

**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 Four:** Design , Implementation and testing. 7](#_Toc103024045)

[**Design Technique** 7](#_Toc103024046)

[Artificial Neural Networks (ANN) 8](#_Toc103024047)

[**Convolutional Neural Networks (CNN)** 7](#_Toc103024048)

[**Recurrent Neural Networks (RNN)** 8](#_Toc103024049)

[**Long Short-Term Memory (LSTM)** 9](#_Toc103024050)

[**Encoder Decoder (Seq2Seq)** 12](#_Toc103024051)

[**The Dataset** 15](#_Toc103024052)

[**Software / Frameworks Requirements** 15](#_Toc103024053)

[**Implementation** 15](#_Toc103024054)

[**Project structure pipeline:** 16](#_Toc103024055)

[**Importing Libraries and frameworks:** 16](#_Toc103024056)

[**Getting the Data:** 17](#_Toc103024057)

[**Data Preprocessing :** 17](#_Toc103024058)

[**Model Building:** 22](#_Toc103024059)

[**Model summary :** 24](#_Toc103024060)

[**Model Training:** 25](#_Toc103024061)

[**Model Weights Saving:** 25](#_Toc103024062)

[**Loading the model to test it :** 26](#_Toc103024063)

[**Creating REST API:** 29](#_Toc103024064)

[**Creating Flutter App:** 30](#_Toc103024065)

[**Chapter Five: Results and discussion .** 34](#_Toc103024066)

[**Results of Machine learning model** 35](#_Toc103024067)

[**Model Summary:** 35](#_Toc103024068)

[**Training Result :** 36](#_Toc103024069)

[**Accuracy of the model :** 37](#_Toc103024070)

[**Analysis after Training :** 38](#_Toc103024071)

[**Comparing results with model with the same dataset :** 39](#_Toc103024072)

[**Results of Flutter App:** 41](#_Toc103024073)

[**Goal Achieved:** 44](#_Toc103024074)

[**Chapter Six : Future work and the Conclusion** 45](#_Toc103024075)

[**Future Work:** 46](#_Toc103024076)

[**Conclusion :** 47](#_Toc103024077)

[**References** (Harvard Style) 48](#_Toc103024078)

[References 48](#_Toc103024079)

# Table of Figures

[Figure 1 : Nural Network 9](#_Toc103025639)

[Figure 2 CNN 11](#_Toc103025640)

[Figure 3 RNN 12](#_Toc103025641)

[Figure 4 LSTM 14](#_Toc103025642)

[Figure 5 : LSTM equations 15](#_Toc103025643)

[Figure 6 : Seq2Seq 16](#_Toc103025644)

[Figure 7 / Encoder 17](#_Toc103025645)

[Figure 8 / Decoder 18](#_Toc103025646)

[Figure 9 / Import the Libraries 20](#_Toc103025647)

[Figure 10 / Getting data 20](#_Toc103025648)

[Figure 11 / Contraction mapping 21](#_Toc103025649)

[Figure 12 / Drop Duplicates 22](#_Toc103025650)

[Figure 13 / Data cleaning 23](#_Toc103025651)

[Figure 14 / Data Analysis 24](#_Toc103025652)

[Figure 15 / Dividing the data 25](#_Toc103025653)

[Figure 16 / Tokenizer 25](#_Toc103025654)

[Figure 17 / code for model building 28](#_Toc103025655)

[Figure 18 / model summary 29](#_Toc103025656)

[Figure 19 / model training 30](#_Toc103025657)

[Figure 20 / saving the tokenizer 30](#_Toc103025658)

[Figure 21 / saving the model layers 31](#_Toc103025659)

[Figure 22 / encoder\_decoder Inference 32](#_Toc103025660)

[Figure 23 / Getting the NPZ files 34](#_Toc103025661)

[Figure 24 / Loading the weights to the model 34](#_Toc103025662)

[Figure 25 / Example of the API 35](#_Toc103025663)

[Figure 26 / flutter code 37](#_Toc103025664)

[Figure 27 / Training result 41](#_Toc103025665)

[Figure 28 / Rouge test score 42](#_Toc103025666)

[Figure 29 / accuracy analysis plot 43](#_Toc103025667)

[Figure 30 / loss plot 44](#_Toc103025668)

[Figure 31 / Comparing results with others 44](#_Toc103025669)

[Figure 32 / test results 46](#_Toc103025670)

# Table of Tables

[Table 1 / Comparing results with others 44](#_Toc103025737)

[Table 2 / flutter results 47](#_Toc103025738)

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

## **Design Technique**

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

### 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)



Figure 1 : Nural Network

* 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)

# **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)

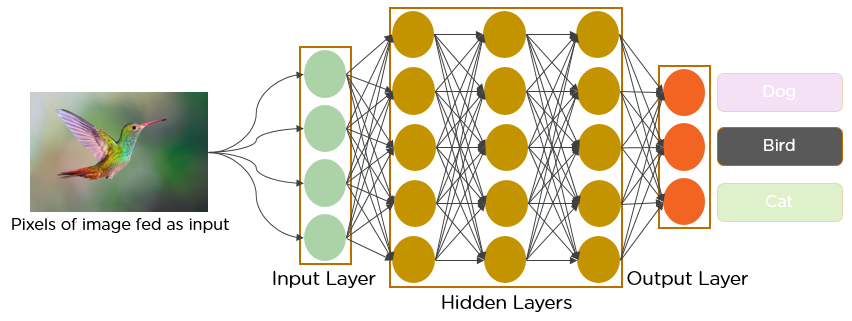
4

Figure 2 CNN

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

# **Recurrent Neural Networks (RNN)**

* What is (RNN)

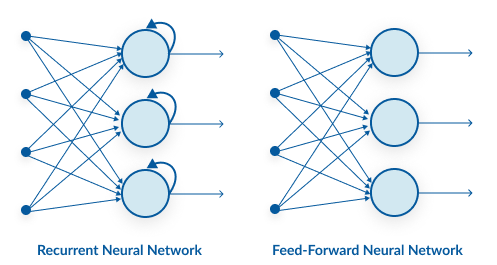


Figure 3 RNN

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

# **Long Short-Term Memory (LSTM)**

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)

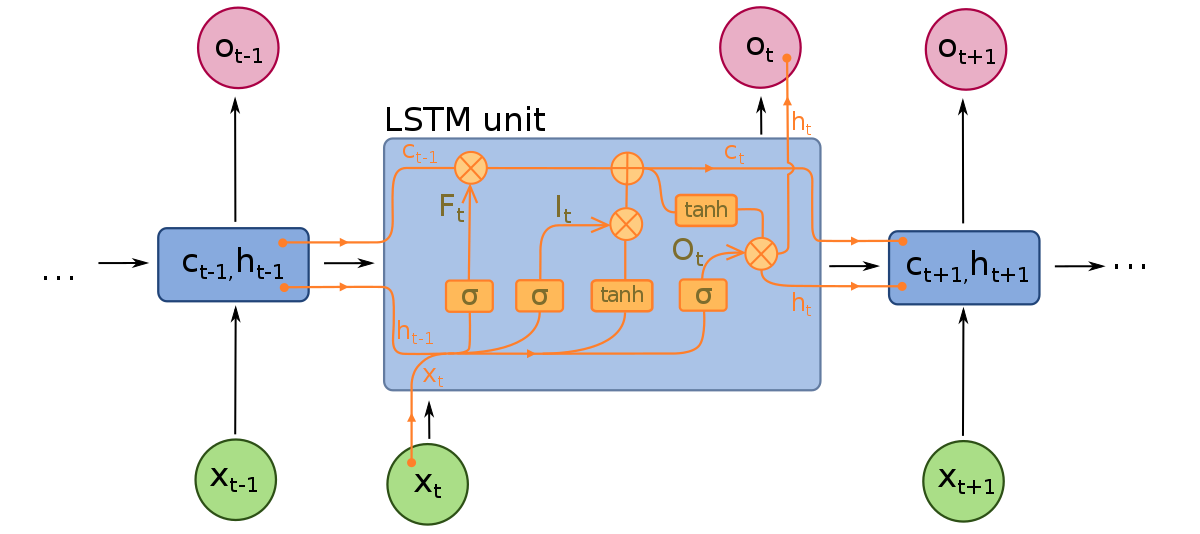
****

Figure 4 LSTM

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.

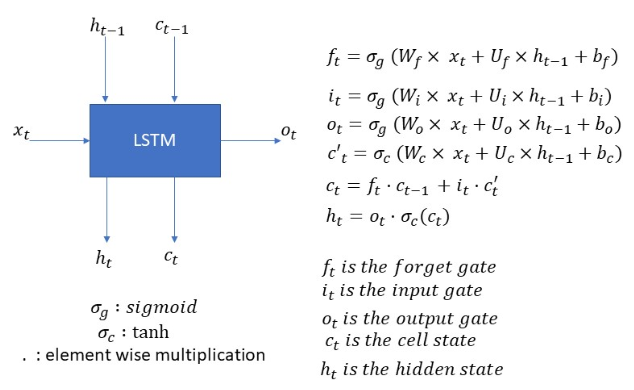
****

Figure 5 : LSTM equations

# **Encoder Decoder (Seq2Seq)**

“**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

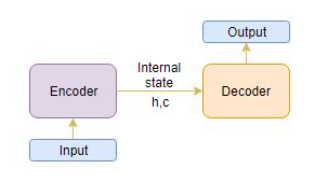


Figure 6 : Seq2Seq

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.

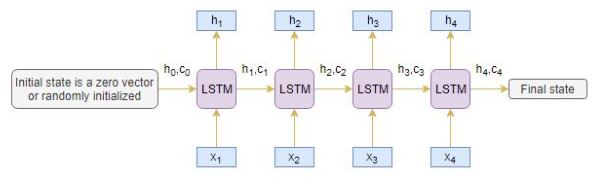
****

Figure 7 / Encoder

(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.

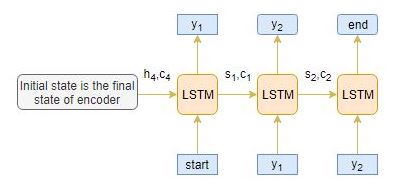


Figure 8 / Decoder

# **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)

# **Software / Frameworks Requirements**

# **Implementation**

# **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”

# **Importing Libraries and frameworks:**

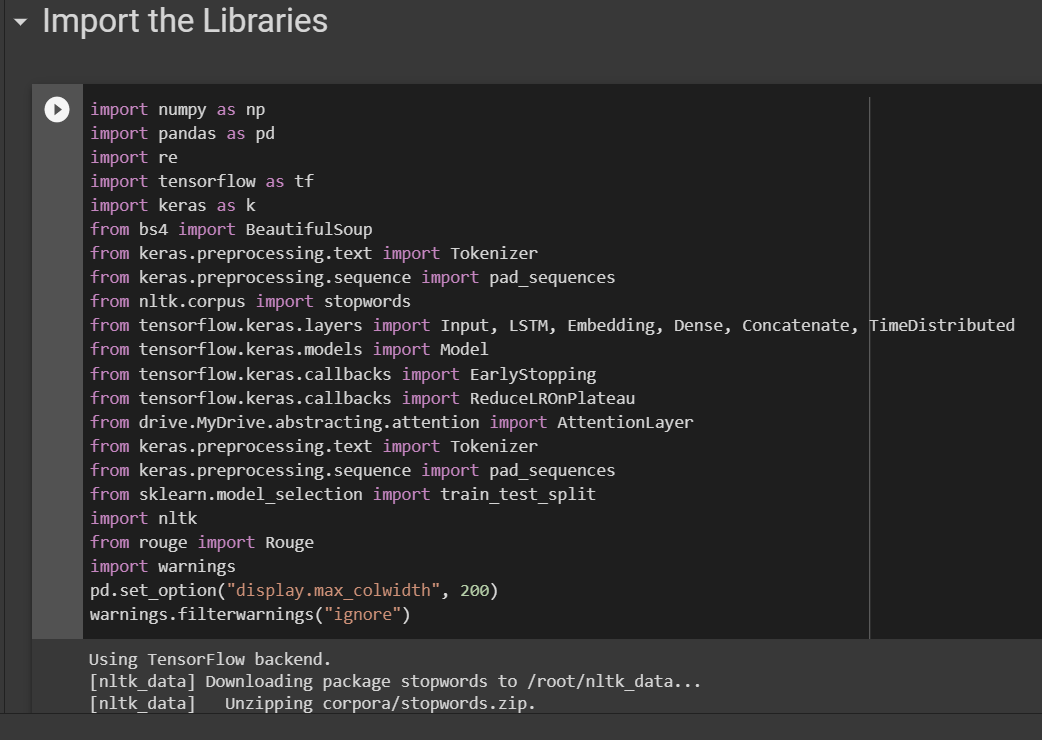
****

Figure 9 / Import the Libraries

# **Getting the Data:**

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

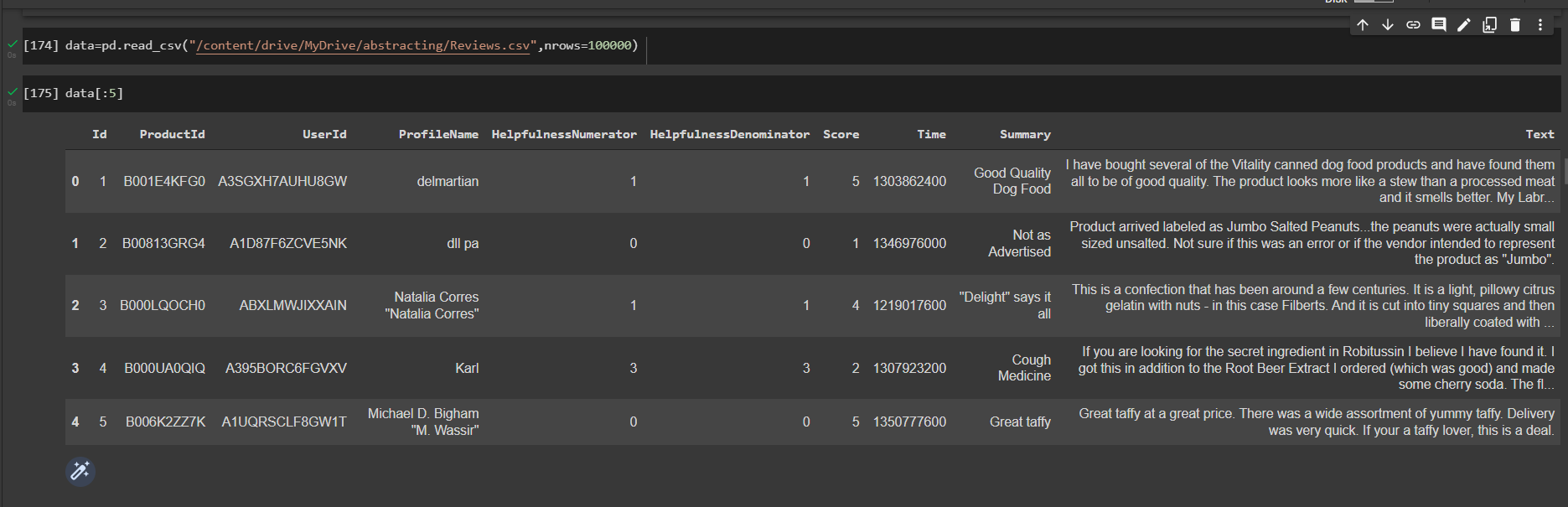


Figure 10 / Getting data

# **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:



Figure 11 / Contraction mapping

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

* Drop Null An Duplicates values

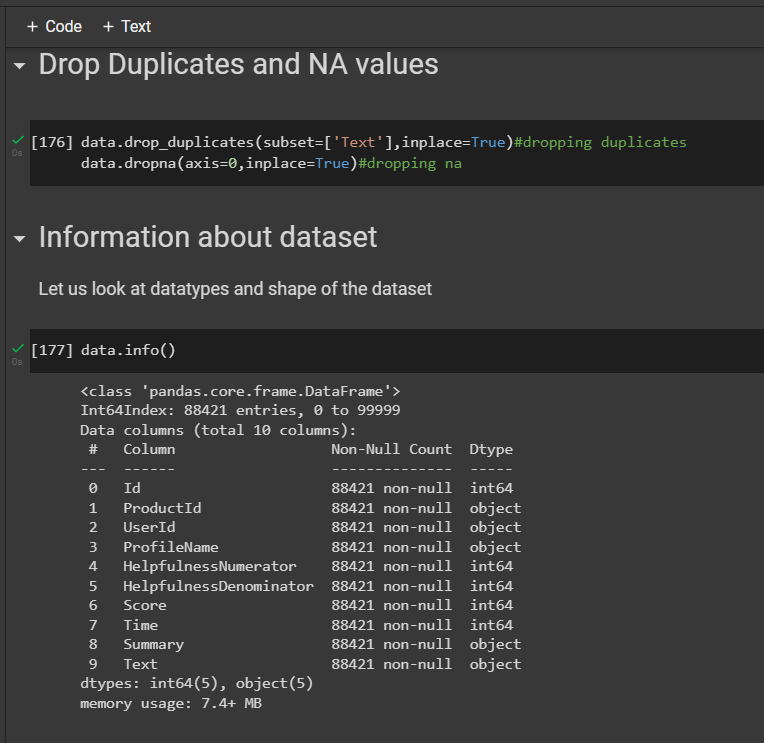


Figure 12 / Drop Duplicates

* 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

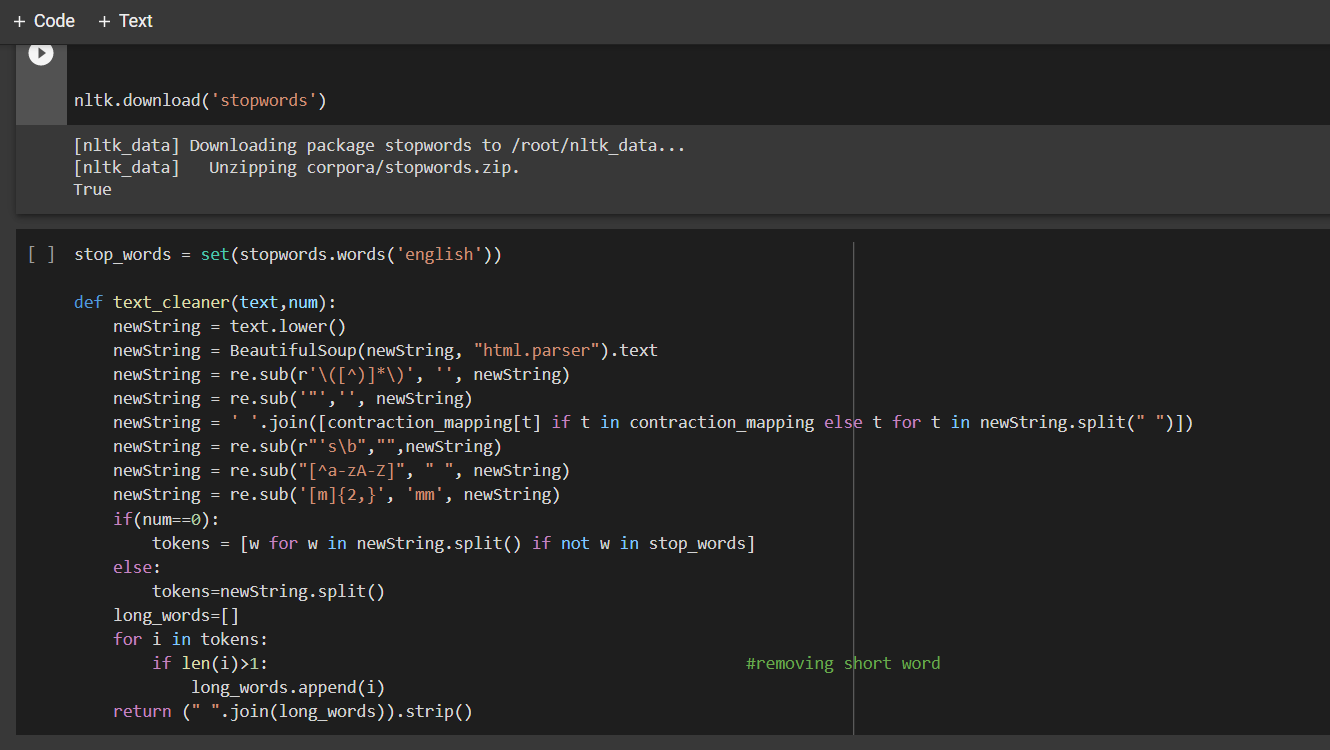
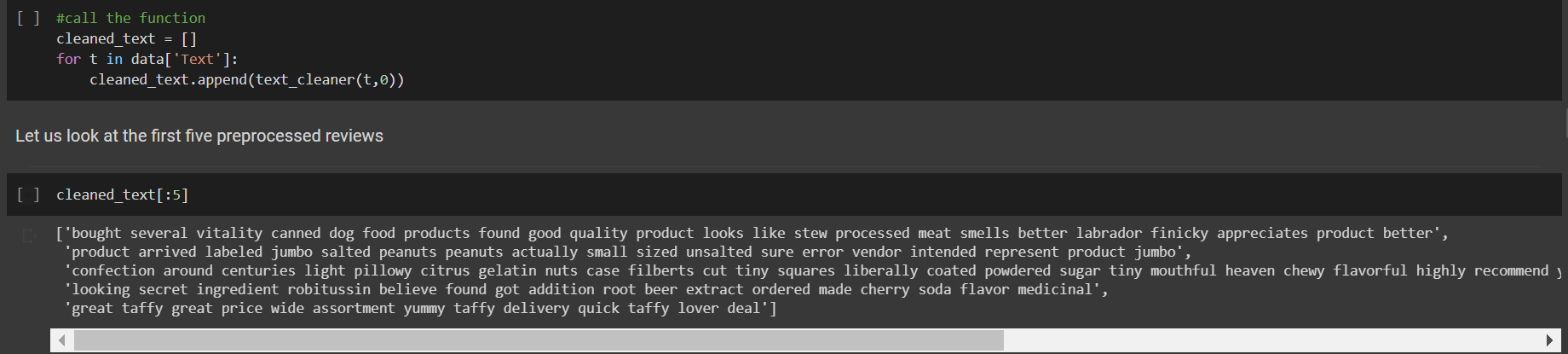
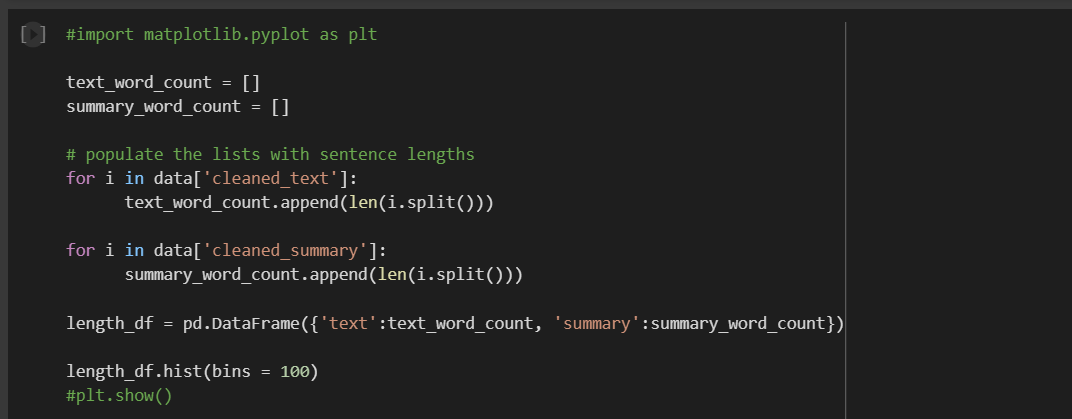


Figure 13 / Data cleaning



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:

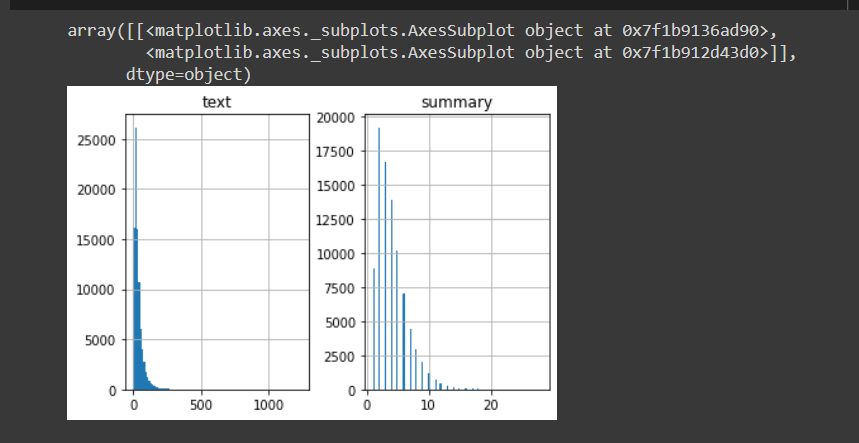


Figure 14 / Data Analysis

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):

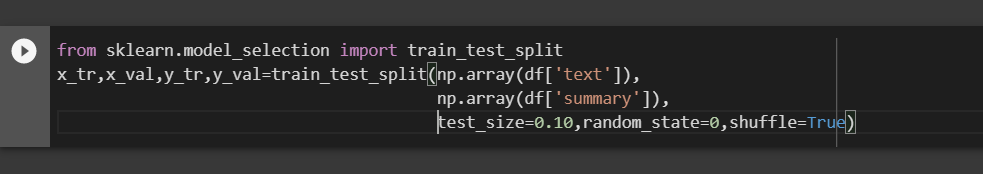


Figure 15 / Dividing the data

* 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:

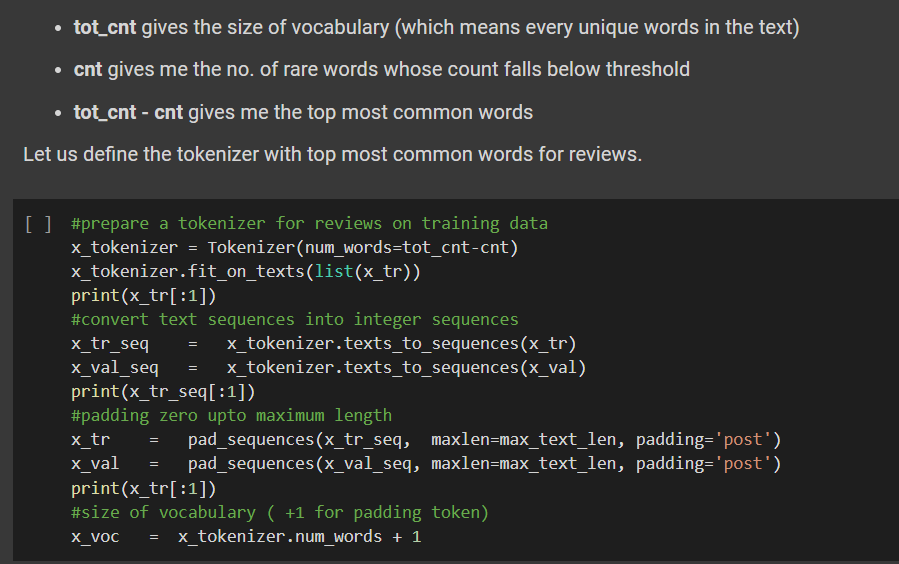
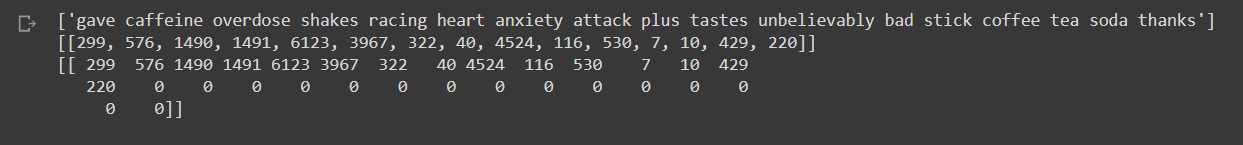


Figure 16 / Tokenizer



# **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) |

Figure 17 / code for model building

# **Model summary :**

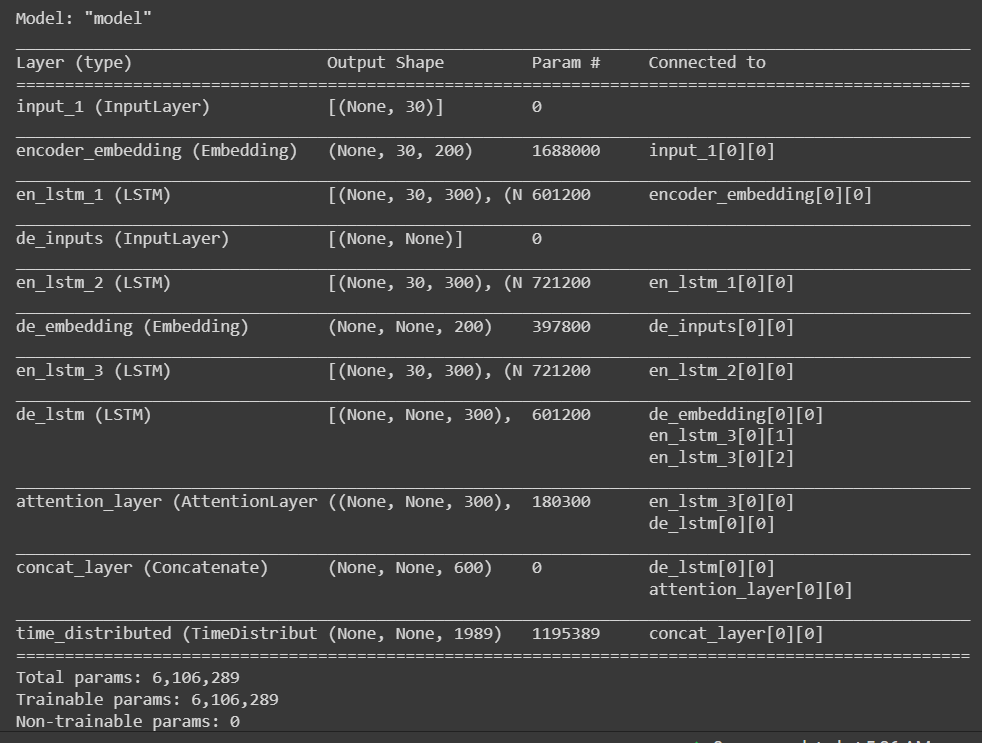


Figure 18 / model summary

# **Model Training:**

* Epochs : 150
* Batch size : 512
* Validation set : 10%

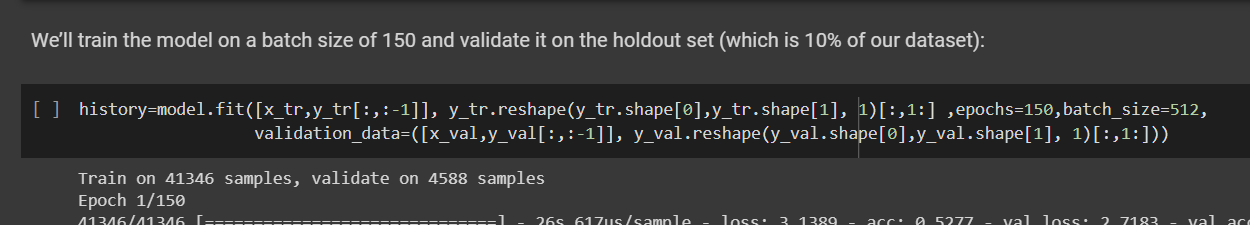
****

Figure 19 / model training

# **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 :

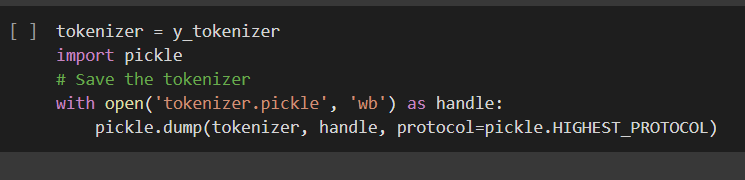
****

Figure 20 / saving the tokenizer

* Saving rest of layers:

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

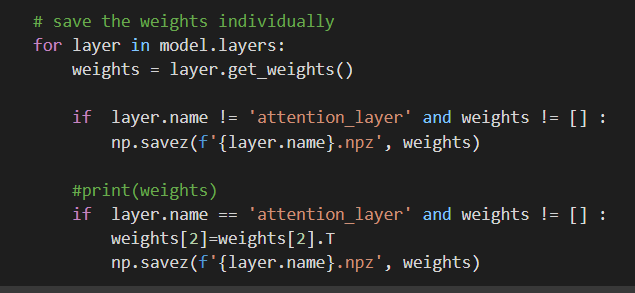


Figure 21 / saving the model layers

# **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]) |

Figure 22 / encoder\_decoder Inference

* 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) |

Figure 23 / Getting the NPZ files

* 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']) |

Figure 24 / Loading the weights to the model

# **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. |

Figure 25 / Example of the API

# **Creating 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,                        ),                      )),                )              ],            ),          ),        ),      );    }  } |

Figure 26 / flutter code

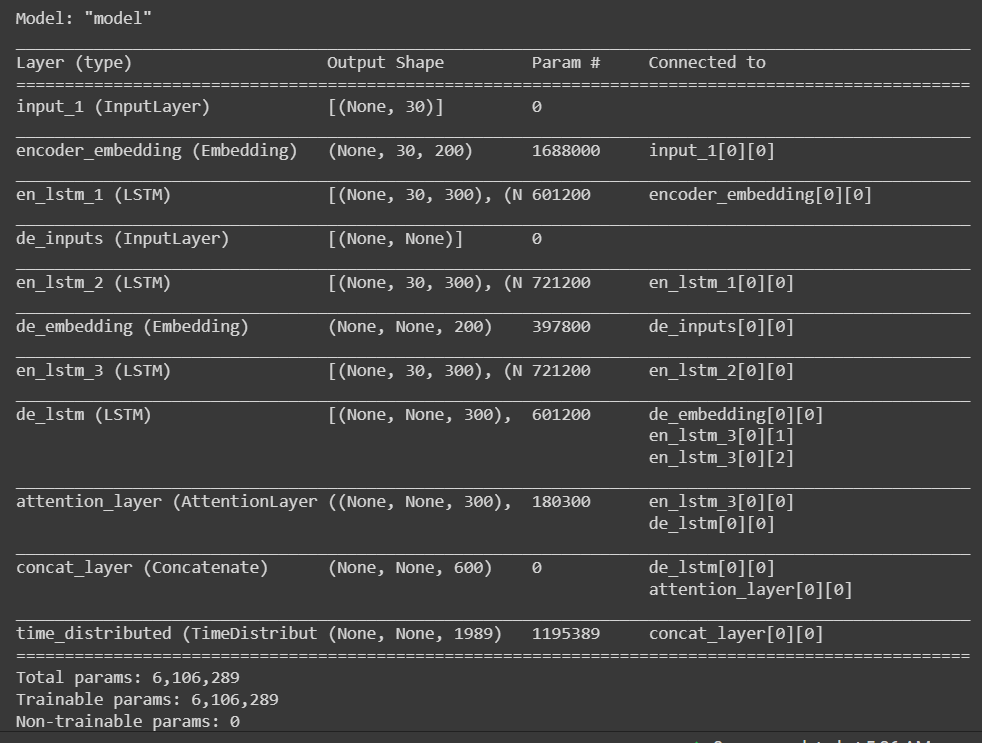
* The second screen contain a filed to display the summary of the text :

|  |
| --- |
| // 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 .**

# **Results of Machine learning model**

# **Model Summary:**



# **Training Result :**

(last 15 epochs)

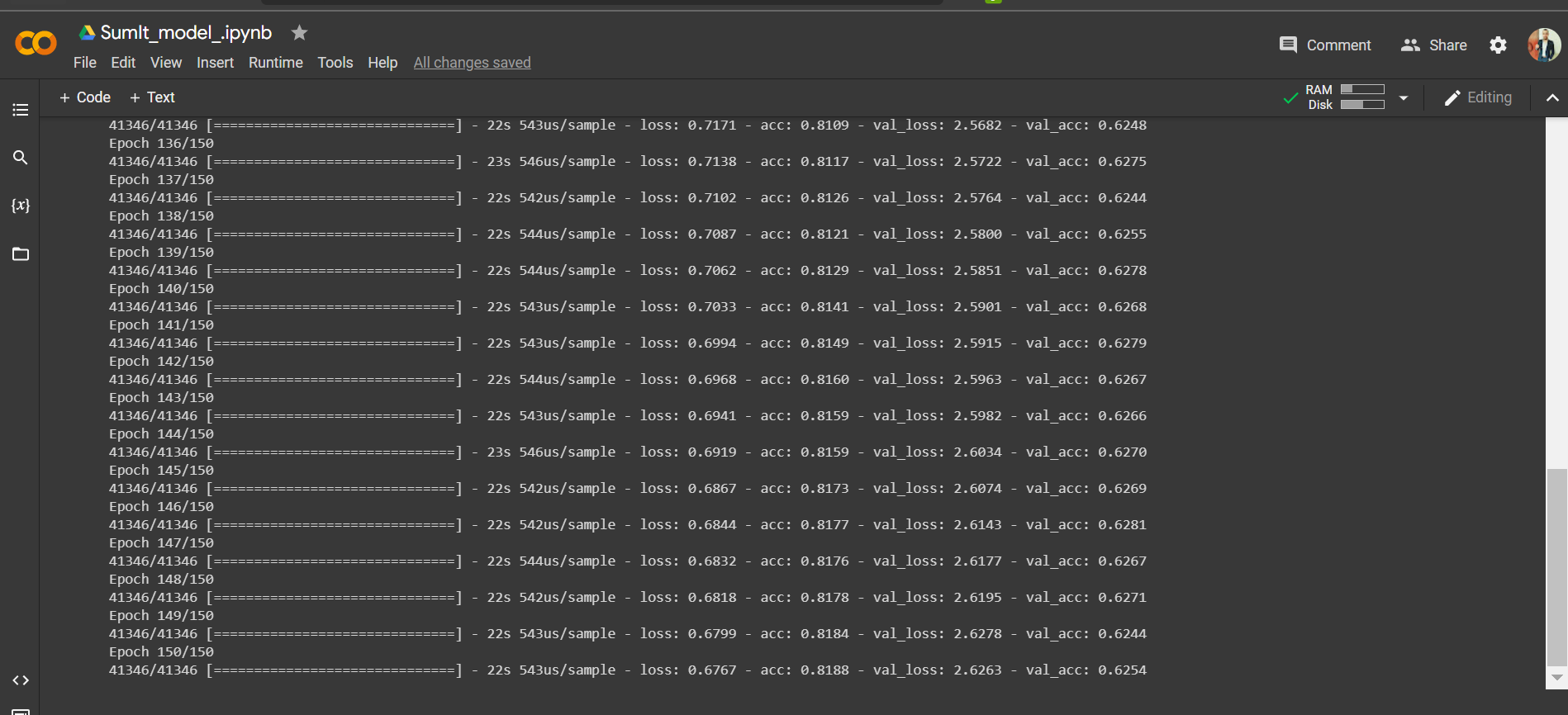


Figure 27 / Training result

# **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.

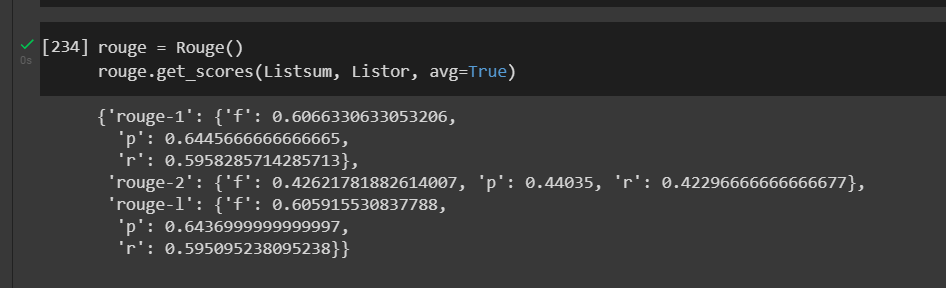


Figure 28 / Rouge test score

# **Analysis after Training :**

The accuracy analysis plot :

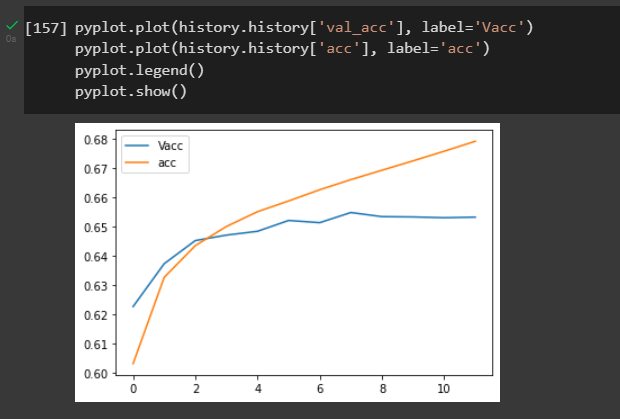


Figure 29 / accuracy analysis plot

The loss analysis plot :

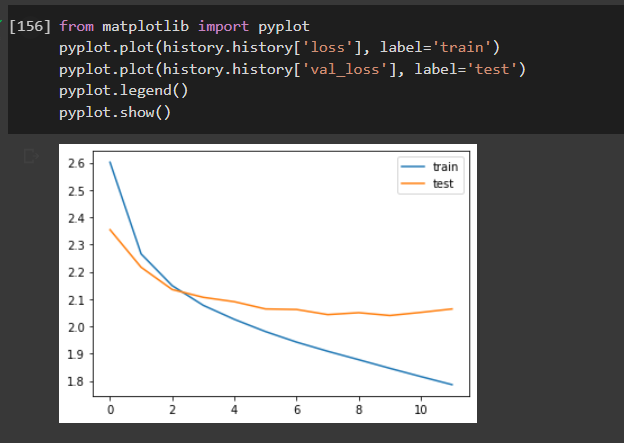


Figure 30 / loss plot

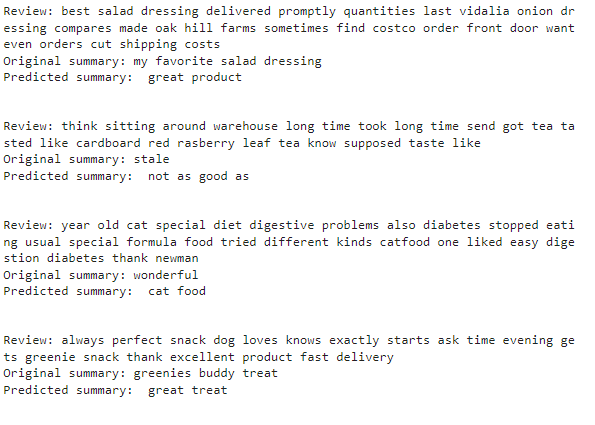
# **Comparing results with model with the same dataset :**

Table 1 / Comparing results with others

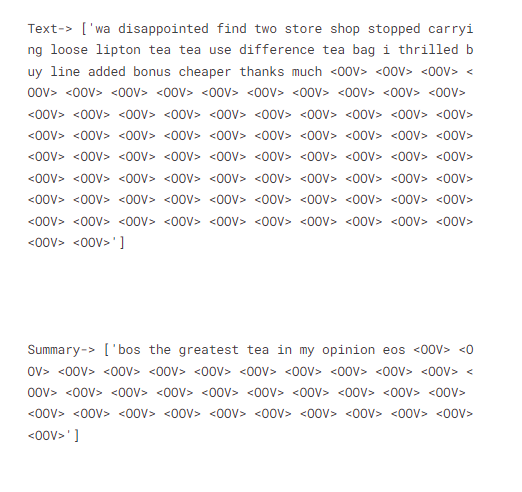
|  |  |  |  |
| --- | --- | --- | --- |
| Source | Acc | Rouge test | Test result |
| (aravindpai, 2019) | 65% | Rouge-1 : 0.20  Rouge-2 : 0.14  Rouge-L : 0.25 | First next screen below |
| (Narayankar, 2021) | 28% | Not implemented | second next screen below |
| Our Model | 81% | Rouge-1 : 0.60  Rouge-2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

Figure 31 / Comparing results with others

* First screen :



* Second screen :



* Third screen :



Figure 32 / test results

# **Results of Flutter App:**

Table 2 / flutter results

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

# **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**

# **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.

# **Conclusion :**

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** (Harvard Style)

# 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].