**Project Description**

**Data Preparation:-**

* First thing we read excel file of training data and show its info to show if there is null or not.

A computer screen shot of a computer code

Description automatically generated

**Preprocessing:**

* **First step :**

Process\_review -> this function takes every review and make on its removal for not important things like digits, spaces, English words, any repeated character replace it with one character.

A screenshot of a computer code

Description automatically generated

* **second step:**

we take values from( train\_data['cleaned\_text']) and store it in variable X .and store values of ( train\_data['rating']) in Y.

A computer code with text

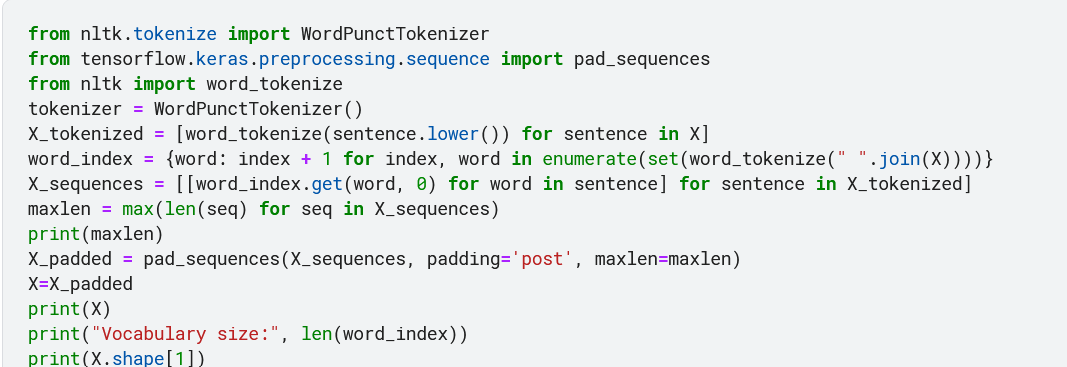
Description automatically generated

* **third step :**
* We split X into words (X\_tokenized) using Wordpunct\_tokenize that recognizes the characters and splits them separately from one another and splits text on whitespace and punctuation.
* Then we make dictionary (word\_index) that contains key and value

Key:- contains unique words from (X) after concatenating all the sentences in the list X into a single string and then tokenizing the string into a list of words

Value:- it contains unique index for each word

* Then For each word in a sentence (X\_tokenized), this retrieves its corresponding index from ( word\_index). If the word is not in the dictionary ( word\_index), it returns 0. Then padding sequences with max length.



* **forth step :**

we convert Y values to numerical labels using label encoding.

A close-up of a computer code

Description automatically generated

* **fifth step :**

we split data into train(60%), test(20%) and validation(20%).

A close-up of a computer code

Description automatically generated

**LSTM model:-**

* **Layers:-**

1. Embedding layer :- it transforms discrete words or tokens into continuous vector representations that can be further processed by subsequent layers in the model.
2. Bidirectional processing :- The LSTM layer processes the input sequences both forward and backward, capturing information from both directions.
3. LSTM layer:- determining the dimensionality of the output space. We put (100) neurons and use dropout (0.5) which helps prevent overfitting.
4. GlobalMaxPool1D:- it reduces the sequence to a fixed-size representation by taking the maximum value along each feature dimension and helps capture the most important information from the input sequences.
5. Dense layer with (relu) activation function (num= 3 layers):- it is fully connected layer that helps to further process the extracted features and make predictions based on the learned representations.
6. Dropout layer with value (0.5)(num=3 ) after dense layer :- it is a regularization technique that helps prevent overfitting in neural networks by randomly setting a fraction of input units to zero during training.
7. Dense is the last layer with (softmax) activation function.

A computer screen shot of a program

Description automatically generated

* We apply model on data and save it .

A computer code with text

Description automatically generated

A screenshot of a computer code

Description automatically generated

A close-up of a number

Description automatically generated

* **Evaluation:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **trails** | **Batch size** | **loss** | **epoch** | **Test accuracy** |
| 1 | 64 | binary\_crossentropy | 4 | 0.8455 |
| 2 | 32 | Sparse\_categorical\_crossentropy | 2 | 0.8311 |
| 3 | 64 | Sparse\_categorical\_crossentropy | **4** | 0.8288 |
| 4 | 32 | sparse\_categorical\_crossentropy | 5 | 0.817 |

* **Test file:-**

We read the excel file of test and make preprocessing(first and third step) on it that apply on file of train. Then apply model on it.

A computer screen shot of a program

Description automatically generated

We save prediction labels in csv file.

A screenshot of a computer code

Description automatically generated

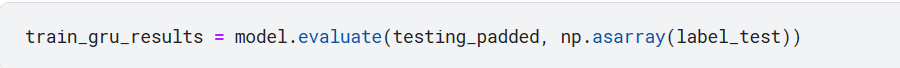
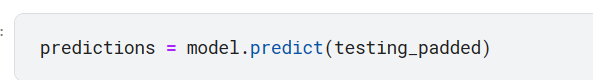
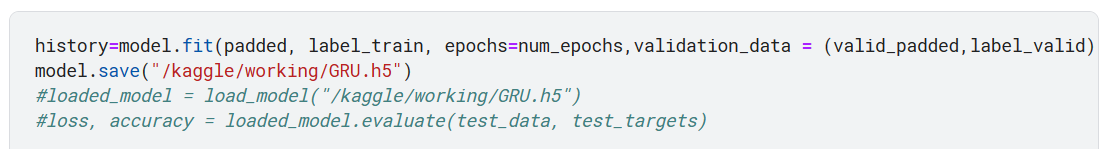
**GRU model:-**

* **Layers:-**

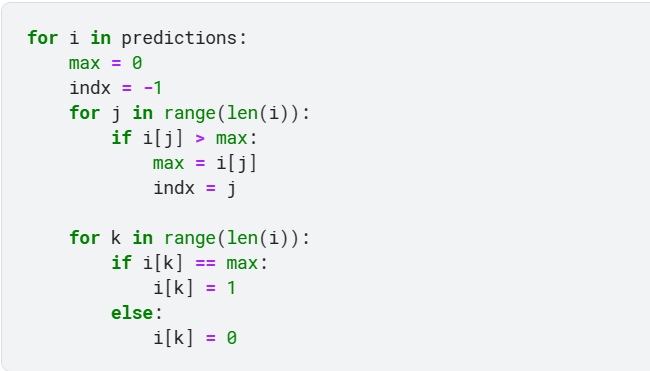
1. Embedding layer :- it transforms discrete words or tokens into continuous vector representations that can be further processed by subsequent layers in the model.
2. GRU layer:- determining the dimensionality of the output space. We put (32) neurons and use dropout (0.5) which helps prevent overfitting.
3. Dropout layer with value (0.5) after dense layer :- it is a regularization technique that helps prevent overfitting in neural networks by randomly setting a fraction of input units to zero during training.
4. Dense is the last layer with (sigmoid) (num of units =3 ) activation function.



* We apply model on data and save it



* Then we convert the output to binary list to be similar to labels



* **Evaluation:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **trails** | **loss** | **epoch** | **Test accuracy** |
| 1 | binary\_crossentropy | 5 | 0.80 |
| 2 | binary\_crossentropy | 10 | 0.80 |
| 3 | categorical\_crossentropy | **14** | 0.81 |
| 4 | categorical\_crossentropy | 5 | 0.80 |

* **Transformer model:-**

tokenization step :

we take the sentance and separate it with the spaces that found between words.

A screen shot of a computer program

Description automatically generated

stop words removal:

here we collect all arabic stop words in language and we remove all of them from the data.

A screen shot of a computer program

Description automatically generated

Then we convert the every sentence to array of words.

Then we convert every array to array of sequences.

And finally we make padding so all data are equivalent to each other.

**Layers:-**

Our transformer block contains:

A computer code with many text

Description automatically generated with medium confidence

1. Multi-head attention layer: allows the model to focus on different parts of the input sequence simultaneously, enabling it to capture diverse patterns and relationships in the data.
2. Dropout layer: reduce the overfitting as it scale the input to 1/(1-rate).
3. Add & Norm layer: helps to reduce the training time by a small fraction and stabilize the network.
4. Feed forward layer: have two linear layers with a non-linear activation function (such as ReLU) in between.
5. Add & Norm layer.

Our model contains:

A computer code with text

Description automatically generated

1. Input Embedding layer: This layer maps each word in the input to a fixed-length vector, which helps the model capture the meaning and context of the words.
2. Transformer block layer.
3. GlobalAveragePooling1D layer: Used for reduce the spatial dimensions of the input sequence.
4. Dense layer: Used for classify the input to number of classes.

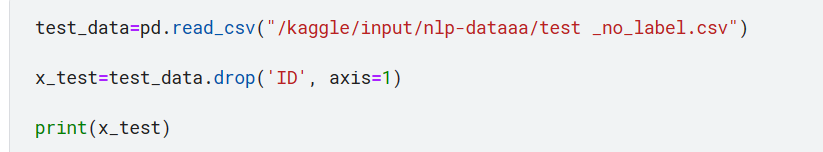
## Evaluation:

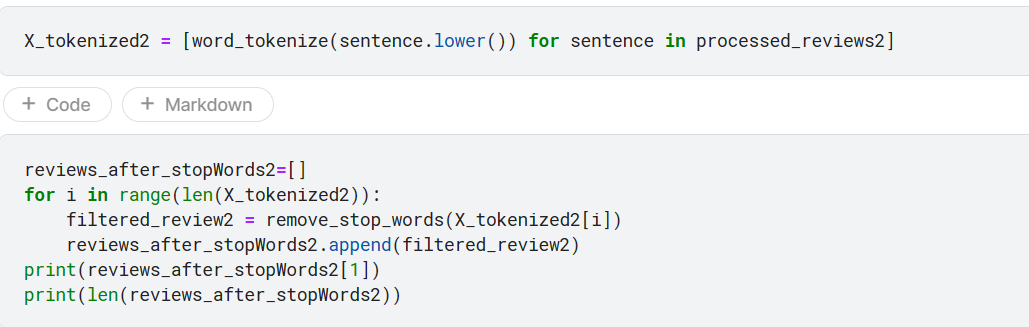
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epochs | Learning Rate | With/without positional layer | Accuracy | loss |
| 10 | 0.1 | without | 0.7641 | 0.8147 |
| 3 | 0.001 | with | 0.79799 | 0.3215 |
| 5 | 0.01 | With | 0.7995 | 0.3673 |
| 3 | 0.001 | without | 0. 825 | 0. 3835 |

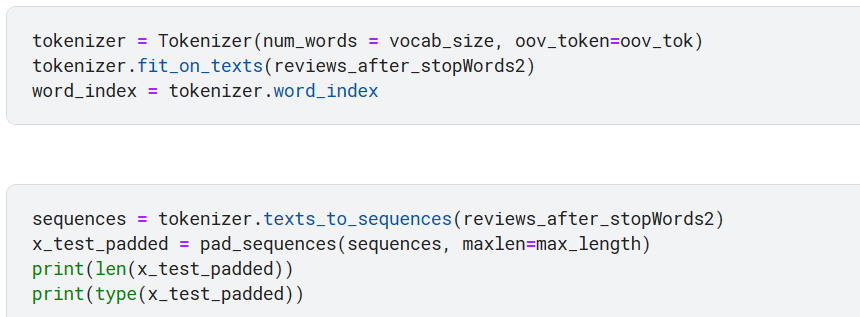
So we found that without the positional layer the accuracy is better so we removed it from the model.

* **Test file:-**

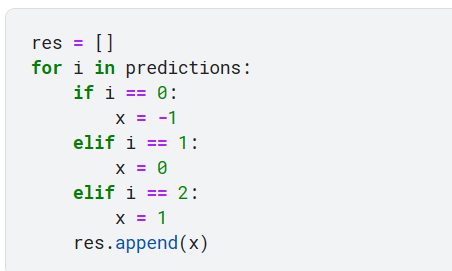
We read the excel file of test and make preprocessing(first and third step) on it that apply on file of train. Then apply model on it.

****





At last we convert binary lists into either -1, 0, or 1



We save prediction labels in csv file.

