ANALYZING THE MODEL PERFORMANCES FOR THE TEXTBOOK TEXT PERFORMING TEXT SUMMARIZATION USING CUSTOM DEEP LEARNING MODEL AND STATE-OF-THE-ART TEXT SUMMARIZATION MODELS.

GROUP 3

INCREMENT 1

Team Members :

Hari Vamshi Krishna (11614861)
Manish Reddy Radha Reddy (11518946)
Ramya Sree Konduru (11601633)
Ranjith Kumar Sadafule (11602281)

GitHub Repository of the Project: https://github.com/thor56/NLPTextSummarization

Goals and Objectives

Motivation:

The main motivation of this project came from preparing summarized notes for exam preparation. Generally, we have our lecture notes, but they are lengthy and has lots of content in it and we really can't use the lecture notes for the exam preparation because, it is covered of small and general examples which help us to understand the concept, repeated concepts with different approach. Let us suppose we have 500 pages of lecture notes, and we

can't go through all the 500pages all the time when we are preparing for our exam. So, to overcome this we make a short note of all the important points and convert the 500 pages notes to 10 pages or so depending on the requirement.

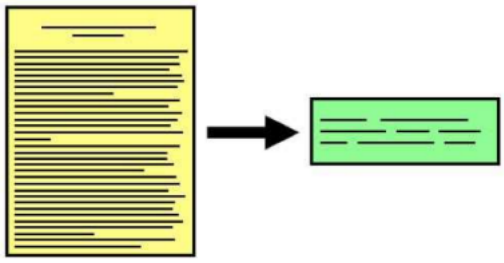

Figure-1: Displaying the Text summarization representation

In the above figure 1, displays the pictorial representation of text summarization, where it shows that 50 lines in one document are summarized into 5 lines.

So, our main aim is to create a text summarization model using Neural network which will provide meaningful sentences without missing the information.

Significance:

The text summarization cannot only be used in the lecture notes summarization, when we look at the broader picture, it can be used for Media monitoring, Newsletters, Search Marketing and SEO, Internal document workflow, financial research, Legal Contract analysis,

social media marketing and many more. People always need their work to be reduced. So, in

the context of reading and making short notes text summarization will be very useful. Important aspects of the text are identified by the Deep Learning model and retained and

reframed to represent the same content.

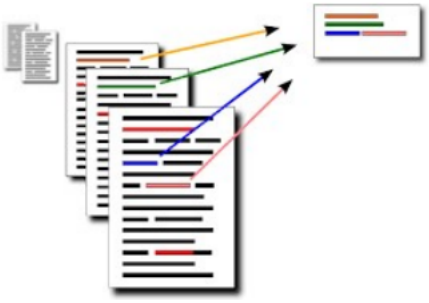

Figure-2: Illustrating the application of Text Summarization.

Objectives:

The main objective of our project is to deliver a text summarization model which performs good when compared to the State of Art models and provide meaningful sentences, without missing any important information. Here, we will be working on different models that are present for text summarization and compare the performance of the model. Using tools and implementations of NLP, the model could obtain consistent output that represents the original context as is but with significantly smaller text content.

Features:

Deep Learning modelling helps in enhancing the identification of keywords in the text, this improvement leads to summarizing the text without missing any important information. Suppose we have different keywords with the same meaning, then this situation can also be overcome by the deep learning model and provide us best results. The working model will reduce the length of text of given context to smaller text length while maintaining coherent and fluent summary which contain only key points of the tex

Related work

The current situation has been concluded from the existing work that has been done on Amazon product reviews to summarize. Several approaches have been devised for the summarization in general, summarizing lecture notes should have similar treatment on the dataset and implementation to summarize the notes. From summarizing customer reviews on online platforms to summarizing Tweets and movie storylines have been shown have great accuracy,

this has been the right motivation to develop a model to be able to summarize lecture notes to save time for students while going through the lecture notes.

Dataset

The dataset has been collected from Kaggle and here the data has been segregated into paragraphs from the lecture. There are various attributes in this dataset such as Title, Department, Field of study, Description and conclusion. All these attributes are properly analyzed for summarizing on point with highest accuracy. In the working of the project we worked with text data and trained the deep learning model by giving the unsummarized and summarized text, so that the deep learning model can learn how to neglect the sentences in a paragraph.

The dataset that we used is small paragraphs from the school social textbooks, and we will see the results of summarized texts in the paper below. The source of the project dataset is from Kaggle, we couldn't get the direct dataset to perform modelling, we had to segregate the dataset arrange the text data properly. The dataset that we used is pushed in the project git repository: https://github.com/thor56/NLPTextSummarization/Dataset

Detailed design of Features

The core design of the model is to summarize and shorten the lecture notes while retaining the amount of information initially conveyed.

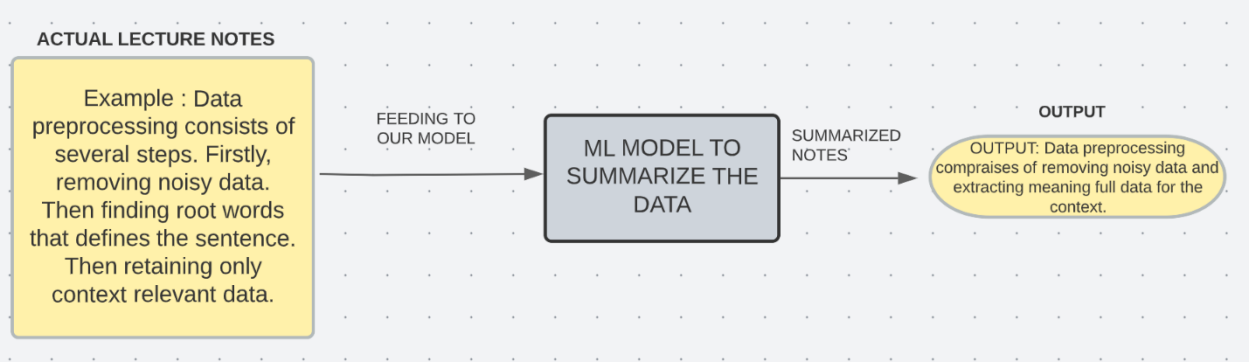

Figure 3: Core feature of the Model

(Source: Self-developed)

The model uses Machine Learning implementation to process and analyze the data in the lecture notes and based on training, It produces shortened notes. It is integrated with several modeling steps to reproduce the exact same meaning of the context.

Analysis

It was initially analyzed that, based on different subjects, the accuracy of the model differs slightly. However, that is a problem for the final increment as the current state of the model still attains great accuracy and outcome. Not a lot of data is required to train the model but significant amount is required to be able keep the summarizing in relevancy to the context.

Implementation

Initially we perform the data preprocessing steps in order to be able to work with the only important data for building the model upon.

Drop Duplicates and NA values

```
data.drop_duplicates(subset=['Text'],inplace=True)#dropping duplicates
data.dropna(axis=0,inplace=True)#dropping na
```

Information about dataset

Let us look at datatypes and shape of the dataset

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 162834 entries, 0 to 199999
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      162834 non-null  int64
 1   ProductId               162834 non-null  object
 2   UserId                  162834 non-null  object
 3   ProfileName             162834 non-null  object
 4   HelpfulnessNumerator    162834 non-null  int64
 5   HelpfulnessDenominator  162834 non-null  int64
 6   Score                   162834 non-null  int64
 7   Time                    162834 non-null  int64
 8   Summary                 162834 non-null  object
 9   Text                    162834 non-null  object
dtypes: int64(5), object(5)
memory usage: 13.7+ MB
```

Figure 4: About the dataset and preprocessing

(Source: Self-developed)

Our model follows Sequence to sequence modeling to process any sequential information.

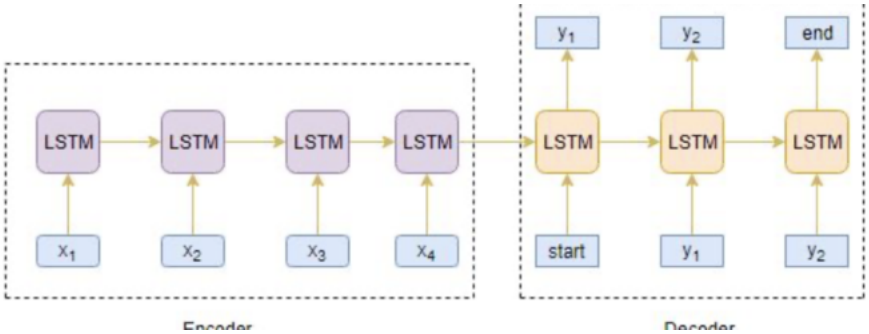

Figure 5: Principle working of the model

(Source: Self-developed)

The Seq 2 Seq model constitute of 2 major components which are Encoder and Decoder phases. Either Gated Recurrent Neural Network or Long Short-Term Memory (LSTM) is preferred for encoding and decoding components. While we choose LSTM

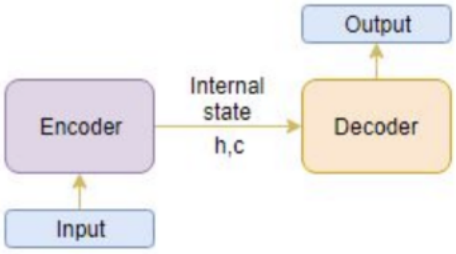

Figure 6: Illustration of the underlying working of Model

(Source: Self-developed)

The LSTM Encoder reads the entire input and passes one word as a token each time from which the context is extracted. Then, LSTM Decoder predicts same sequences with certain offset and it is also taught to anticipate following words.

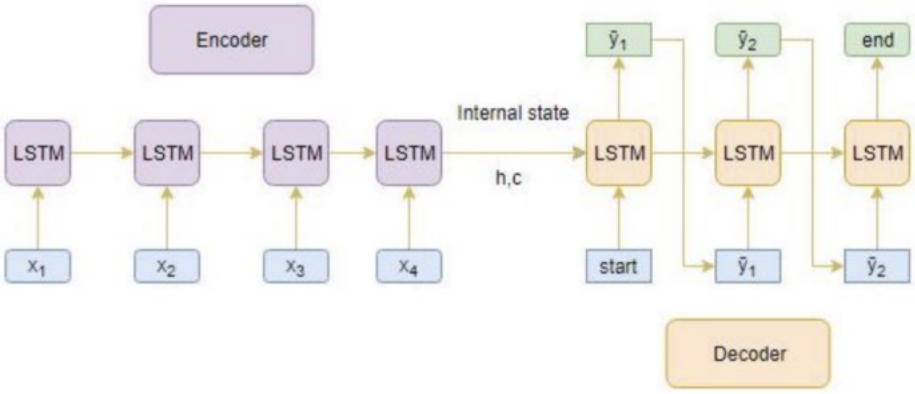

Figure 7: Formation of comprehension

(Source: Self-developed)

The initial breakdown and processing of our model on the dataset is visualized as follows:

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 30)]         0           []

 embedding (Embedding)          (None, 30, 100)      1146600     ['input_1[0][0]']

 lstm (LSTM)                    [(None, 30, 300),    481200      ['embedding[0][0]']
                                 (None, 300),
                                 (None, 300)]

 input_2 (InputLayer)           [(None, None)]       0           []

 lstm_1 (LSTM)                  [(None, 30, 300),    721200      ['lstm[0][0]']
                                 (None, 300),
                                 (None, 300)]

 embedding_1 (Embedding)        (None, None, 100)    296500      ['input_2[0][0]']

 lstm_2 (LSTM)                  [(None, 30, 300),    721200      ['lstm_1[0][0]']
                                 (None, 300),
                                 (None, 300)]

 lstm_3 (LSTM)                  [(None, None, 300),  481200      ['embedding_1[0][0]',
                                 (None, 300),                     'lstm_2[0][1]',
                                 (None, 300)]                     'lstm_2[0][2]']

 attention_layer (AttentionLaye  ((None, None, 300),  180300     ['lstm_2[0][0]',
 r)                              (None, None, 30))                'lstm_3[0][0]']

 concat_layer (Concatenate)     (None, None, 600)    0           ['lstm_3[0][0]',
                                                                  'attention_layer[0][0]']

 time_distributed (TimeDistribu  (None, None, 2965)  1781965     ['concat_layer[0][0]']
 ted)

==================================================================================================
Total params: 5,810,165
Trainable params: 5,810,165
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 8: LSTM and Attention layer breakdown

(Source: Self-developed)

The implementation involves attention layer which is used to define the scope of the word or phrase with respect to rest of the context. We have used about 10 Epoch to train our model.

```
history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=10,callbacks=[es],batch_size=128
Epoch 1/10
604/604 [==============================] - 657s 1s/step - loss: 2.8004 - val_loss: 2.5370
Epoch 2/10
604/604 [==============================] - 655s 1s/step - loss: 2.4766 - val_loss: 2.3749
Epoch 3/10
604/604 [==============================] - 636s 1s/step - loss: 2.3480 - val_loss: 2.2856
Epoch 4/10
604/604 [==============================] - 636s 1s/step - loss: 2.2699 - val_loss: 2.2358
Epoch 5/10
604/604 [==============================] - 636s 1s/step - loss: 2.2142 - val_loss: 2.1954
Epoch 6/10
604/604 [==============================] - 644s 1s/step - loss: 2.1674 - val_loss: 2.1604
Epoch 7/10
604/604 [==============================] - 648s 1s/step - loss: 2.1296 - val_loss: 2.1364
Epoch 8/10
604/604 [==============================] - 656s 1s/step - loss: 2.0967 - val_loss: 2.1183
Epoch 9/10
604/604 [==============================] - 660s 1s/step - loss: 2.0679 - val_loss: 2.0968
Epoch 10/10
604/604 [==============================] - 642s 1s/step - loss: 2.0423 - val_loss: 2.0852
```

Figure 9: Model training

(Source: Self-developed)

Preliminary results

The initial results seem promising with close to actual summary of the text. We have generated many summarized out of which some are produced below

```
In [55]: for i in range(0,100):
             print("Review:",seq2text(x_tr[i]))
             print("Original summary:",seq2summary(y_tr[i]))
             print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,max_text_len)))
             print("\n")

Review: policy use non gmo products possible cannot guarantee product produce
d non gmo corn low prices majority corn produced us gmo assume gmo product gr
eat amazon return policy food items stuck
Original summary: warning they use corn for this product
1/1 [==============================] - 0s 353ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
Predicted summary:  not worth the money
```

```
In [55]: for i in range(0,100):
             print("Review:",seq2text(x_tr[i]))
             print("Original summary:",seq2summary(y_tr[i]))
             print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,max_text_len)))
             print("\n")
```

```
Predicted summary:  not the best


Review: special treat babies pricey would think dogs would like feel look lik
e vegetable shaped crayons full sorts goodies although feeding grain free die
t first ingredient hydrolyzed wheat protein
Original summary: my dogs love love love
1/1 [==============================] - 0s 369ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
Predicted summary:  my dogs love these
```

```
Review: gluten free oatmeal delicious always think like apple cinnamon flavor
better fast shipping exceptional pricing
Original summary: delicious
1/1 [==============================] - 0s 350ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
Predicted summary:  great gluten free snack
```

As we can see, the comprehended text already seems pretty close to the actual expected outcome.

Project management

Work completed:

Description

Dataset has been collected from Kaggle and preprocessing has been performed on it. Basing on the LSTM modeling technique, a basic ML model has been developed which processes by breaking down individual text and predicting next in the sequence and matching it with the

actual following word. This is called Sequence to Sequence modeling. The data is feed to Encoder which breaks down the phrase into words and send them as input to the decoder which predicts the next word based on the key context. This also involves custom attention layer which is responsible to maintain relevancy factor with respect to the key context (distinguishing whether the phrase is local or global context).

Responsibility :

Manish Reddy Radha Reddy : Analysed the existing models for similar goals and decided the suitable approach. Furthermore, is working on increasing the accuracy of the model based on the State-of-the-Art model available out there.
Ramya Sree Konduru : Gathered the Dataset and preprocessed the data for the training to be carried out. Documented the teamwork and process.
Hari Vamshi Krishna : Defined the suitable Model and approach to build the model. Tried and built the model based on ML approach
Ranjith Kumar Sadafule : Performed model training and model testing and validation on the processed dataset and produced the preliminary results. Documented the teamwork and process.

Contribution:
Manish Reddy Radha Reddy : 25%
Ramya Sree Konduru : 20%
Hari Vamsi Krishna Samayamantri :30%
Ranjith Kumar Sadafule :25%

Work To Be Completed:
In the next project increment we will be focusing on working on the state-of-the-art models and see how those models are summarizing the text, we will also be focusing on the accuracy metrices such as ROUGE-N, ROUGE-L, SacreBLEU.

Description:

The initial implementation has worked pretty well in favor our model compared to the actual output. Although there can be a lot of improvement when compared to the existing State-of-the-Art model that perform similar operations. In developing the deep leaning models we had a hard time in fitting the data into the model and attaining the results, we had to change the data to tensors and then push the data to the model. We are working towards increasing the accuracy of out model to be on par with the SOTA models as mentioned.

Responsibility and Contribution:

Manish Reddy Radha Reddy : Adding improvements to the existing model to attain greater accuracy. Analyzing the results with state of the art models and developing the accuracy metrices to calculate the score of the results.

Ramya Sree Konduru : Make potential changes to data preprocessing steps to eliminate some more irrelevant data and retain context-familiar data

Hari Vamshi Krishna : Analyze differences in key working mechanism of the State of the art model and our models to identify key performance pull backs.

Ranjith Kumar Sadafule : Run multiple combinations of model training and testing

References

- Text summarization using unsupervised deep learning. Author links open overlay panel MahmoodYousefi-AzarLenHamey
- Abstractive text summarization using LSTM-CNN based deep learning
  Song, Shengli; Huang, Haitao; Ruan, Tongxiao.
- Deep Extractive Text Summarization. Author links open overlay panel Rupal Bhargava [a]Yashvardhan Sharma[a]
- https://ieeexplore.ieee.org/document/7396858
- https://dl.acm.org/doi/10.1145/2857546.2857619