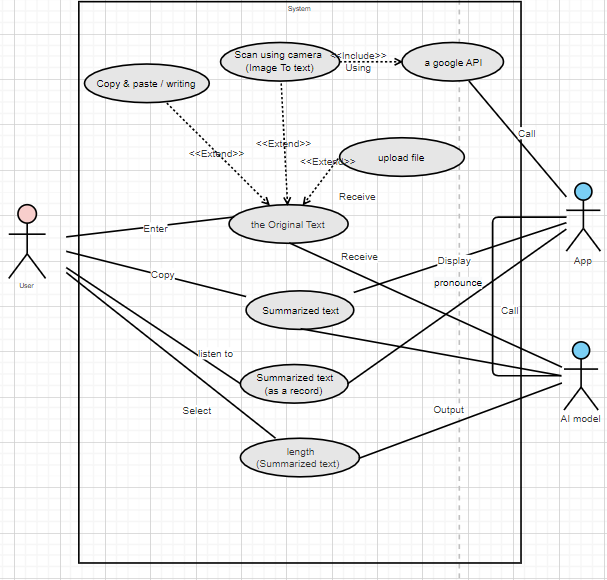


Fig3. 1 Use case Diagram

* + 1. Sequence diagram:

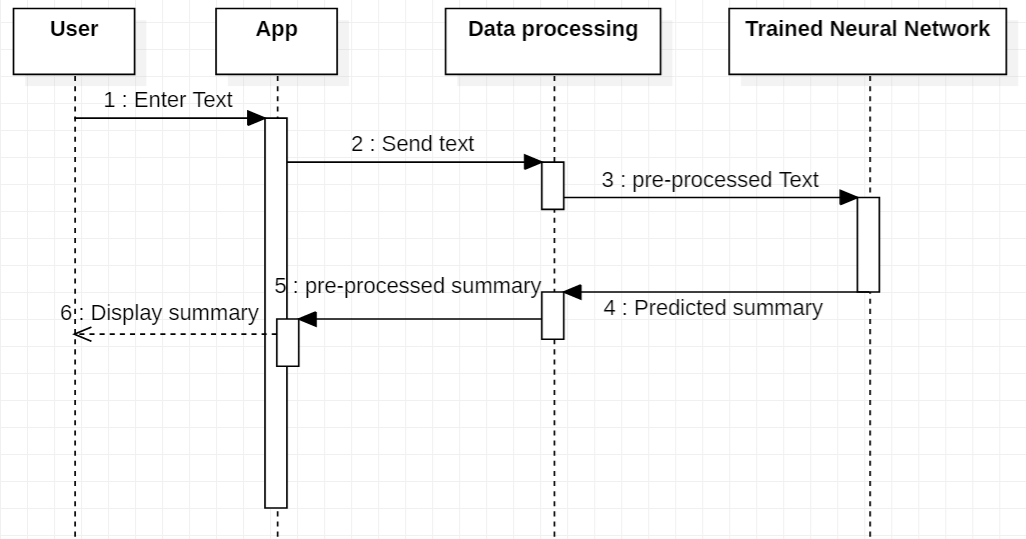
****

Fig3. *3* Sequence Diagram

* + 1. Block Diagram

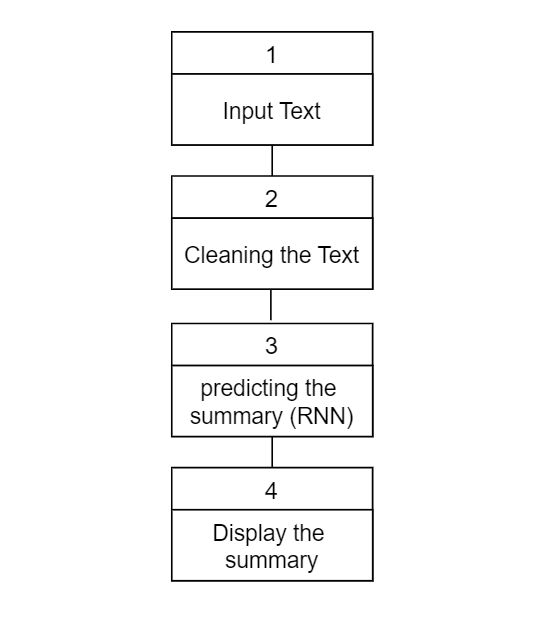


Fig3.4 Block Diagram

* + 1. Flowchart Diagram

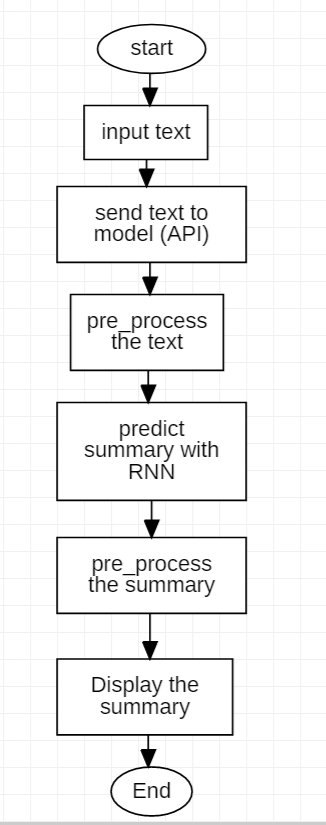


Fig3.5 Flowchart Diagram

* + 1. State Diagram

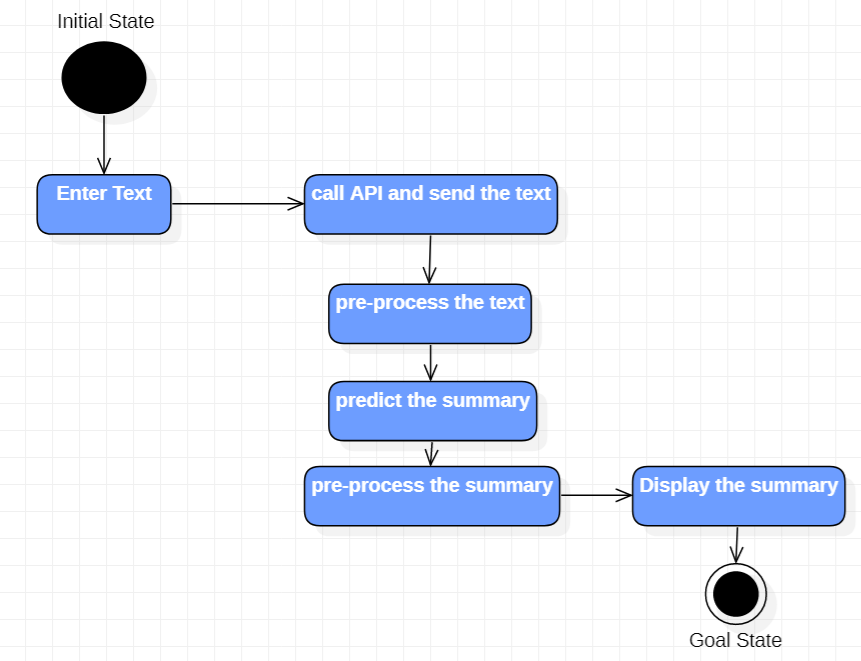


Fig3.5 State Diagram

# **Chapter Four:** Design , Implementation and testing

* 1. Design Technique

Our project will be using a deep learning algorithms , so there are three types of such algorithms :

1. Artificial Neural Networks (ANN)

* What is (ANN)

An artificial neural network (ANN) is a system of hardware and/or software modeled after the activity of neurons in the human brain in information technology (IT). Artificial neural networks, or ANNs, are a type of deep learning technology that falls under the umbrella of artificial intelligence, or AI. (Burke, 2021)



* Advantages of (ANN)

1. The network's parallel processing capabilities imply it can handle several tasks at once.
2. Not simply a database, but a complete network stores information.
3. The capacity to learn and model nonlinear, complicated interactions aids in the modeling of real-life input-output relationships.
4. The input variables are not restricted in any way, including how they should be distributed.
5. Machine learning refers to an artificial neural network's ability to learn from events and make judgments based on those observations.

* Disadvantages of (ANN)

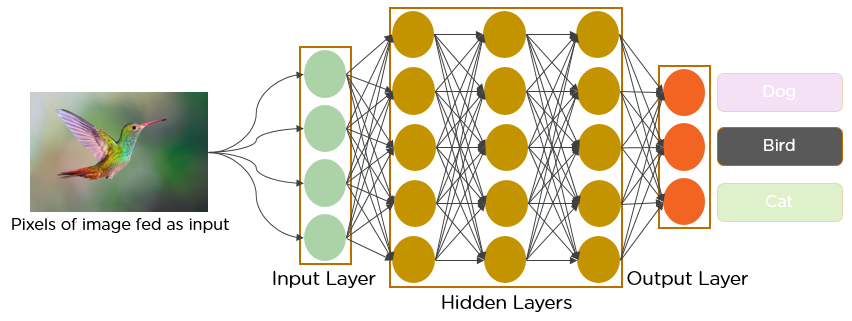
1. Because there are no criteria for selecting the best network topology, the best artificial neural network design can only be discovered by trial and error.
2. Because the network operates on numerical data, all problems must be converted into numerical values before being submitted to the ANN.
3. One of the most serious flaws with ANNs is the lack of justification for probing solutions. The inability to explain why or how the solution was implemented leads to a loss of trust in the network.

(Burke, 2021)

1. Convolutional Neural Networks (CNN)

* What is (CNN)

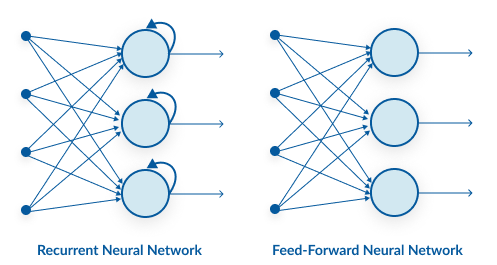
A convolutional neural network, often known as a ConvNet or CNN, is a type of neural network that consists of neurons and learnable parameters such as weights. Each neuron gets a large number of inputs and then processes the weighted sum of the inputs through an activation function to produce an output. (Bhuya, 2021)



They're utilized in the field of computer vision so we cannot use such algorithm in our project

1. Recurrent Neural Networks (RNN)

* What is (RNN)



RNN has a recurrent connection on the concealed state, as seen above. This looping requirement ensures that the supplied data has sequential information.

Recurrent neural networks can be used to handle challenges such as: Data from time series, text data, and audio data.

* Advantages of (RNN)

1. The sequential information in the input data is captured by RNN.
2. The parameters of RNNs are shared between time steps. This is commonly referred to as parameter sharing. As a result, there are fewer parameters to train and the computational cost is lower.

As it is quite clear that this type of neural network is the best suitable type for our project

We will select the (LSTM) Long Short-Term Memory) as our RNN algorithm , now will explain why :

We prioritize our appointments when we plan our day's schedule, right? We know which meeting may be canceled to accommodate a prospective me if we need to make some room for anything crucial.

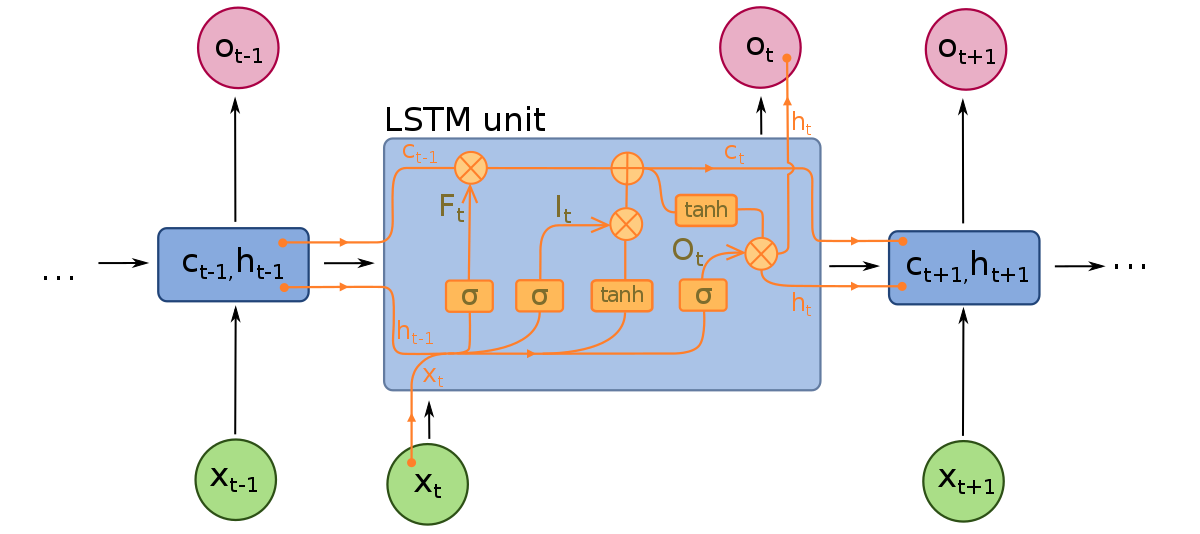
An RNN, it turns out, does not. It applies a function to totally modify the current information in order to add new information. As a result, the entire information gets altered, i.e. there is no distinction between 'important' and 'not so important' information.

LSTMs, on the other hand, use multiplications and adds to make little changes to the data. Information travels through a mechanism known as cell states in LSTMs. LSTMs may selectively recall or forget information in this way. There are three different dependencies on the information at a specific cell state.

These dependencies can be generalized to any problem as:

1. The previous cell state *(i.e. the information that was present in the memory after the previous time step)*
2. The previous hidden state *(i.e. this is the same as the output of the previous cell)*
3. The input at the current time step *(i.e. the new information that is being fed in at that moment)* (Tatman, 2017)

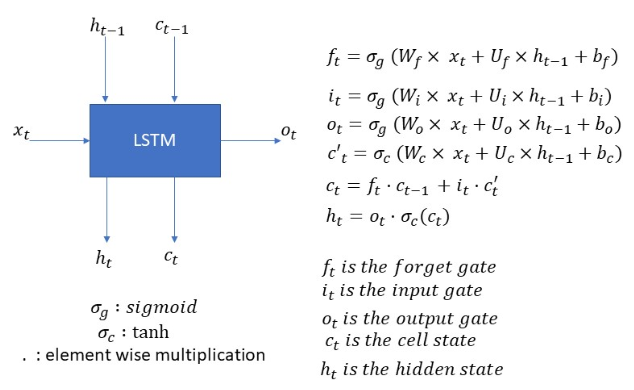
* The architecture of (LSTMs)



A typical LSTM network is made up of several memory blocks known as cells (the rectangles that we see in the image). The cell state and the hidden state are the two states that are passed to the following cell. Memory blocks are in charge of remembering things, and they are manipulated by three basic mechanisms known as gates. (pranj52, 2017)

* A forget gate is in charge of erasing data from the cell state. By multiplying a filter, information that is no longer necessary for the LSTM to comprehend things or that is of lesser value is eliminated. This is essential for the LSTM network's performance to be optimized.
* The input gate is in charge of adding information to the state of the cell. As seen in the picture above, this information addition is a three-step procedure.
* The Output Gate : is in charge of selecting useful information from the current cell state and showing it out as an output .
* The LSTM equations

The inputs and outputs of an LSTM for a single timestep are shown in the diagram below. This is a time unrolled form with one timestep input, output, and equations. The input x(t) of the LSTM might be the output of a CNN or the input sequence itself. The inputs from the previous timestep LSTM are h(t-1) and c(t-1). The output of the LSTM for this timestep is o(t). The LSTM also creates the c(t) and h(t) for the following time step LSTM to consume.



“**SummarizeIt**” is a summarizer app so it is quite clear that the inputs in our project are a sequence of texts and the outputs are also a sequence of texts.

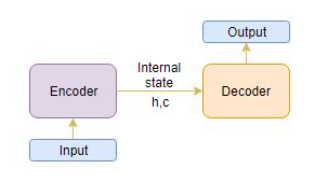
So we can model this as a Many-To-Many Seq2Seq problem.

now we have to build a sequence to sequence model (Seq2Seq)

* Seq2Seq Model

A Seq2Seq model consists of two fundamental components:

1. Encoder
2. Decoder



The Encoder-Decoder architecture is mostly used to address sequence-to-sequence (Seq2Seq) challenges with different length input and output sequences.

Let's look at it from the standpoint of text summary. The input is a large string of words, and the output is a brief summary of the input.

The encoder-decoder can be set up in two stages:

1. Training phase
2. Inference phase

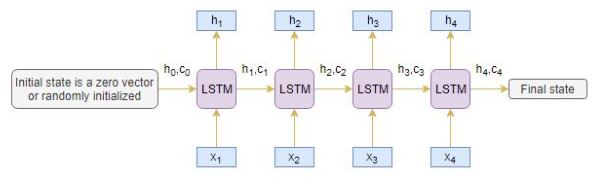
* Training phase

(the encoder)

We will first set up the encoder and decoder during the training phase. The model will then be trained to anticipate the target sequence with one timestep offset. Let's look at how to set up the encoder and decoder in more detail.

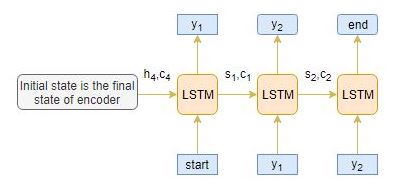
Encoder

The complete input sequence is read by an encoder Long Short Term Memory model (LSTM), with one word being sent into the encoder at each timestep. The information is then processed at each timestep, and the contextual information included in the input sequence is captured.



(the decoder)

The decoder is likewise an LSTM network that analyzes the whole target sequence word-by-word and predicts a sequence that is one timestep delayed. Given the previous word, the decoder is trained to anticipate the next word in the sequence.



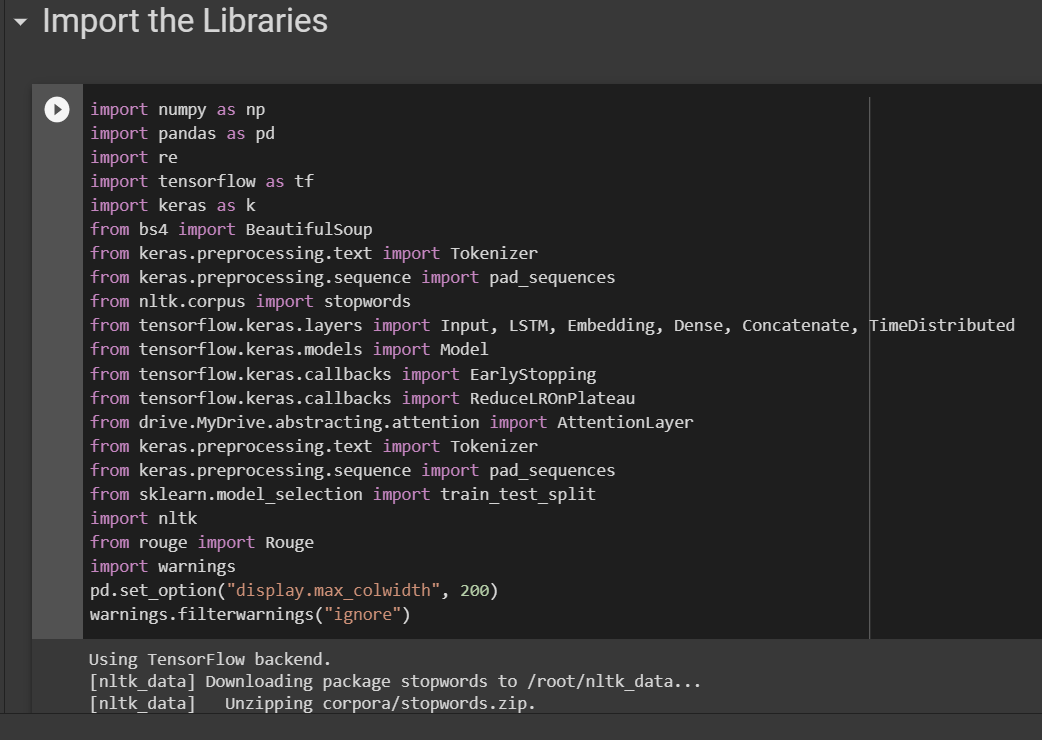
* **The Dataset**

The dataset that we used to train our model is “ Amazon Fine Food Reviews “

This dataset contains Amazon reviews on exquisite meals. The data covers more than a decade and includes all 500,000 reviews up until October 2012. Product and user information, reviews summary , ratings, and a plain text review are all included in reviews. It also contains reviews from all of Amazon's other categories. (Kaggle, 2018)

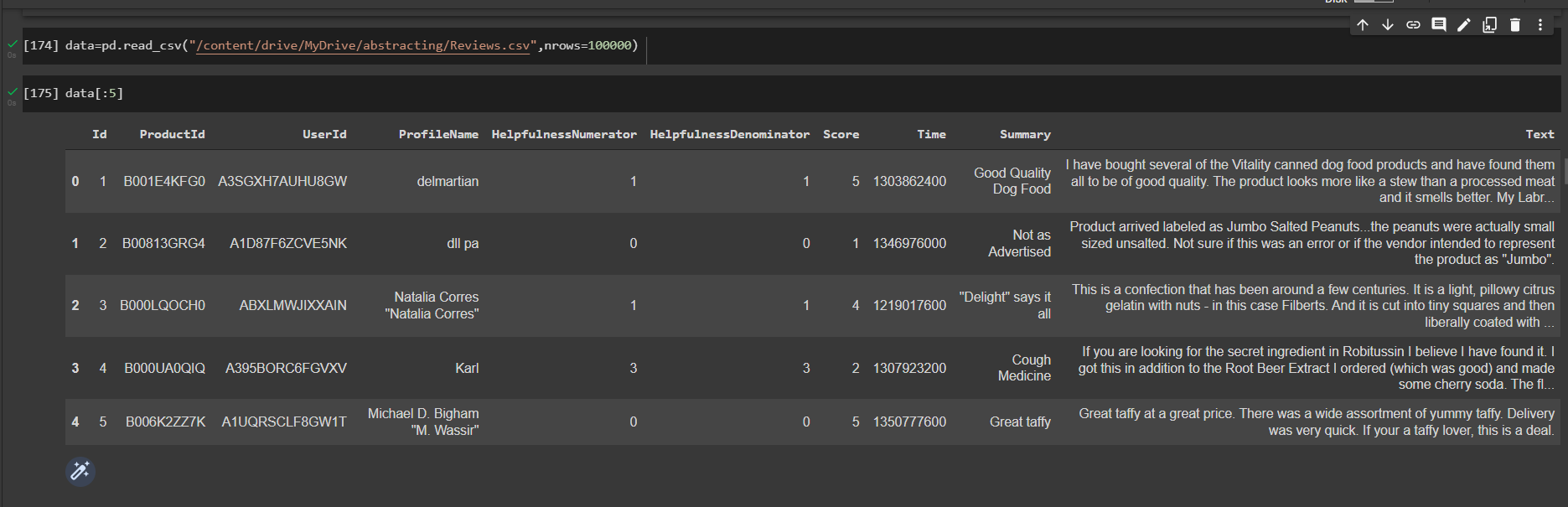
* 1. Implementation
     1. **Project Structure Pipeline:**

1. Importing libraries and frameworks
2. Getting Dataset from “kaggel”
3. Data Preprocessing
4. Build the Model
5. Train the model with supervised learning
6. Saving the model weights
7. Loading the model to try it
8. Create Rest API to call it from the flutter app
9. Build the “SummarizeIt” flutter app
10. Test “SummarizeIt”
    * 1. **Importing Libraries and frameworks:**

****

* + 1. **Getting the Data :**

We’ll take a sample of 100,000 reviews to reduce the training time of our model.



* + 1. **Data Preprocessing:**

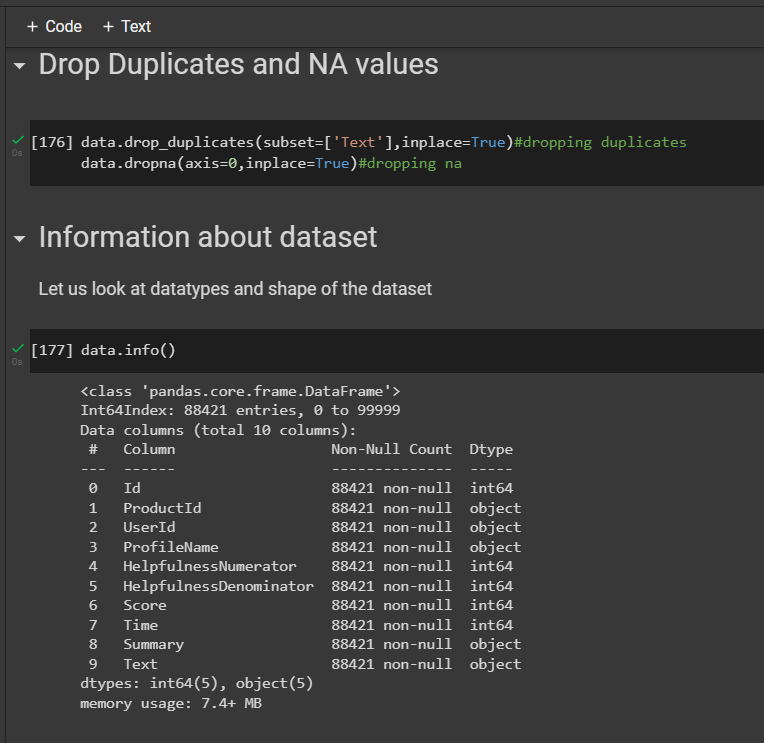
Before we start developing the model, we must first complete some fundamental preprocessing tasks. Using filthy and sloppy text data may be harmful. As a result, in this stage, we will remove any unneeded symbols, letters, and other elements from the text that do not influence the problem's goal.

* The contraction mapping:

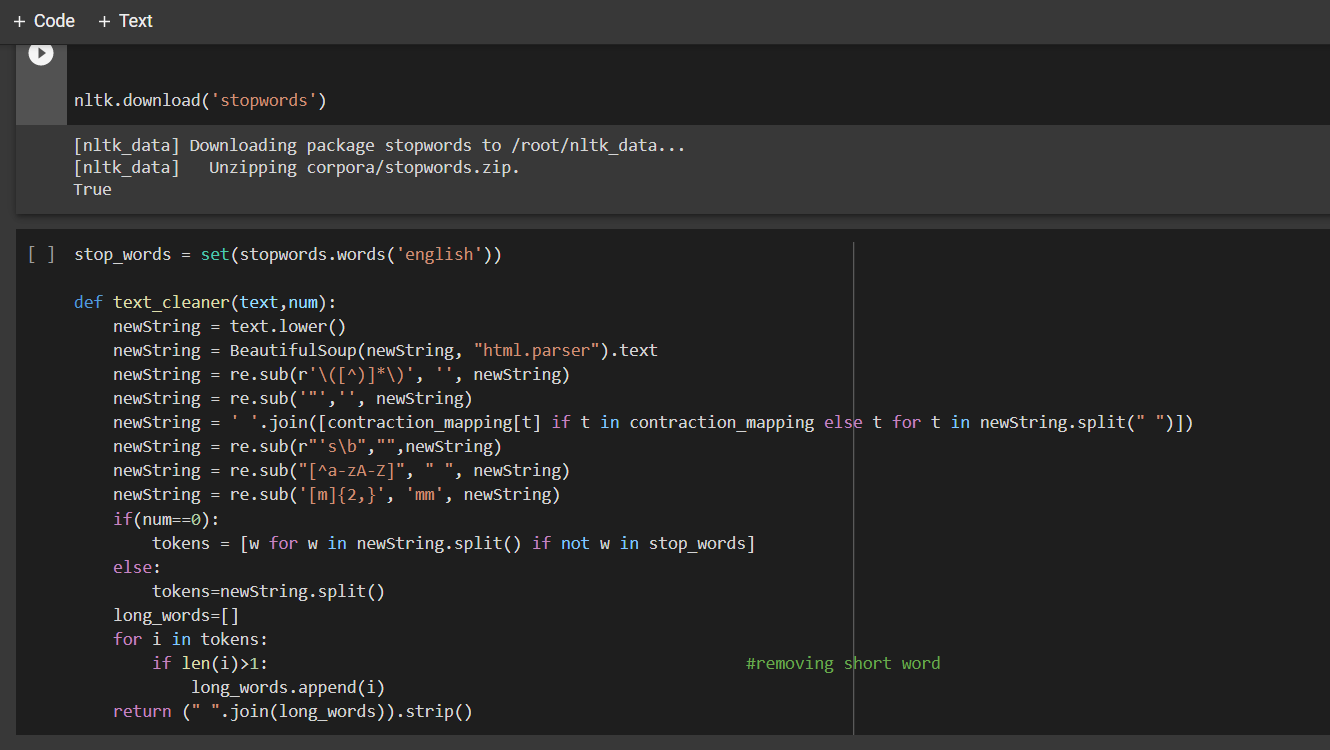


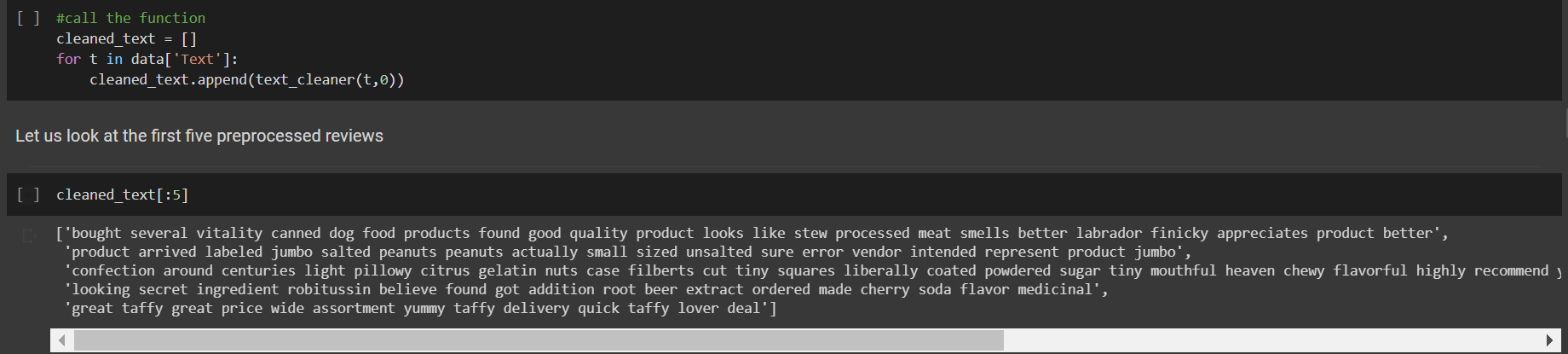
For our data, we will conduct the following preprocessing tasks:

* Drop Null An Duplicates values

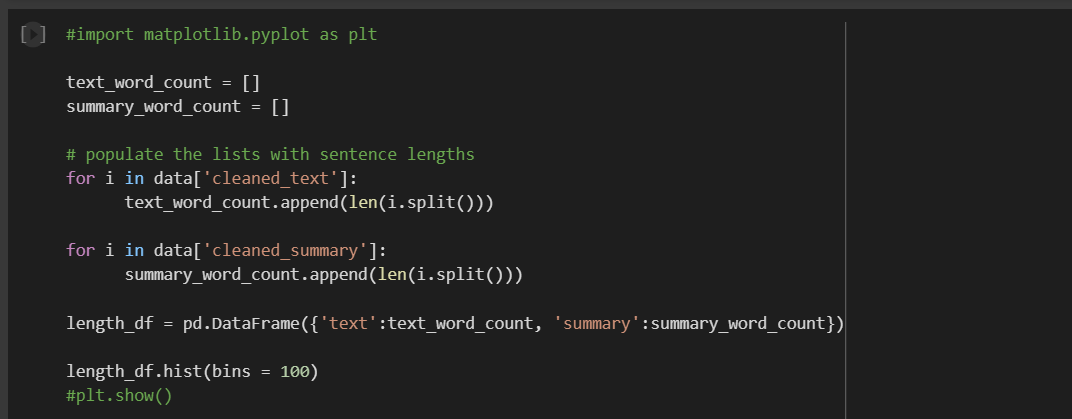


* Convert all the data to lowercase
* Remove HTML tags
* Remove(‘s)
* Remove stopwords
* Remove short words
* Contraction mapping
* Remove any text inside the parenthesis ()
* Eliminate special characters and punctuations

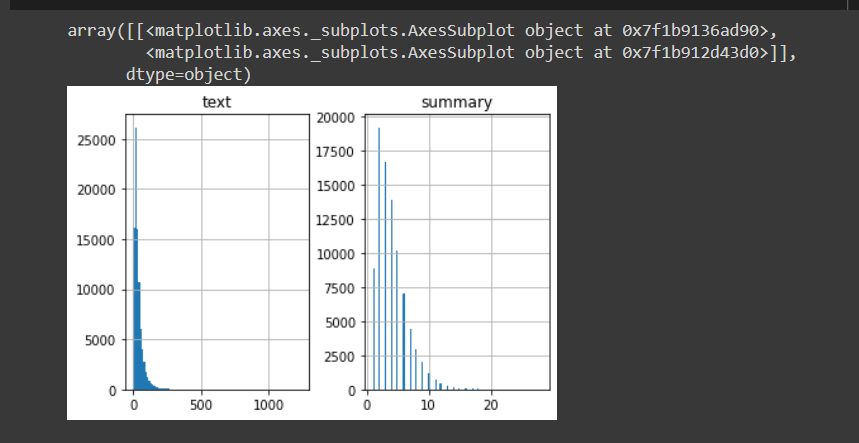




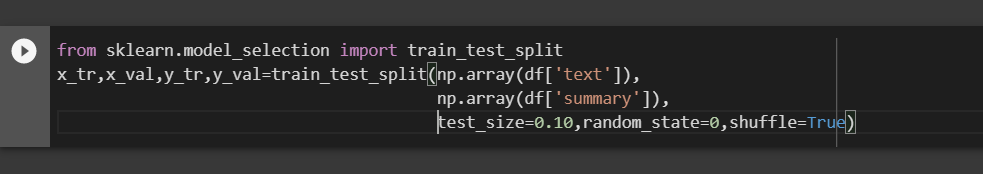
We'll look at the length of the reviews and the summary to obtain a sense of the text's overall length distribution. This will assist us in determining the sequence's maximum length:



Output:

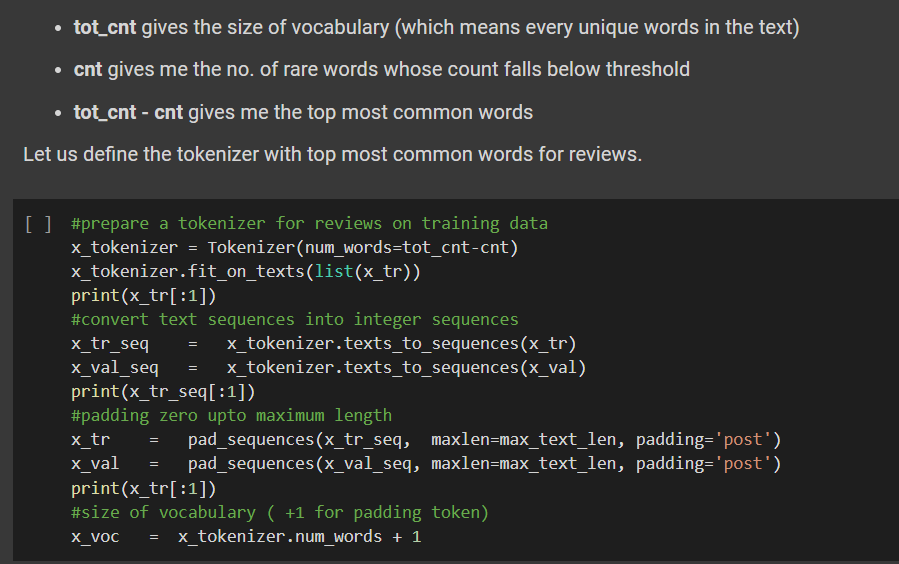


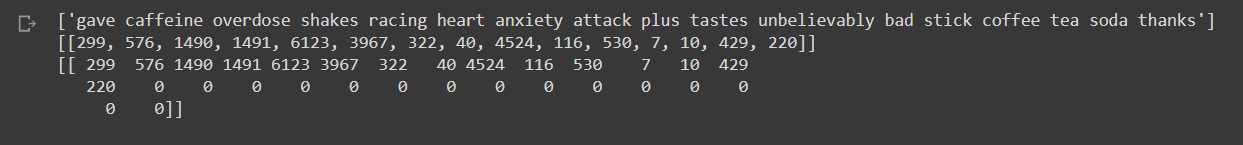
We must divide our data into two sets: training and validation. We'll use 90% of the dataset as training data and measure performance against the remaining 10% (holdout set):



* Text tokenizer

A tokenizer creates a vocabulary and turns a string of words into an integer sequence. Build the following tokenizers for text and summary:





* + 1. **Model Building :**

Finally, we've arrived at the modeling stage. But before we begin, we must first become acquainted with a few terminology that are necessary for the model's construction.

True if Return Sequences is true: LSTM outputs the hidden state and cell state for each timestep when the return sequences parameter is set to True.

True if the state is returned: When return state = True, the last timestep's hidden state and cell state are produced by LSTM. only

For the first timestep, this is used to initialize the internal states of the LSTM.

Multiple layers of LSTM are piled on top of each other in stacked LSTM. As a result, the sequence is better represented. I recommend experimenting with the LSTM's various layers piled on top of each other (it's a terrific way to learn).

For the encoder, we're making a three-stacked LSTM:

from keras import backend as K

K.clear\_session()

latent\_dim = 300

embedding\_dim=200

# Encoder

encoder\_inputs = Input(shape=(max\_text\_len,))

#embedding layer

enc\_emb =  Embedding(x\_voc, embedding\_dim,trainable=True, name="encoder\_embedding")(encoder\_inputs)

#encoder lstm 1

encoder\_lstm1 = LSTM(latent\_dim,return\_sequences=True,return\_state=True,dropout=0.4,recurrent\_dropout=0.4, name="en\_lstm\_1")

encoder\_output1, state\_h1, state\_c1 = encoder\_lstm1(enc\_emb)

#encoder lstm 2

encoder\_lstm2 = LSTM(latent\_dim,return\_sequences=True,return\_state=True,dropout=0.4,recurrent\_dropout=0.4, name="en\_lstm\_2")

encoder\_output2, state\_h2, state\_c2 = encoder\_lstm2(encoder\_output1)

#encoder lstm 3

encoder\_lstm4=LSTM(latent\_dim, return\_state=True, return\_sequences=True,dropout=0.4,recurrent\_dropout=0.4, name="en\_lstm\_3")

encoder\_outputs, state\_h, state\_c= encoder\_lstm4(encoder\_output2)

# Set up the decoder, using `encoder\_states` as initial state.

decoder\_inputs = Input(shape=(None,), name="de\_inputs")

#embedding layer

dec\_emb\_layer = Embedding(y\_voc, embedding\_dim,trainable=True, name="de\_embedding")

dec\_emb = dec\_emb\_layer(decoder\_inputs)

decoder\_lstm = LSTM(latent\_dim, return\_sequences=True, return\_state=True,dropout=0.4,recurrent\_dropout=0.2, name="de\_lstm")

decoder\_outputs,decoder\_fwd\_state, decoder\_back\_state = decoder\_lstm(dec\_emb,initial\_state=[state\_h, state\_c])

# Attention layer

attn\_layer = AttentionLayer(name='attention\_layer')

attn\_out, attn\_states = attn\_layer([encoder\_outputs, decoder\_outputs])

print(attn\_out.dtype)

# Concat attention input and decoder LSTM output

decoder\_concat\_input = Concatenate(axis=-1, name='concat\_layer')([decoder\_outputs, attn\_out])

#dense layer

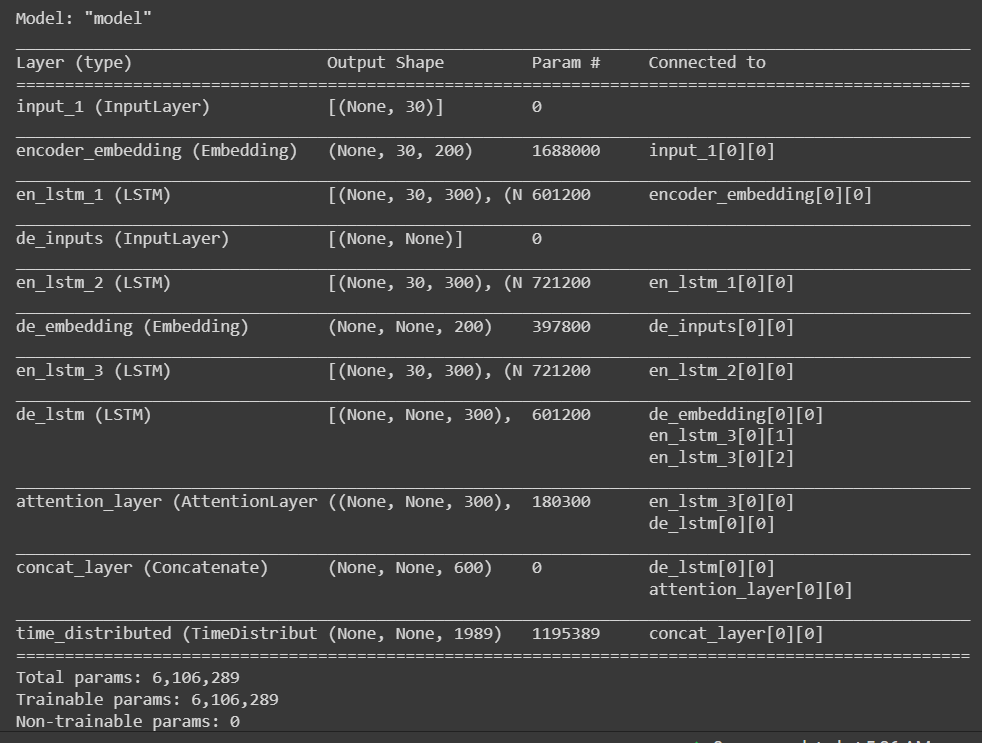
decoder\_dense =  TimeDistributed(Dense(y\_voc, activation='softmax', name="dense\_layer"))

decoder\_outputs = decoder\_dense(decoder\_concat\_input)

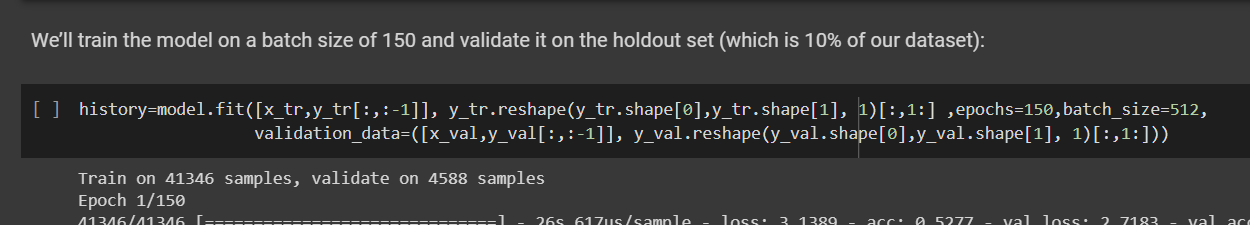
# Define the model

model = Model([encoder\_inputs, decoder\_inputs], decoder\_outputs)

* Model summary:



* + 1. **Model Training :**
* Epochs : 150
* Batch size : 512
* Validation set : 10%

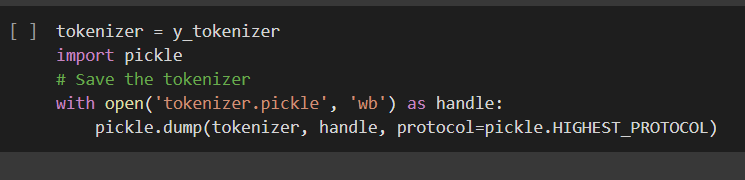
****

* + 1. **Model weights Saving :**
* Because of the structure of the model , we have

Encoder layers , decoder layers , attention layer and dense layer

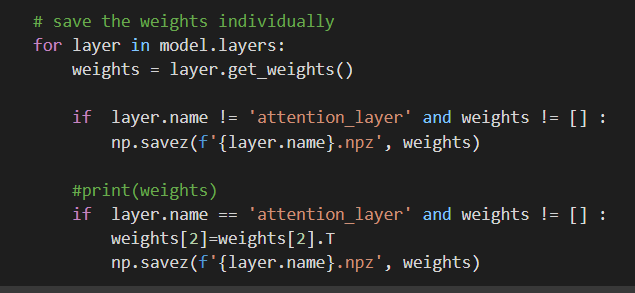
So we need to save each layer individually .

* Saving the tokenizer :

****

* Saving rest of layers:

We will save the weights in a files .**npz**



* + 1. **Loading the model to test it :**
* First we need to build the encoder\_decoder **Inference :**
* # Encode the input sequence to get the feature vector
* encoder\_model = Model(inputs=encoder\_inputs,outputs=[encoder\_outputs, state\_h, state\_c])
* # Decoder setup
* # Below tensors will hold the states of the previous time step
* decoder\_state\_input\_h = Input(shape=(latent\_dim,))
* decoder\_state\_input\_c = Input(shape=(latent\_dim,))
* decoder\_hidden\_state\_input = Input(shape=(max\_text\_len,latent\_dim))
* # Get the embeddings of the decoder sequence
* dec\_emb2= dec\_emb\_layer(decoder\_inputs)
* # To predict the next word in the sequence, set the initial states to the states from the previous time step
* decoder\_outputs2, state\_h2, state\_c2 = decoder\_lstm(dec\_emb2, initial\_state=[decoder\_state\_input\_h, decoder\_state\_input\_c])
* #attention inference
* attn\_out\_inf, attn\_states\_inf = attn\_layer([decoder\_hidden\_state\_input, decoder\_outputs2])
* decoder\_inf\_concat = Concatenate(axis=-1, name='concat')([decoder\_outputs2, attn\_out\_inf])
* # A dense softmax layer to generate prob dist. over the target vocabulary
* decoder\_outputs2 = decoder\_dense(decoder\_inf\_concat)
* # Final decoder model
* decoder\_model = Model(
* [decoder\_inputs] + [decoder\_hidden\_state\_input,decoder\_state\_input\_h, decoder\_state\_input\_c],
* [decoder\_outputs2] + [state\_h2, state\_c2])

Now we need to define a function for the Inference process:

def decode\_sequence(input\_seq):

    # Encode the input as state vectors.

    e\_out, e\_h, e\_c = encoder\_model.predict(input\_seq)

    # Generate empty target sequence of length 1.

    target\_seq = np.zeros((1,1))

    # Populate the first word of target sequence with the start word.

    target\_seq[0, 0] = target\_word\_index['sostok']

    stop\_condition = False

    decoded\_sentence = ''

    while not stop\_condition:

        output\_tokens, h, c = decoder\_model.predict([target\_seq] + [e\_out, e\_h, e\_c])

        # Sample a token

        sampled\_token\_index = np.argmax(output\_tokens[0, -1, :])

        sampled\_token = reverse\_target\_word\_index[sampled\_token\_index]

        if(sampled\_token!='eostok'):

            decoded\_sentence += ' '+sampled\_token

        # Exit condition: either hit max length or find stop word.

        if (sampled\_token == 'eostok'  or len(decoded\_sentence.split()) >= (max\_summary\_len-1)):

            stop\_condition = True

        # Update the target sequence (of length 1).

        target\_seq = np.zeros((1,1))

        target\_seq[0, 0] = sampled\_token\_index

        # Update internal states

        e\_h, e\_c = h, c

    return decoded\_sentence

now we can load the model:

* Getting the .npz files from the memory :

import pickle

tokenizer = Tokenizer()

# load the tokenizer

with open('tokenizer.pickle', 'rb') as handle:

    tokenizer = pickle.load(handle)

# load the weights

w\_encoder\_embeddings = np.load('encoder\_embedding.npz', allow\_pickle=True)

w\_decoder\_embeddings = np.load('de\_embedding.npz', allow\_pickle=True)

w\_encoder\_lstm\_1 = np.load('en\_lstm\_1.npz', allow\_pickle=True)

w\_encoder\_lstm\_2 = np.load('en\_lstm\_2.npz', allow\_pickle=True)

w\_encoder\_lstm\_3 = np.load('en\_lstm\_3.npz', allow\_pickle=True)

w\_decoder\_lstm = np.load('de\_lstm.npz', allow\_pickle=True)

w\_dense = np.load('time\_distributed.npz', allow\_pickle=True)

w\_attention\_layer = np.load('new\_attention\_layer.npz', allow\_pickle=True)

* Load the weights to the model layers

 # set the weights of the model

model.layers[1].set\_weights(w\_encoder\_embeddings['arr\_0'])

model.layers[5].set\_weights(w\_decoder\_embeddings['arr\_0'])

model.layers[2].set\_weights(w\_encoder\_lstm\_1['arr\_0'])

model.layers[4].set\_weights(w\_encoder\_lstm\_2['arr\_0'])

model.layers[6].set\_weights(w\_encoder\_lstm\_3['arr\_0'])

model.layers[7].set\_weights(w\_decoder\_lstm['arr\_0'])

model.layers[8].set\_weights(w\_attention\_layer['arr\_0'])

model.layers[10].set\_weights(w\_dense['arr\_0'])

* + 1. **Creating REST API :**
* We used Flask Back-end Technology to set up the REST API ,

The reason of using such technology is that our model is created in python and Flask is based on python so it will easier to implement it.

* The flask app will contain a function that uses the same implementation of the Inference and loading weights above to predict the summary of the new text .
* The API receives the new input text from the flutter app

And return the summary of that text .

* The flask app will be running locally , and we can access it on the default port: 5000
* Example of the API :

http://127.0.0.1:5000/api?query=I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than  most.

* + 1. **Creating the Flutter APP :**

**“SummarizeIt”** will be just tow screens

* First screen contain a field to enter the text , and a button to summarize the text .

// ignore\_for\_file: prefer\_const\_constructors, avoid\_print

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'summarized\_text.dart';

import 'package:flutter\_easyloading/flutter\_easyloading.dart';

import 'package:http/http.dart' as http;

class TextPage extends StatefulWidget {

  const TextPage({Key? key}) : super(key: key);

  @override

  State<TextPage> createState() => \_TextPageState();

}

class \_TextPageState extends State<TextPage> {

  var mycontroller = TextEditingController();

  String? mytext;

  String? mysummary;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Your Text'),

        centerTitle: true,

      ),

      body: Padding(

        padding: const EdgeInsets.all(5.0),

        child: Container(

          color: Colors.white,

          child: Column(

            mainAxisSize: MainAxisSize.max,

            children: [

              Expanded(

                child: TextField(

                  controller: mycontroller,

                  expands: true,

                  maxLines: null,

                  showCursor: true,

                  style:TextStyle(

                    fontSize: 23,

                    //fontWeight: FontWeight.bold,

                  ) ,

                ),

              ),

              Container(

                width: double.infinity,

                color: Color.fromARGB(255, 202, 230, 255),

                child: TextButton(

                    onPressed: () async {

                      mytext = mycontroller.text;

                      EasyLoading.show(status: 'Summarizing...');

                      var url =

                          Uri.http('10.0.2.2:5000', '/api', {'query': mytext});

                      var response = await http.get(url);

                      if (response.statusCode == 200) {

                        String resText =response.body;

                        EasyLoading.dismiss();

                        Navigator.push(

                          context,

                          MaterialPageRoute(

                              builder: (context) =>  SummarizedText(sumedText : resText)),

                        );

                      } else {

                        print(

                            'Request failed with status: ${response.statusCode}.');

                      }

                    },

                    child: Text(

                      'Summarize',

                      style: TextStyle(

                        fontSize: 25,

                        fontWeight: FontWeight.bold,

                      ),

                    )),

              )

            ],

          ),

        ),

      ),

    );

  }

}

* The second screen contain a filed to display the summary of the text :

// ignore\_for\_file: prefer\_const\_constructors

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

class SummarizedText extends StatelessWidget {

  const SummarizedText({Key? key , required this.sumedText}) : super(key: key);

  final String sumedText;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        centerTitle: true,

        title: Text('Summary'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(5.0),

        child: Container(

          color: Colors.white,

          child: Text(

            sumedText,

            style:TextStyle(

                      fontSize: 23,

                      fontWeight: FontWeight.bold,

                    )

          ),

        ),

      ),

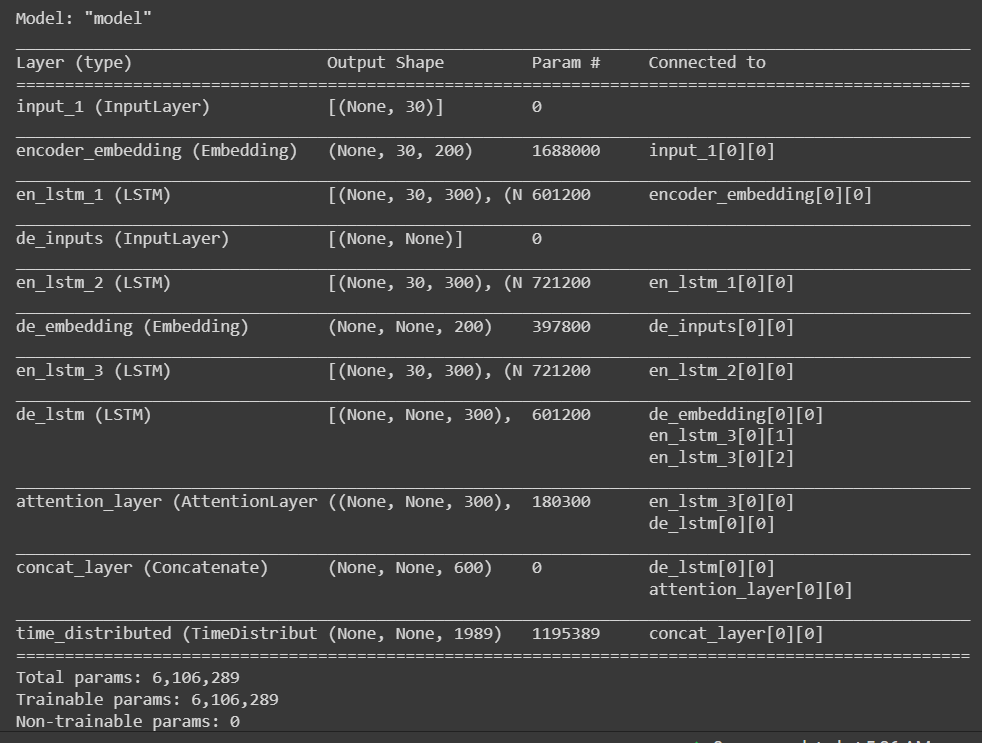
    );

  }

}

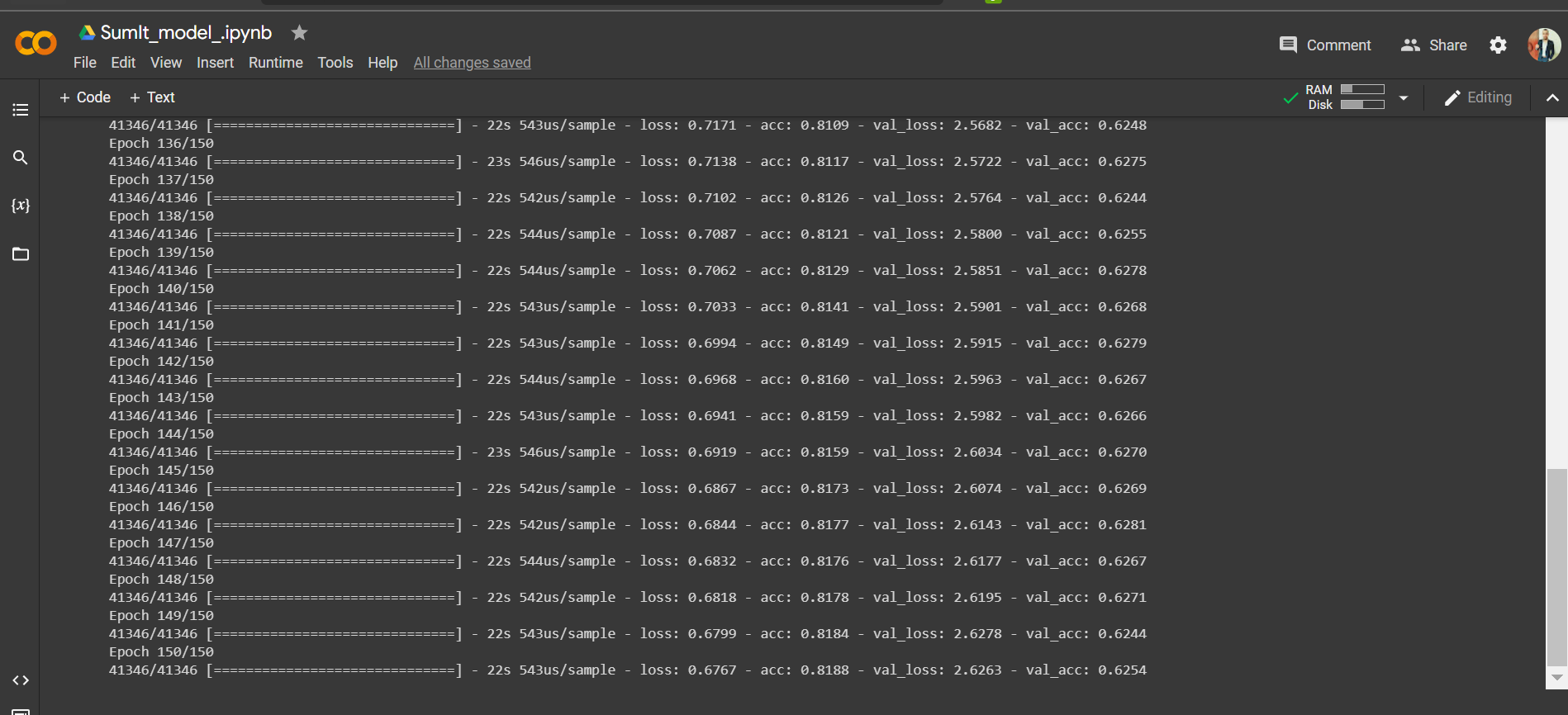
**Chapter Five:** Results and discussion**.**

* 1. Results of Machine learning model
     1. Model summary:



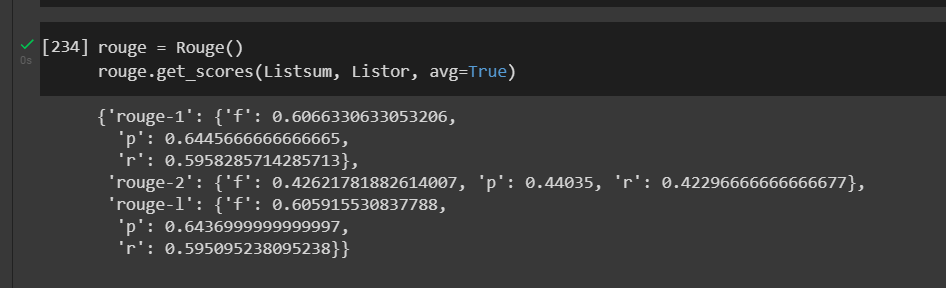
* + 1. Training Result:

(last 15 epochs)



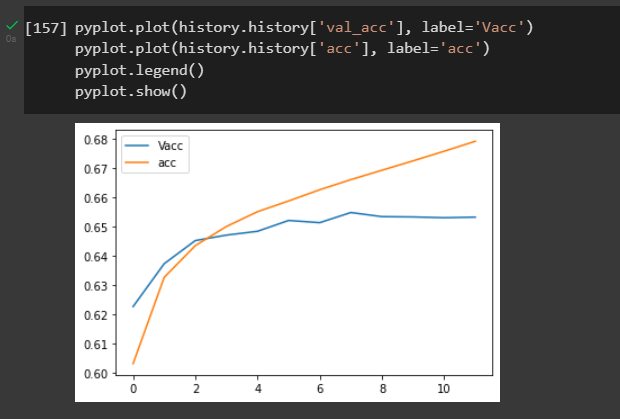
* + 1. Accuracy of the model :
* It’s important to know that our model work in abstracting approach so Measurement equation will not be the accuracy of the training ,

So we implemented the ROUG test.

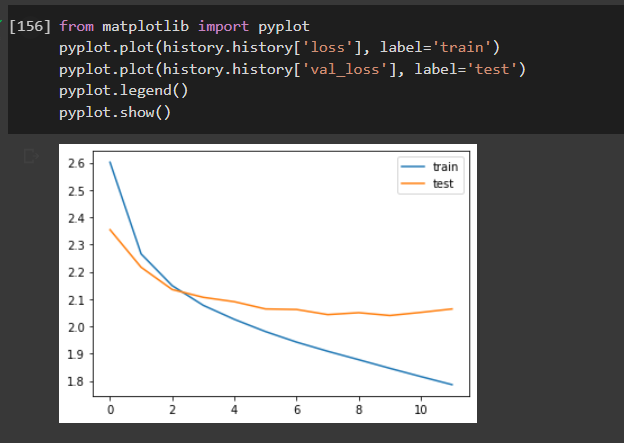


* + 1. Analysis after Training :

The accuracy analysis plot :



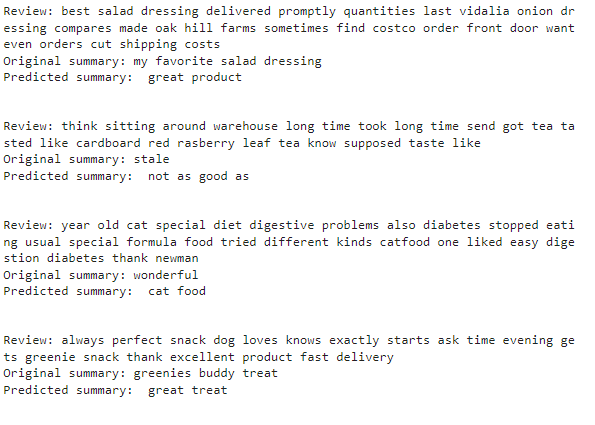
The loss analysis plot :



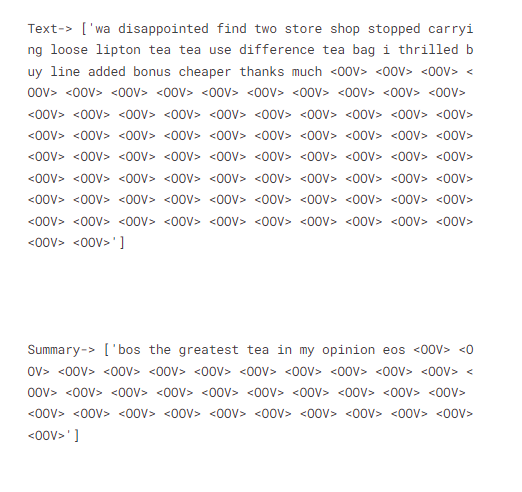
* 1. Comparing results with model with the same dataset :

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Acc | Rouge test | Test result |
| (aravindpai, 2019) | 65% | Rouge-1 : 0.20  Rouge-2 : 0.14  Rouge-L : 0.25 | First next screen below |
| (Narayankar, 2021) | 28% | Not implemented | second next screen below |
| Our Model | 81% | Rouge-1 : 0.60  Rouge-2 : 0.40  Rouge-L : 0.64 | third next screen below |

* First screen :



* Second screen :



* Third screen :



* 1. Results of Flutter App :

|  |  |
| --- | --- |
| The text | The Summary |
|  |  |
|  |  |

* 1. Goal achieved :

working with the text in the machine learning not easy because the data need a lot of preprocessing before any building of the model ,

i succeeded to preprocess the dataset that i used to train the model in a good way ( cleaning , analyzing , tokenyzing ,etc)

I succeeded in building an AI model that using deep learning with the RNN (LSTM) algorithm .

it has an accuracy of 81.8% ,

Rouge-1 = 60 , Rouge-2 = 40 , Rouge-L = 64 scores

Our model considered as stat-of-the art model that trained on this dataset which is “Amazon fine food reviews” after training the model can predict a good summary for the review that looks like the human summary .

Implementing the encoder-decoder was not easy but I succeeded on it ,

“SummarizIt” works by applying cross-platform with flutter technology , which allows one code to work on more than one operating system .

“SummarizIt” dos not need any privet information about the user

**Chapter Six:** **Future work and the Conclusion**

* 1. Future work :

1. I will do the best to improve the performance of the model with the next ways:

* I will train the model on a larger dataset .
* I will implement the BILSTM algorithm (which I already working on currently ) .
* I will implement the Beam search algorithm to solve the problem of repeating the predicted words .

1. I will improve the functionality of the model with :

* Train the model on a deferent datasets to be a general summarizer app

1. I will improve the functionality of the flutter app

* Implementing the feature of scanning text from images
* Implementing the feature of upload a file from the memory
* Implementing the feature of text to speech

1. Making a free version and a paid version for companies with more features that are specially developed according to the needs of each institution.
   1. Conclusion :

at the end "Summarize It" is the app which i created as the graduation project , it is an abstracting summarizer , that build in a sequence to sequence structure encoder decoder , build on a RNN (lstm algorithm) .

the way to use the app is just you need to open the app input the text you wont to summarize press the puttom and the summary will displayed on the screen .

I had build this project alone , with my own efforts , a spend alot of time to study courses of AI , ML , DL and flutter mobile application , it was a hard proccess , too long , very hard , i faced a lot and a lot of problems with the understanding the proccess of development of the ML for the first time to learn , errors in evry line of the code

, so i needed a lot of help and i asked evry one i knwed that he can help me .

at the end of all that , i learned alot of things , a huge amount of expereance ,

i'm very excited to the future to see how the project will help people.

# References

aravindpai, 2019. *AnalyticsVidhya.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/  
[Accessed 5 5 2022].

Bhuya, S., 2021. *opengenusIQ.* [Online]   
Available at: https://iq.opengenus.org/disadvantages-of-cnn/  
[Accessed 6 5 2022].

Burke, E. B. /. J., 2021. *SearchEnterpriseAI.* [Online]   
Available at: https://www.techtarget.com/searchenterpriseai/definition/neural-network  
[Accessed 6 5 2022].

devopedia, 2020. *Text Summarization.* [Online]   
Available at: https://devopedia.org/text-summarization  
[Accessed 5 6 2022].

Gambhir, M., 2016. Recent automatic text summarization techniques:. *Recent automatic text summarization techniques*, 29 3, p. 66.

Halifax, K. K., 2017. A Brief Survey of Text Mining: Classification, Clustering and. *Classification, Clustering and*, 28 9, p. 13.

Kaggle, 2018. *Kaggle.* [Online]   
Available at: https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews  
[Accessed 5 5 2022].

Narayankar, S., 2021. *NLP Summarization.* [Online]   
Available at: https://www.kaggle.com/code/striker7/nlp-summarization  
[Accessed 5 5 2022].

pranj52, 2017. *AnalyticsVidhya.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/?utm\_source=blog&utm\_medium=comprehensive-guide-text-summarization-using-deep-learning-python  
[Accessed 5 5 2022].

Shen, D., 2009. *Springer link.* [Online]   
Available at: https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9\_424  
[Accessed 5 6 2022].

Tatman, R., 2017. *Analytics Vidhy.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/?utm\_source=blog&utm\_medium=comprehensive-guide-text-summarization-using-deep-learning-python  
[Accessed 5 5 2022].

Vashisht, A., 2019. *OpenGenus IQ \_Text Summarization using RNN.* [Online]   
Available at: https://iq.opengenus.org/text-summarization-using-rnn/  
[Accessed 5 6 2022].