**Table of Content \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Acknowledgement \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* 2

*Declaration \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3*

1. ***Problem Statement \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 6***
2. **Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 7**
   1. Extractive Method
   2. Abstractive Method
3. **Research Paper Analysis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8**
4. **Work Design \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9**
5. **Basic Insight \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 10**
   1. Deep Learning
   2. RNN
   3. LSTM
   4. Encoder Decoder
6. **Implementation \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 15**
7. **Experimental Results and Analysis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 17**
   1. Simple Encoder Decoder
   2. Teacher Enforcing Encoder Decoder
8. **Conclusion of the Report and Future Scope \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 19**
9. **References \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20**
10. **Problem Statement:**

**Issue:**

There is too much text to read and too little time. Currently, there are vast quantities of textual data available, including online documents, articles, news, and reviews that contain long strings of text that need to be summarised.

**Solution:**

With the help of Text Summarization one can easily and quickly gather relevant information, and hence have a quick glance at the important points the article has to offer. This reduces the time and energy required to go through lengthy passage.

**Our Approach:**

# In this work we have experimented with standard Long Short-Term Memory (LSTM) sequence-to-sequence encoder – decoder Model and teacher enforced Encoder-Decoder Model. This model utilizes LSTM cell for storing long sequences which prior using simple RNN cell was not possible. We develop both LSTM based Simple Encoder-Decoder model and LSTM based Teacher Forcing Encoder-Decoder Model for generating each word of the summary conditioned on the input sentence.

# While the model is structurally simple, it can easily be trained end-to-end and scales to a large amount of training data. We apply our model to the Fine-food-review dataset. We evaluate the reconstructed paragraph using standard metrics like ROUGE, showing that neural models can encode texts in a way that preserve syntactic, semantic, and discourse coherence.

# 2. INTRODUCTION:

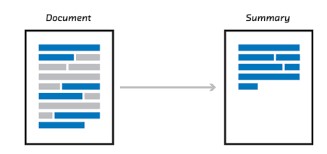
Text summarizing is the process of gathering the essential information and the overall meaning of a text and producing a quick and simple summary. Natural language processing techniques, such as page rank algorithms and others, are used to summarize text. While these algorithms achieve the goal of text summarization, they are unable to construct new sentences that are not present in the material, like people can. They may also contain grammatical mistakes. Deep Learning comes to our rescue in this situation. We can utilize deep learning to create a model for text summarization that is both efficient and quick. The use of deep learning technologies allows us to create summaries that are both grammatically correct and can be constructed with new phrases and sentences.

Text Summarization is broadly classified into two types:

1. Abstractive Summarization
2. Extractive Summarization

## 2.1 Extractive Text Summarization

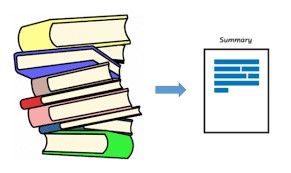
The extractive text summarization involves taking out key phrases from the source/original document and combining them to make a summary. We identify important words or phrases from the text and extract only those for the summary.



**Fig -1:** Extractive Text Summarization

## 2.2 Abstractive Text Summarization

The abstractive text summarization can create new phrases and sentences that relay the most useful information from the original text. The sentences generated through this method may not be present in the original document.



**Fig -2:** Abstractive Text Summarization

**3. Research Paper Analysis:**

We have studied around Ten Research and Analyzed 4 Dataset to proceed with our project.

Nallapati R, Zhai F, Zhou B we concluded that with extractive text summarization using RNN network-based sequence model, it was not possible to generate summarized text in proper meaningful sentence. This approach was limited to length of sentence as well as type of summarization that is termed as Extractive Text Summarization.

Using LSTM cell will solve the memory issue. LSTM cell when compared to RNN cell are 10-20x more capable of holding data. This helps to store and process large sequence at time, without forgetting the initial input.

Now with the help of Encoder-Decoder Model, we are able to take all the data as input and process on it as a whole rather then processing on an individual input at a time. Use of Encoder – Decoder also allows up to process data of different length. That is with the help of encoder and decoder LSTM architecture, we can process input data of length X and output data of length y.

With the help of Bidirectional RNN, applied on LSTM cell, model is able to see in the future, that is the word present at the last, rather than just been limited to present and past. This helps model to better understand the input data. Use of Bidirectional further increases the demand of storage as well as processing power since in bidirectional RNN we are simultaneously implementing Two RNN cell to see process data from left to right as well as from right to left. Right to left processing of data helps to RNN data to look into the future.

Further, selection of dataset is also an important part in developing machine learning model. Because if our input data is not properly processed and suitable to our model, our model would not be able to better converge on dataset and as a result will not be able to produce result.

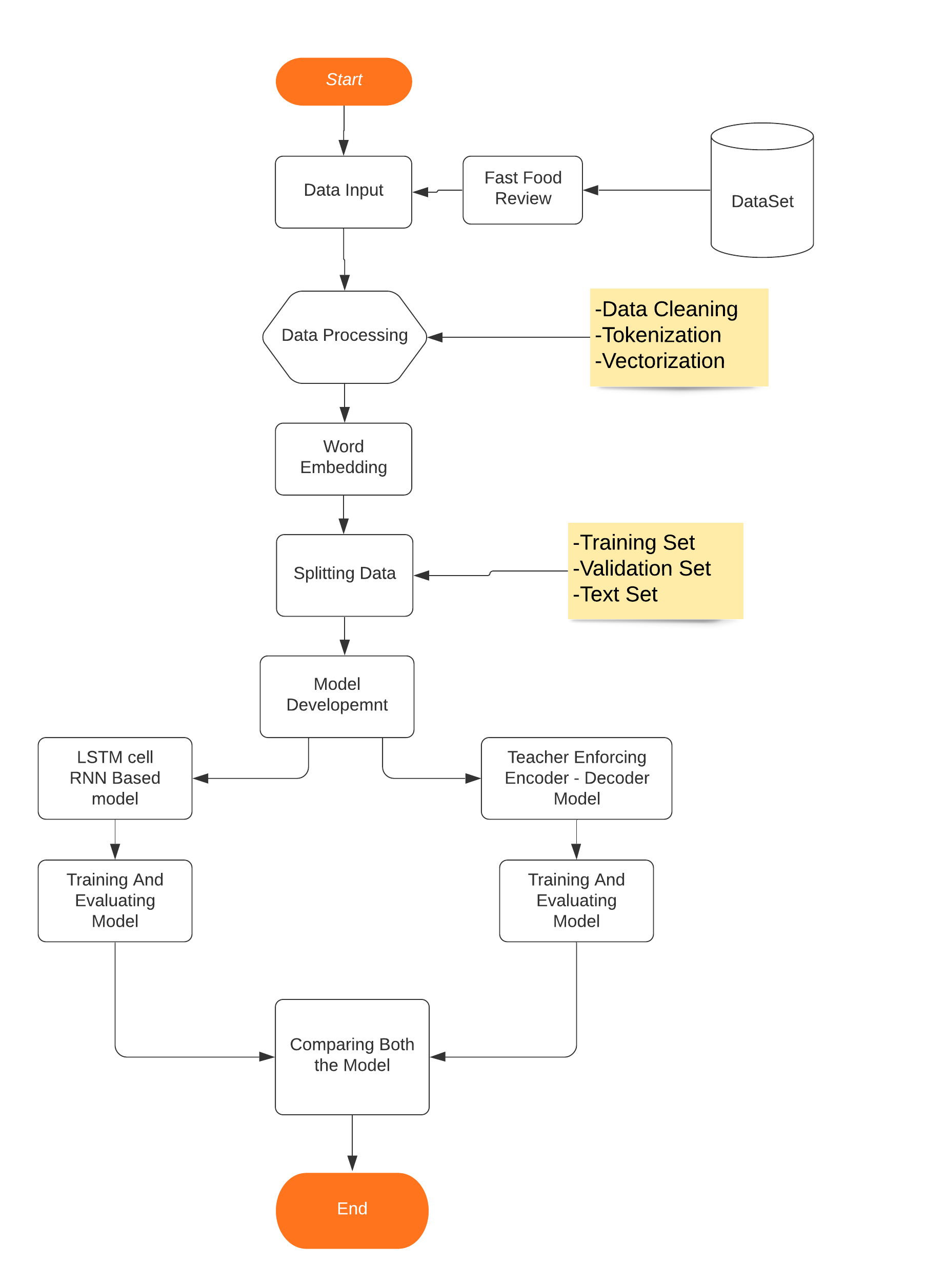
One of the best datasets to use of NLP is Gigaword Dataset. But since it is a licensed dataset, we would be using Amazon Fine Food Review Dataset for our model.

Language Used: Python

Library Used:

* TensorFlow Keras Api
* NumPy and Pandas
* ConcetnetNumberbatch
* Rebux to clean data by removing punctuation.
* NLTK corpus to input Stop words data set. This helps to removing stop wards such as helping verbs (is, are etc.).

**4. Work Design**



**5. Deep Learning:**

In this project, we'll utilize Deep Learning to create an abstractive summarizer based on a food review dataset. So, before we get started on the model, let's have a look at what deep learning is. Figure 3 depicts the basic structures of a neural network, including its hidden layer. Summarizers, for example, use Neural Networks (NN) for Natural Language Processing (NLP). Almost any machine learning classification problem may be solved with neural networks. The number of hidden layers to be employed, the number of hidden units to be included in each layer, the activation function for each node, the data error threshold, and the kind of interconnections are all important criteria to consider when establishing the architecture of a neural network (NN). Deep learning employs deep neural networks to learn accurate representations of incoming data that can then be applied specific tasks.

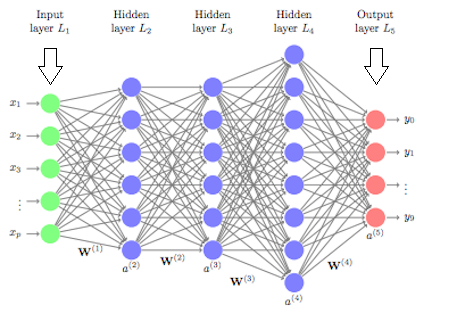


Fig 3 Deep Learning Structure

**Recurrent Neural Network (RNN)**

In the 1980s, recurrent neural networks were developed. They're especially useful with sequential data because each neuron or unit can store information about the prior input in its internal memory. This is fantastic because "I went shopping" and "I went shopping" are vastly different in terms of terminology. Model is able to comprehend the phrase better due to internal memory. While reading in input, an RNN has loops that allow information to be transmitted across neurons. In Figure 4, xt represents some input, A represents a component of the RNN, and ht represents the output. Basically, you may feed words from a sentence or even characters from a string as xt, and the RNN will generate an ht. The purpose is to compare your desired data to ht as an output. After that, we'll calculate our mistake rate. With the error rate in hand, we can apply a technique called Back Propagation Through Time after comparing our output to your desired data (BPTT). Backchecking the network with BPTT and adjusting the weights based on the error rate. This modifies the network and allows it to learn more effectively. RNNs can theoretically handle context from the start of a sentence, allowing for more accurate word predictions at the end of a sentence. In practice, this isn't always the case with vanilla RNNs. This is one of the main reasons why RNNs fell out of favor for a while until some impressive results were obtained by incorporating a Long Short-Term Memory (LSTM) unit into the Neural Network.

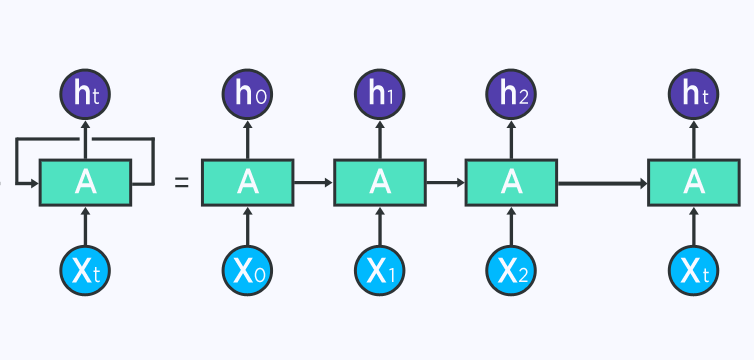


Fig4 RNN

Long Short-Term Memory (LSTM)

The LSTM is an RNN architecture capable of remembering previous contextual values. During the training of the model, these stored values do not change. LSTM is made up of four parts: LSTM Units, LSTM Blocks, LSTM Gates, and LSTM Recurrent Components. The LSTM Unit can store values for a long or short period of time. LSTM has no activation functions for their recurrent components. Since there are no activation function the values of units does not change for some period until the context is changed. A LSTM Block contains such many units. LSTMs are considered as deep neural networks. These LSTMs are implemented in parallel systems. The information flow is controlled by four gates in LSTM blocks. These gates are implemented using logistic functions to compute a value between 0 and 1. Multiplication of values with these logistic functions is done to enable or deny information flow into or out of the memory. To control the flow of new values into memory, input gate plays key role. The extent to which a value remains in memory is controlled by forget gate. Output gate controls the extent to which the value in memory is used to compute the output activation of the block. When new value which is worth remembering is available then we can forget the old value. This represents the combining effect of input and forget gate of LSTM

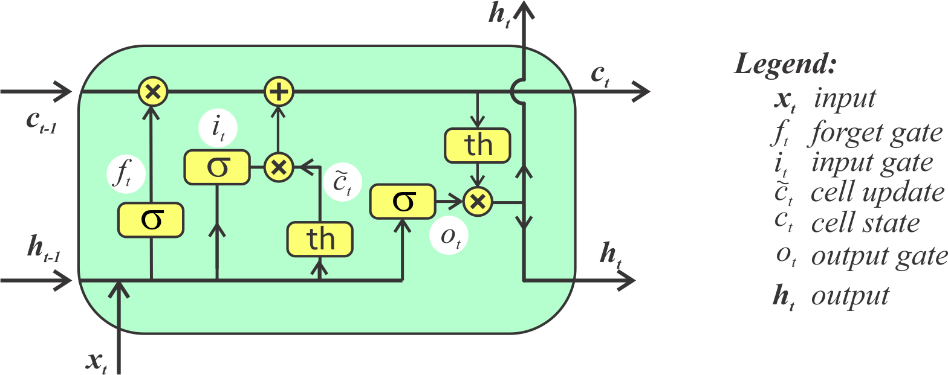


Fig 5 LSTM Architecture

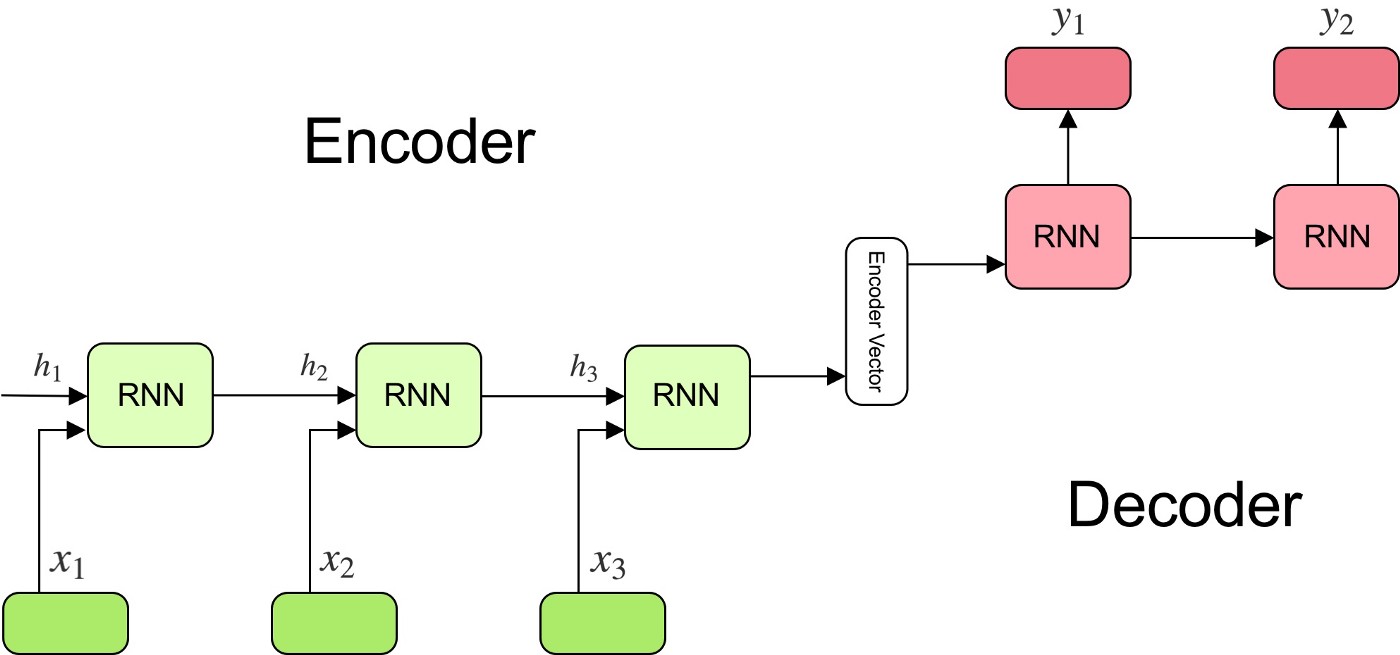
Encoders and Decoders

The Encoder-Decoder LSTM is one technique to seq2seq prediction problems that has proven to be quite effective.

This architecture is made up of two models: one that reads the input sequence and encodes it into a fixed-length vector, and another that decodes the fixed-length vector and outputs the predicted sequence. The architecture is known as Encoder-Decoder LSTM built specifically for seq2seq situations since the models are used together. The Encoder-Decoder LSTM was created for natural language processing problems where it displayed state-of-the-art performance, particularly in statistical machine translation (text translation). In one of the first applications of the architecture, English-to-French translation, the internal representation of the encoded English phrases was visualized. The plots revealed a qualitatively meaningful learned structure of the phrases harnessed for the translation task. Further, the model was shown to be effective even on very long input sequences. This approach has also been used with image inputs where a Convolutional Neural Network is used as a feature extractor on input images, which is then read by a decoder LSTM.

We know that for tasks like object or speech recognition, all of the information needed to complete the job is encoded in the data (because humans can perform these tasks from the raw data). Natural language processing systems, on the other hand, have typically treated words as discrete atomic symbols, so 'cat' may be represented as Id537 and 'dog' as Id143. These encodings are arbitrary, and they give the system no helpful information about the possible relationships between the individual symbols.

We use Concept Net Number batch word embedding in our project. The Concept Net Number batch contains state-of-the-art semantic vectors (also known as word embeddings) that can be used to represent word meanings directly or as a starting point for future machine learning. The Concept Net Number batch is part of the open data project Concept Net. Word embeddings are one of the many ways to compute with word meanings that Concept Net offers. A snapshot of just the word embeddings is called a number batch.



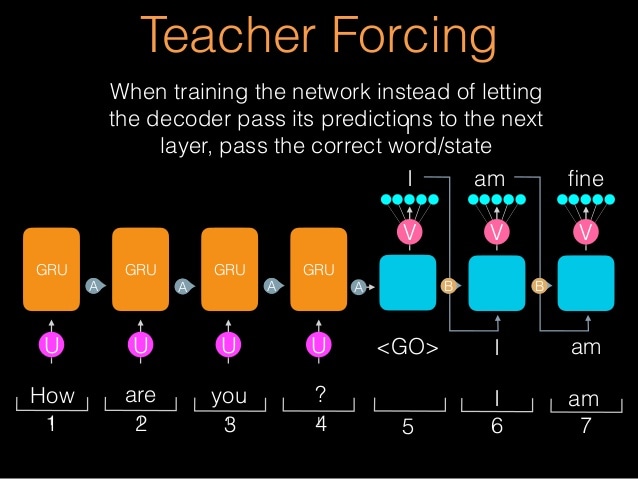
Teacher Forcing Encoder – Decoder:

Teacher forcing is a technique for rapidly and efficiently training recurrent neural network models with ground truth from a previous time step as input.

There are sequence prediction models that use the outcome from the previous time step y(t-1) as input for the current time step X(t).

Language models that produce one word at a time and utilise the output word as input to generate the next word in the sequence commonly use this type of model.

Instead of using the output generated by the network, teacher forcing uses the actual or predicted output from the training dataset at the current time step y(t) as input in the next time step X(t+1).



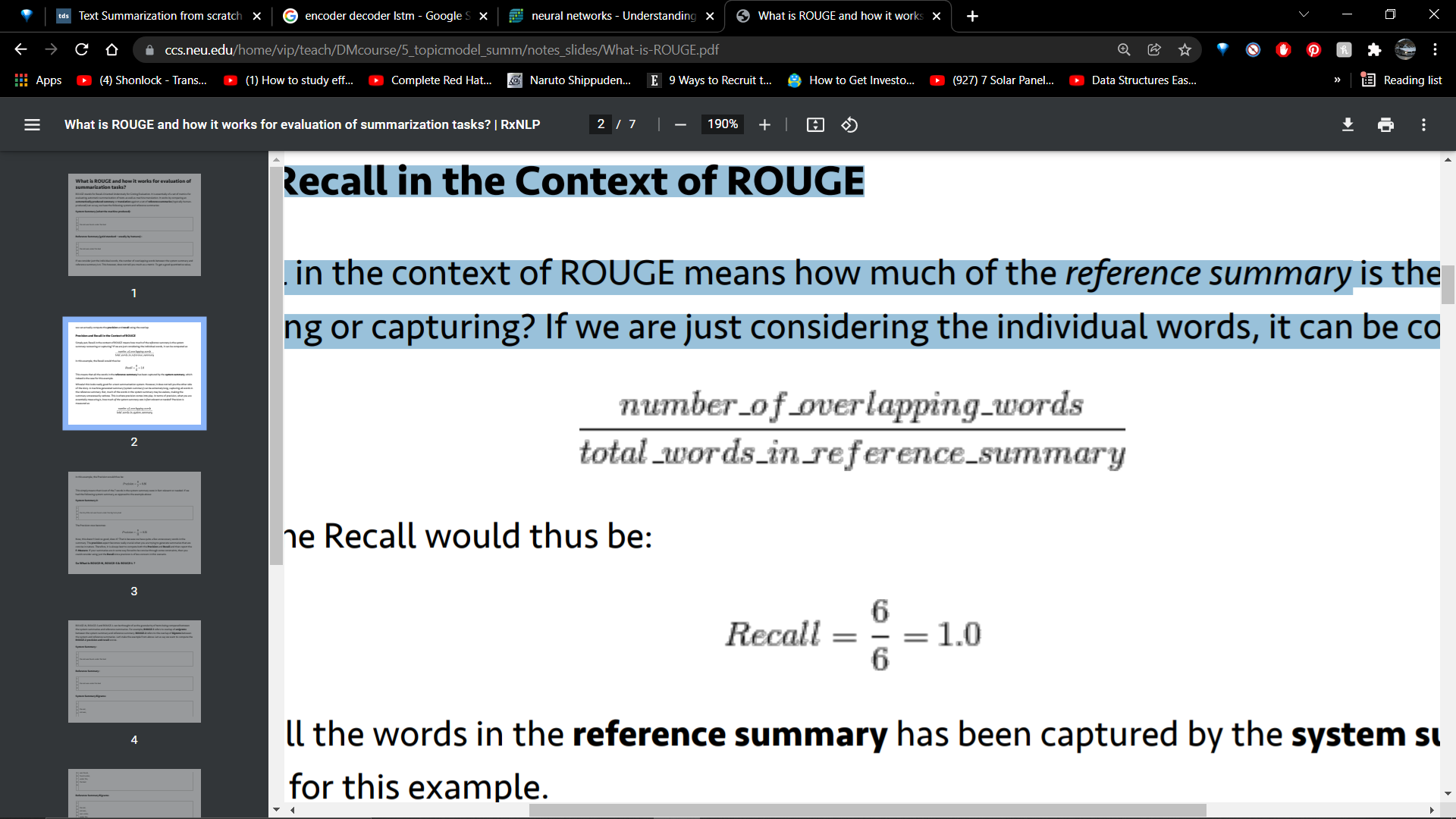
Evaluation Method:

ROUGE MODEL:

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It consists mostly of a set of measures for assessing automatic text summarization and machine translation. It works by comparing a summary or translation generated automatically with a set of reference summaries (typically human produced)

Precision and Recall in the Context of ROUGE

Simply put, recall in the context of ROUGE means how much of the reference summary is the system summary recovering or capturing. If we are just considering the individual words, it can be computed as:

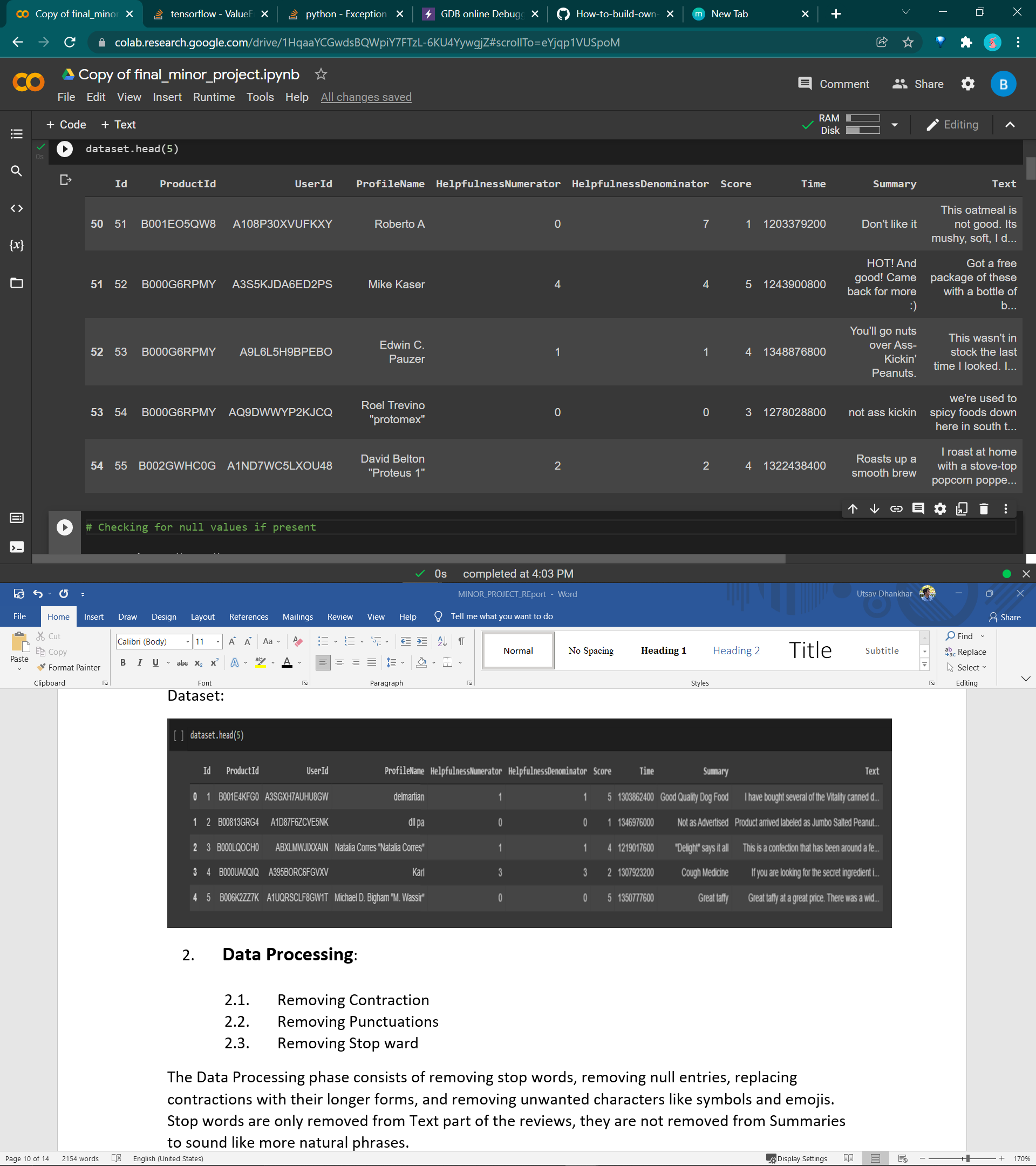


**6. Implementation:**

Using Google Colab

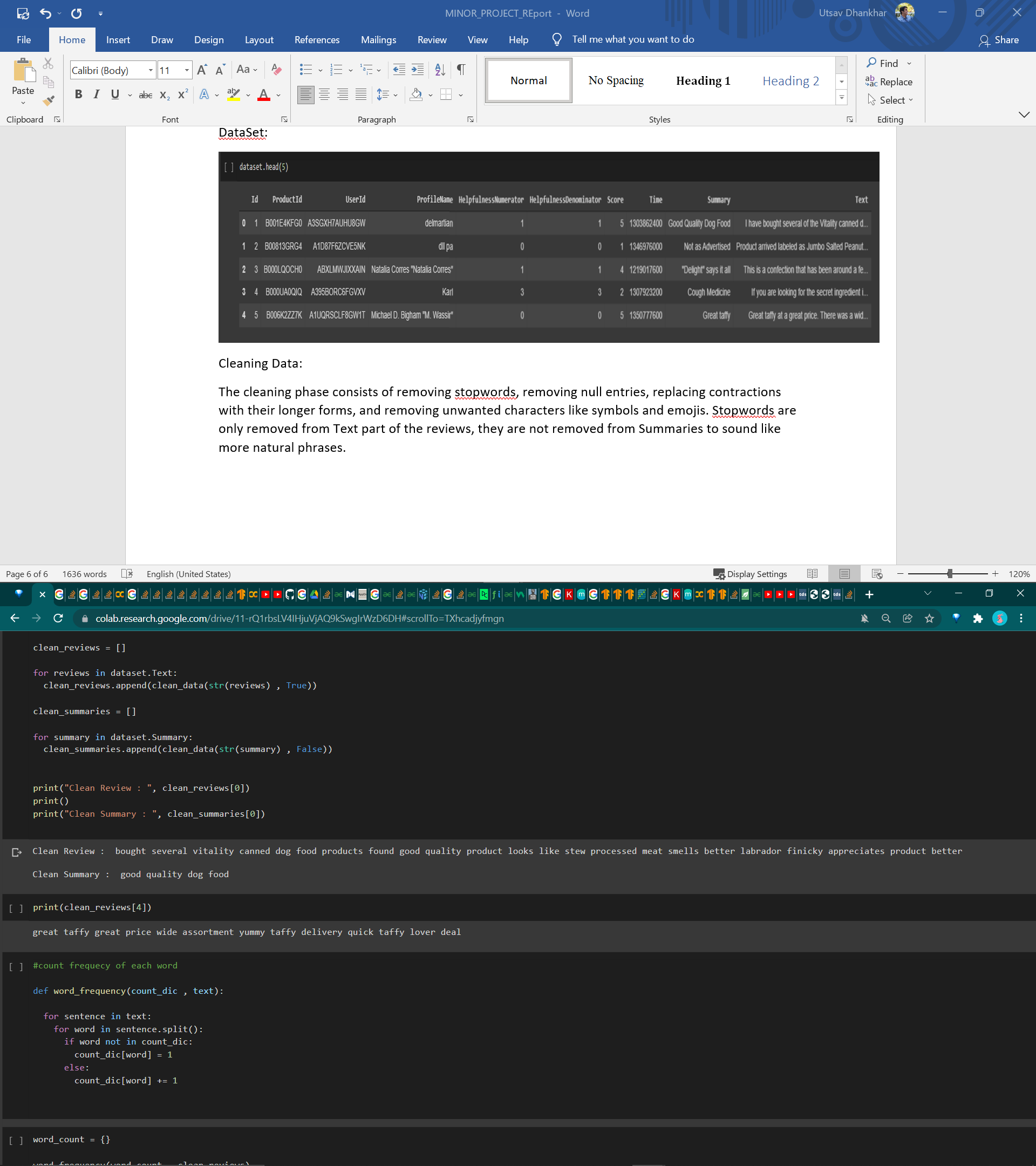
1. Importing Dataset to Google Colab

Dataset:



1. **Data Processing**:
   1. Removing Contraction
   2. Removing Punctuations
   3. Removing Stop ward

The Data Processing phase consists of removing stop words, removing null entries, replacing contractions with their longer forms, and removing unwanted characters like symbols and emojis. Stop words are only removed from Text part of the reviews, they are not removed from Summaries to sound like more natural phrases.



1. **Word Embedding**
   1. Tokenization
   2. Creating Vocabulary
   3. Creating Dictionary
   4. Word Vectorization

Once the data is cleaned, it is tokenized and converted into vectors so that it can be processed by the model. We then load our pre-trained Concept Net Number batch word embedding. We find the number of words missing in the embedding by comparing all the tokens from our reviews data to loaded embedding. We also add token <UNK> token for missing word words and we add in the end of line <EOS> token to indicate that it’s the end of line. Finally, we sort the summary and text by the length of text, i.e., from shortest to longest.

1. **Padding and Splitting Dataset:**

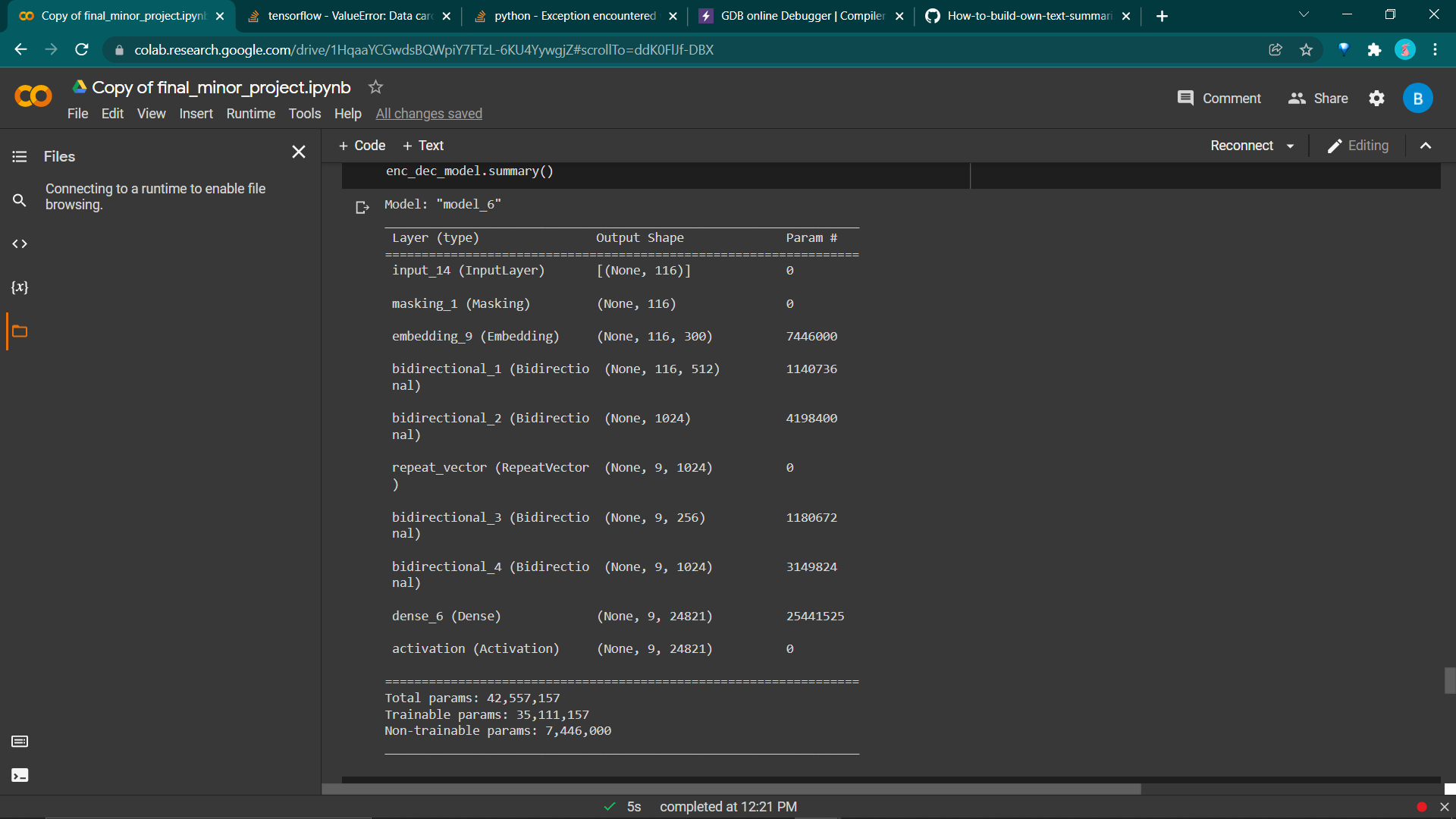
The RNN and Encoder-Decoder model requires input of same length. Since sentences are usually not of same length, padding is done on review and the summary dataset to make them of same size. For Padding we use <PAD> token and this token are appended at the end of the sentences

The Dataset is split in Training Dataset, Validation Dataset and training Dataset proportion of .8, .1, .1 respectively. Validation Dataset is used to Tune Hyperparameter for model to better adjust to the dataset.

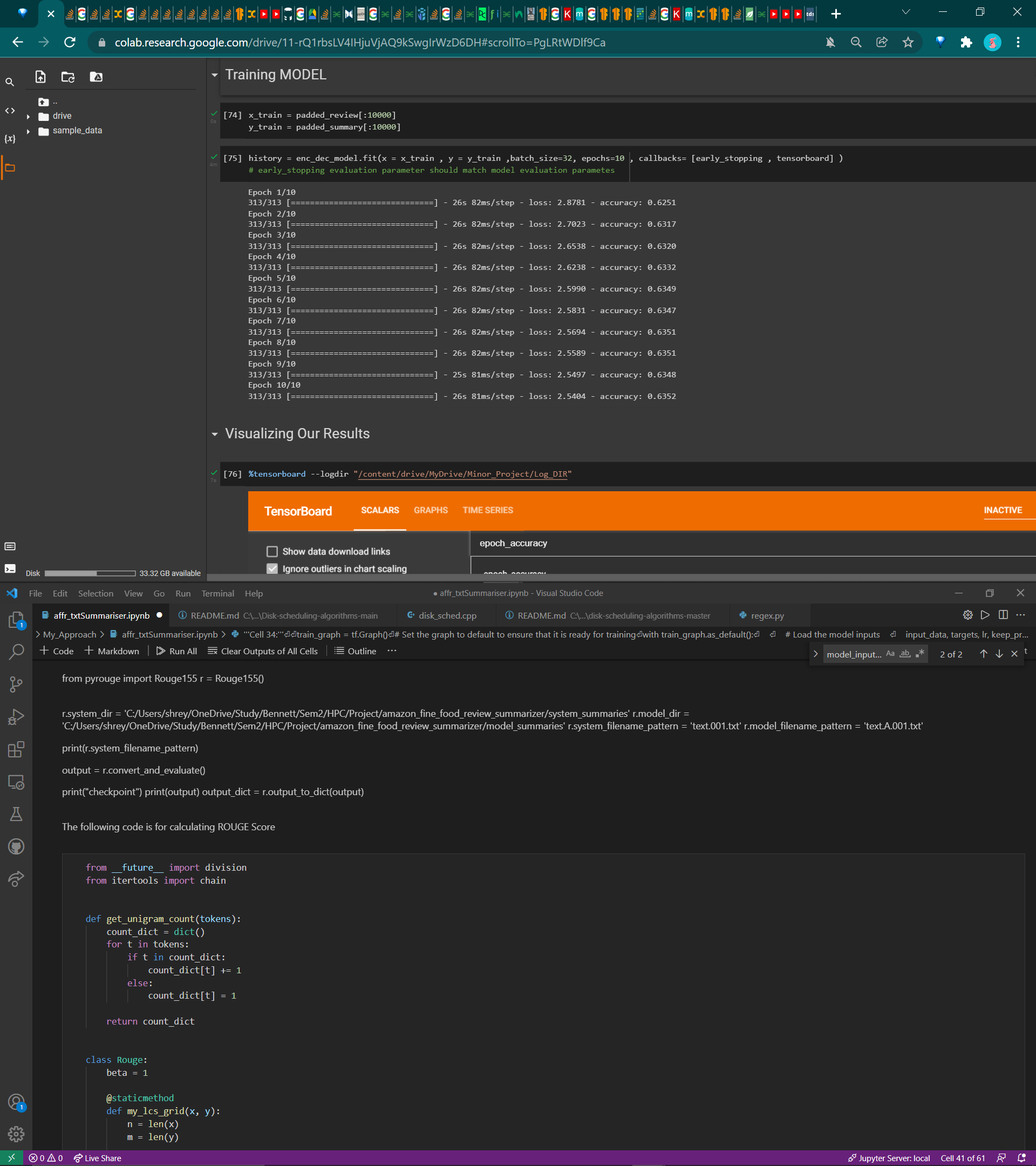
1. **Masking:**

Since our dataset consists of word, and machine can only understand number (binary form) we need to convert our data into numbers. Now it is not necessary that every input data or every output data is of same length so we apply padding. Now to explain model not to process padding as input we define a masking layer. This tell our Encoder-Decoder Model to not process pass as input and hence <PAD> tokens will be ignored.

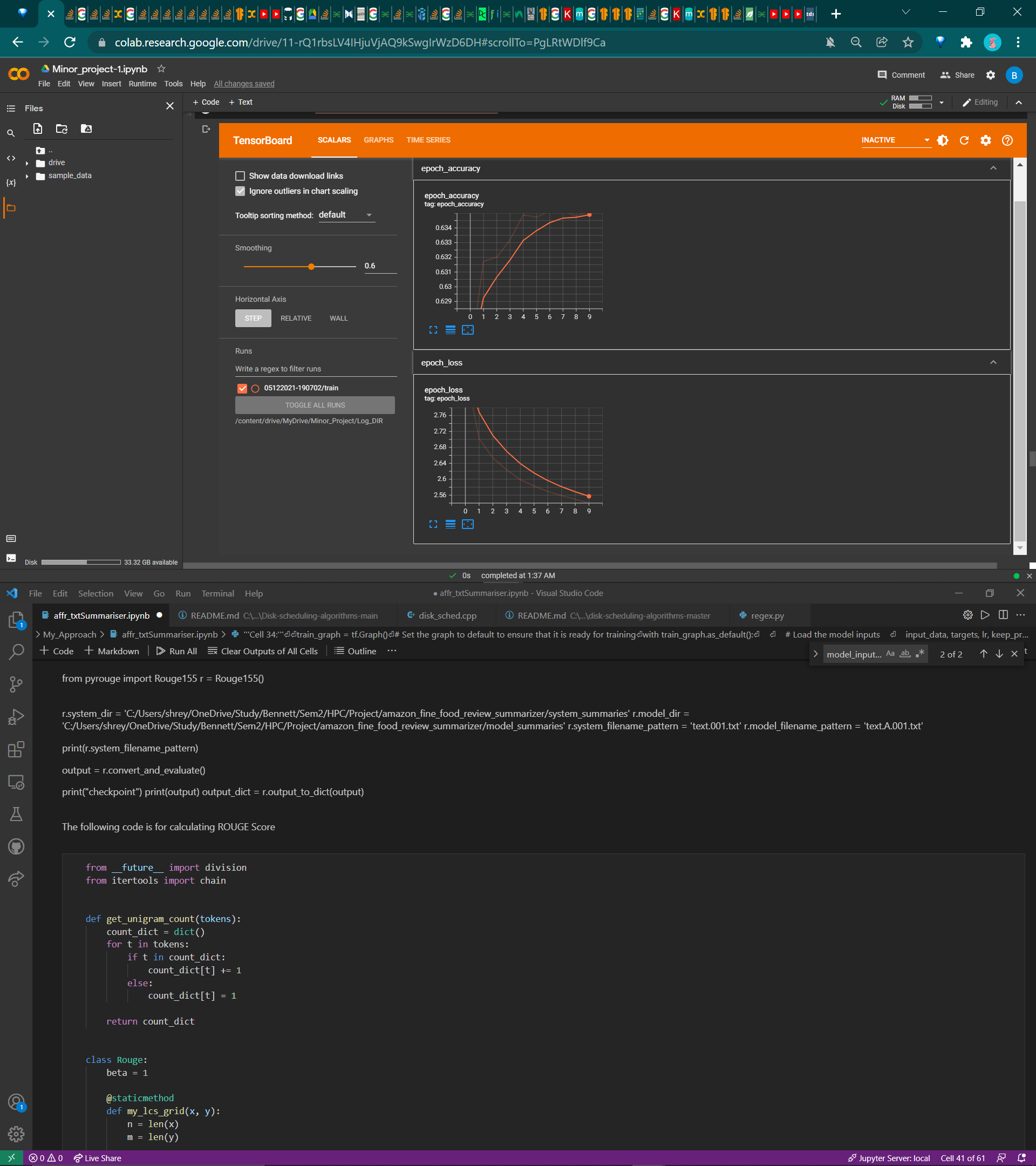
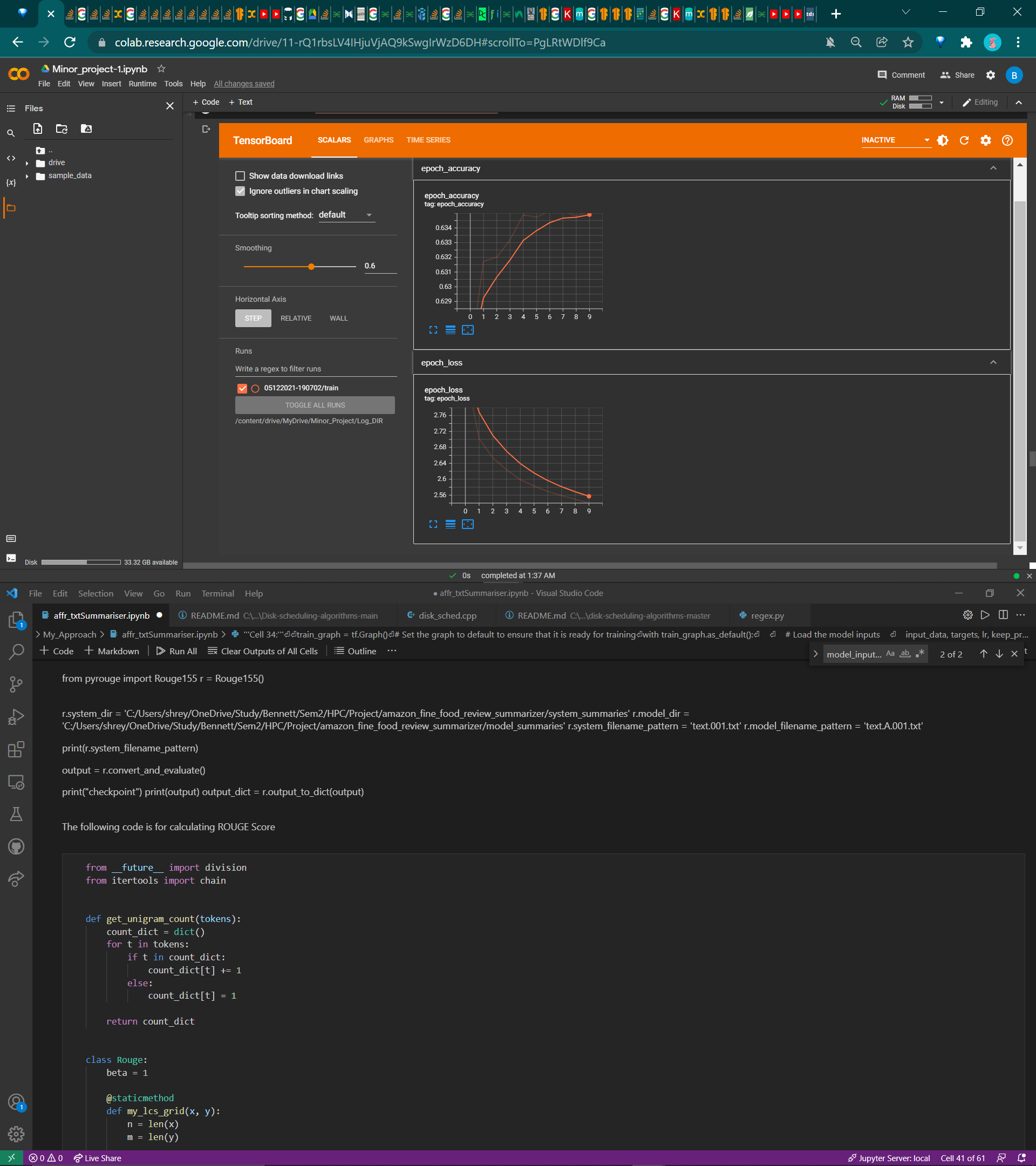
1. **Developing MODEL:**
   1. Simple LSTM Based Encoder-Decoder architecture Design:



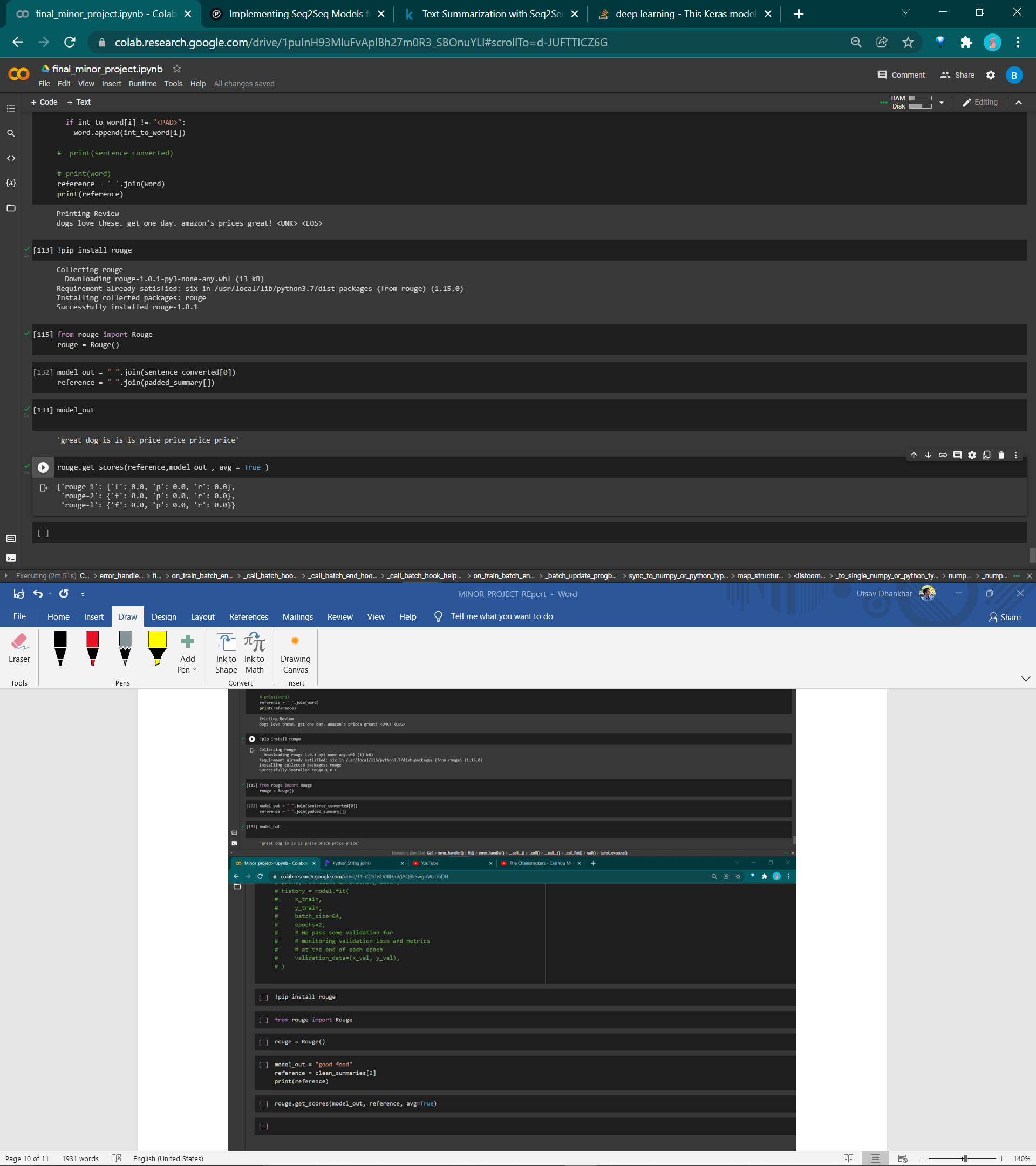
Model Training Progress:

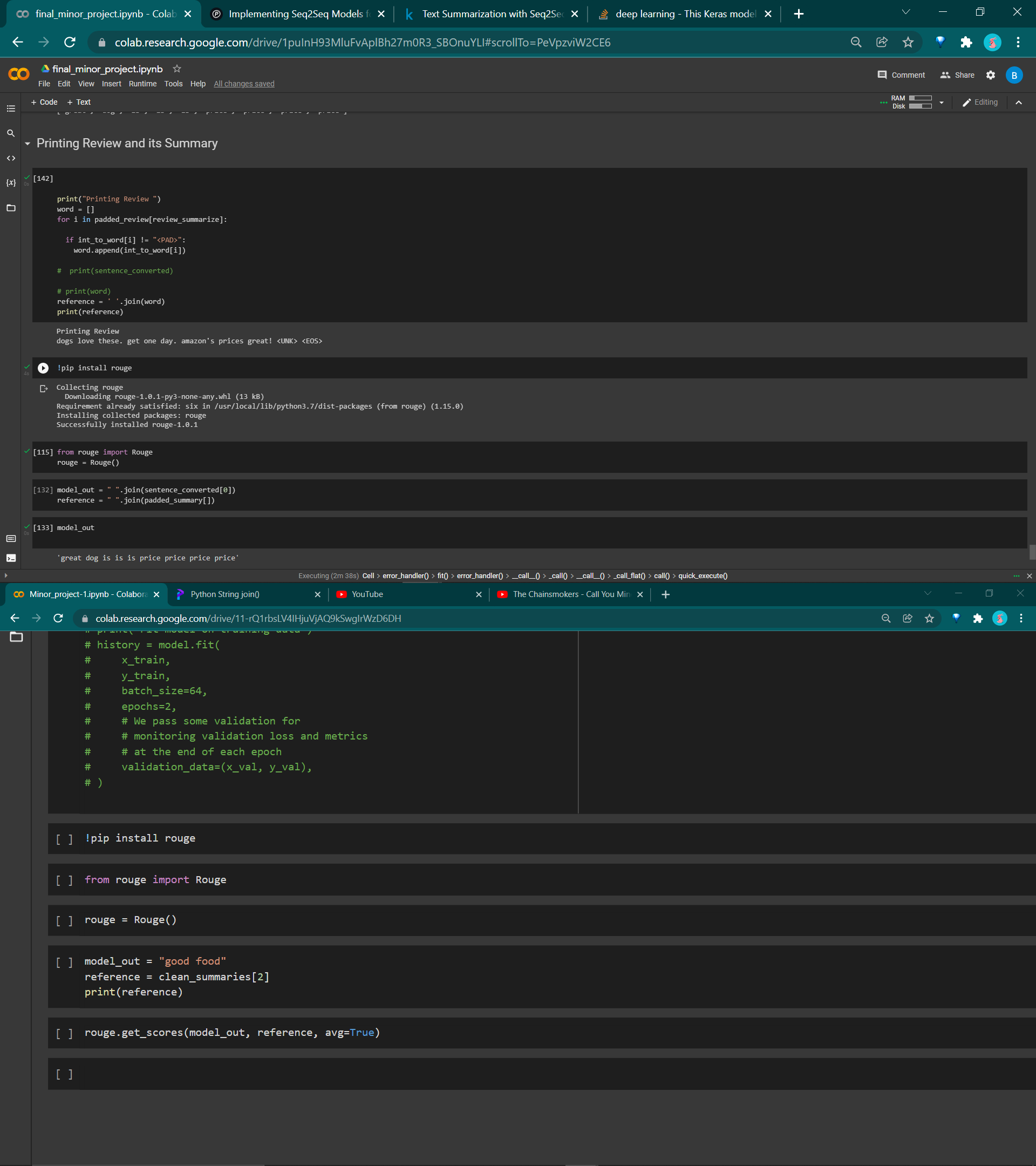


Graph: Accuracy/Loss of our MODEL:

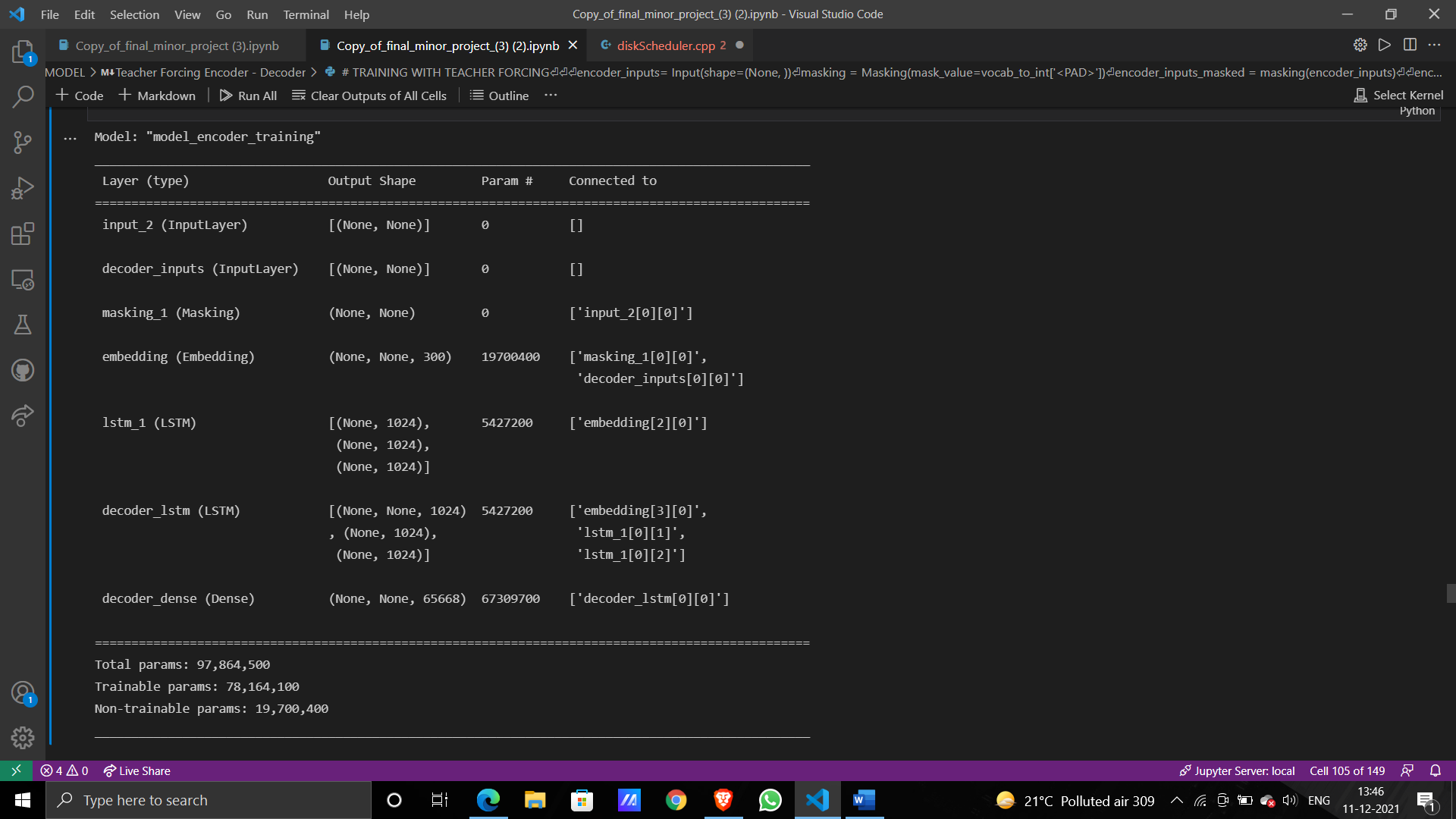
 

Predicted Output:

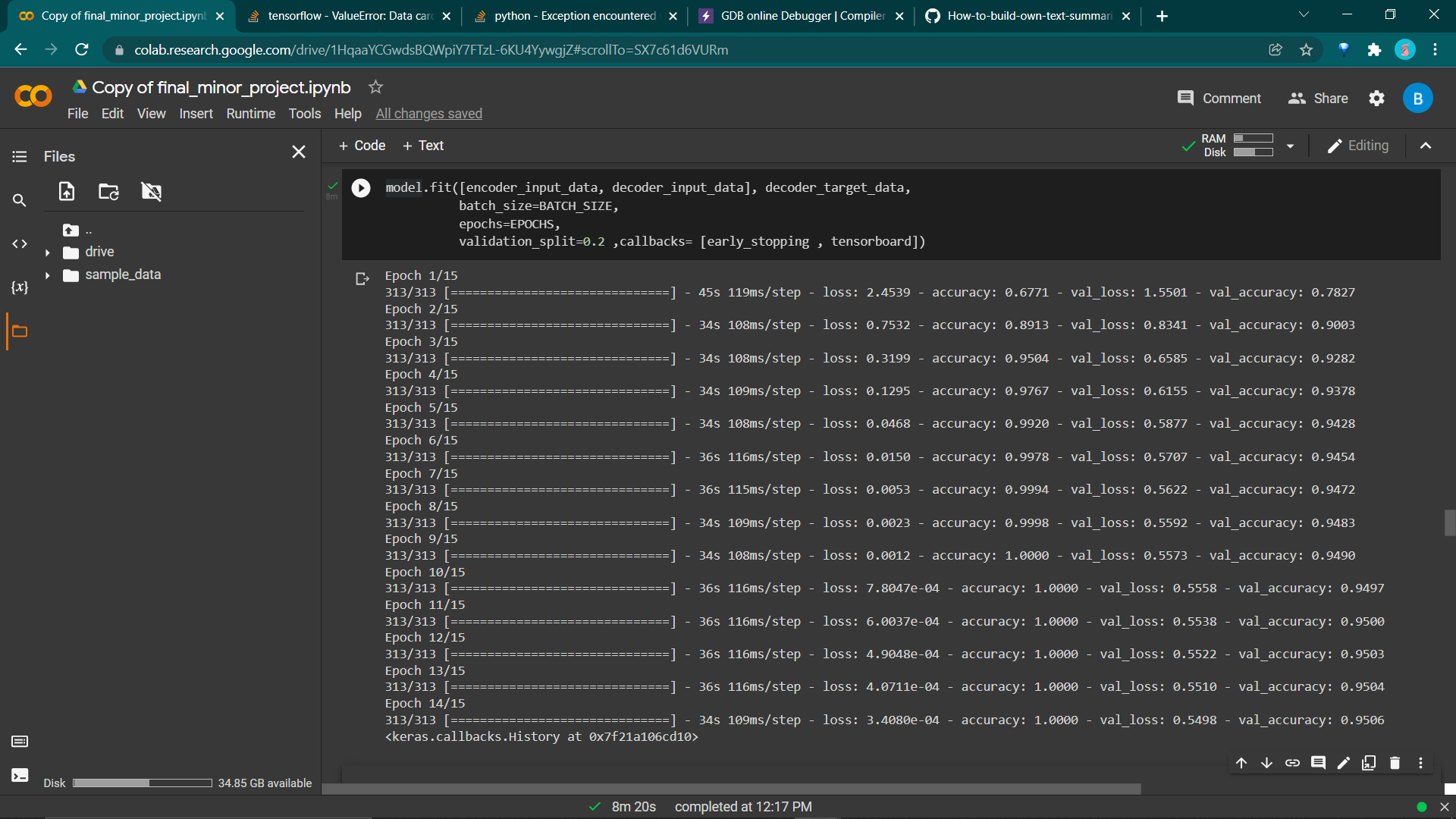




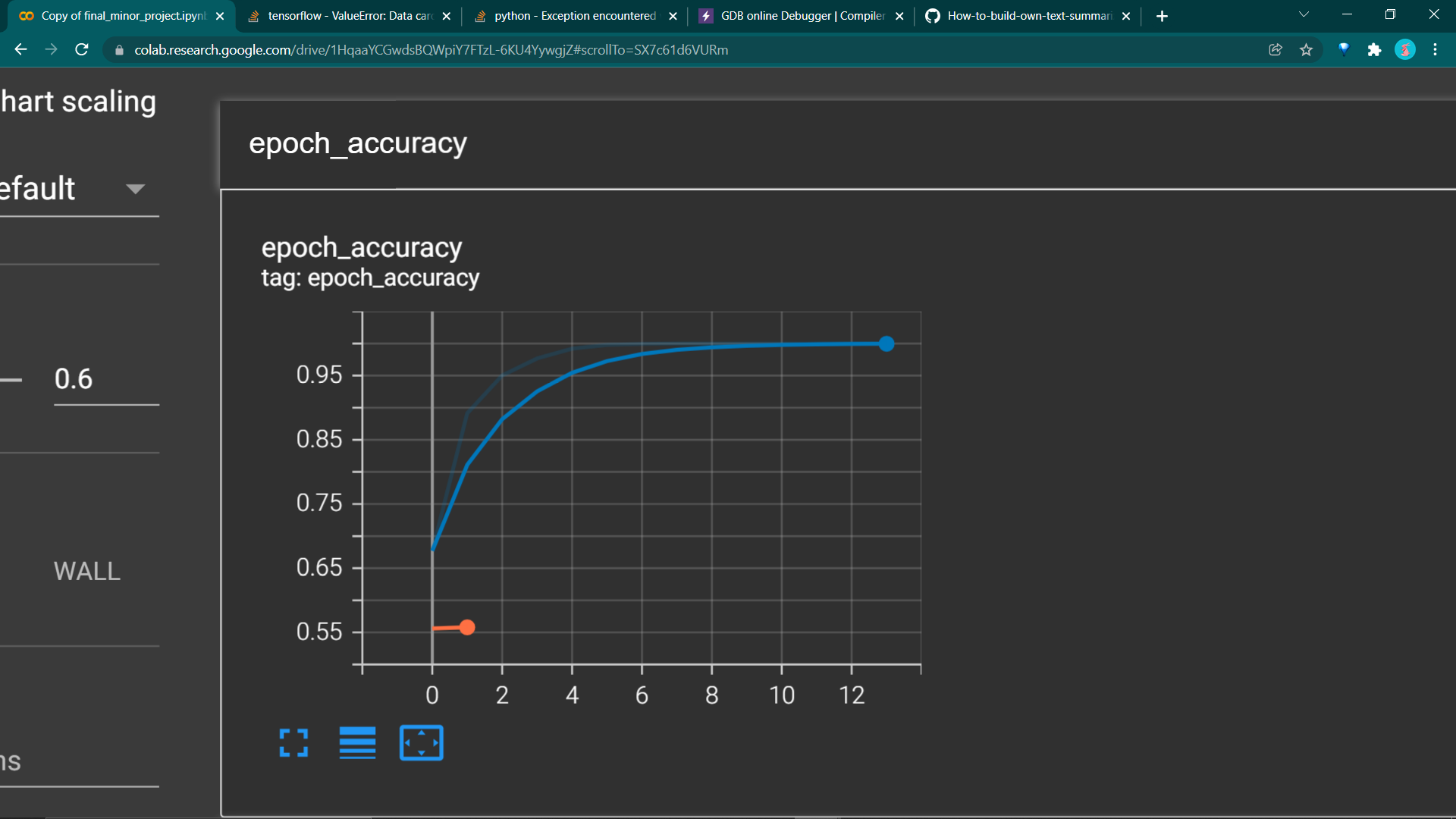
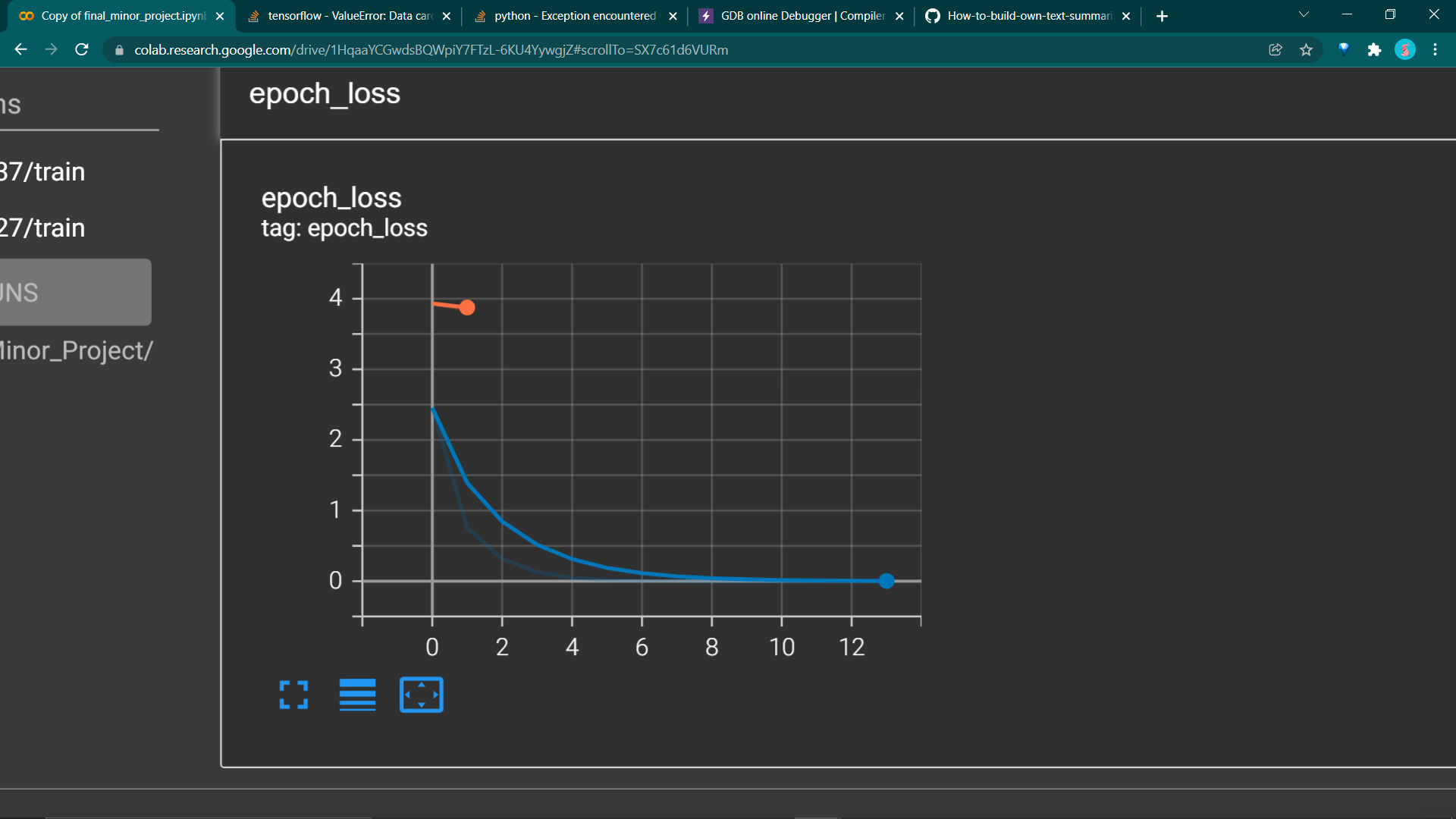
* 1. Using Teacher Forcing Encoder Decoder Method:



Model Training Progress:



Graph: Accuracy/Loss of our MODEL:

1. **Conclusion**

Although Teacher Forcing Encoder – Decoder Model converge better on Training and Validation set, when compared to Simple Encoder – Decoder Model, but when tested both on the Test Set, they seem to produce similar results. This is because of although Teacher forcing help encoder-decoder model to converge better on training data set by better setting weights, there is not much difference in design and processing method of both the model. Hence, we achieve similar results in our output.

We have successfully implemented state-of-the-art model for abstractive sentence summarization to a recurrent neural network architecture. The model is trained on the Amazon-fine-food-review corpus to generate summaries of review.

There are few limitations of the model which can be improved in further work.

* First limitation is that it sometimes generates repeated words in the summary.
* The other problem is, it takes too much time to generate a summary if the input text size is large enough.
* At times, if the length of sentence is greater than certain threshold, the LSTM cell starts to forget the initial input of the encoder.
* At times, it gives summary that is completely out of context.
* For better analysis, i.e., for the model to converge better on dataset we need a better Dataset.
* Since we are limited by the processing power of our machine, certainly a more powerful system would be better to work on large quantity of dataset. This will help model to learn better by performing more epochs on dataset.

**Further Study:**

With further research in topics such as Attention layer, or BERT method, the unnecessary reptation of word in summary can be resolved. With the help of attention layer, the model is able to handle large quantity of data set and better converge on dataset.

**9. References:**

1). Lopyrev K Generating news headlines with recurrent neural networks[J]. arXiv preprint arXiv:1512.01712

2). Nallapati R, Zhou B, Gulcehre C et al (2016) Abstractive text summarization using sequence-to-sequence rnns and beyond[J]. arXiv preprint arXiv:1602.06023

3). Nallapati R, Zhai F, Zhou B (2017) Summarunner: a recurrent neural network based sequence model for extractive summarization of documents. In: Proc. Thirty-First AAAI Conference on Artificial Intelligence, pp 3075–3081

4). S. Syed, *Abstractive Summarization of Social Media Posts: A case Study using Deep Learning*, Bauhaus University, Weimar, Germany, 2017, Master’s thesis.

5). C. Sunitha, A. Jaya, and A. Ganesh, “A study on abstractive summarization techniques in Indian languages,” *Procedia Computer Science*, vol. 87, pp. 25–31, 2016.