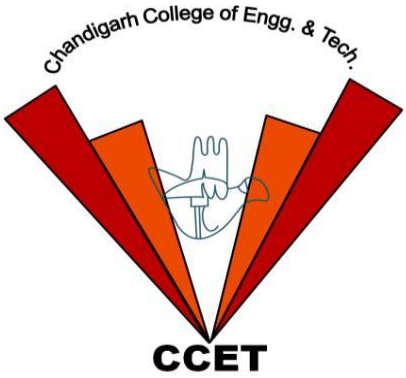
Lab Project Report

On

Text Summarizer using Attentive RNNs

Under the supervision of



CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY
(DEGREE WING)
Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University, Chandigarh
Sector-26, Chandigarh. PIN-160019

AUGUST-DECEMBER, 2022

Submitted by
ABHISHEK PANT (CO18305)
SHUVAM ROY (CO18349)
CSE 7th SEM

Table of Content

ABSTRACT

In this project report, we have implemented a Text Summarizer using Attentive RNNs for a dataset and also we have implemented on our input too. So basically, Text Summarizer or Text abstraction is a way to reduce long text into smaller fragments. The intent is to create a consistent and fluent summary that includes only the main points outlined in the document. Text summarization is a common issue in Neural Networks and natural language processing (NLP) & also Machine Learning. As Neural Networks sequence to sequence models have supplied a possible revolutionary method for abstractive textual content summarization (that means they may be now no longer confined to virtually choosing and rearranging passages from the unique textual content). However, those models have their limitations and shortcomings i.e. they may be prone to reproduce real information inaccurately, and that they have a tendency to copy themselves . In this project we've used well known Long Short-Term Memory(LSTM) sequence to sequence attentional version. This approach makes use of a nearby interest version for producing every phase of the precise condition at the enter sentence. While the version is structurally simple, it may effortlessly learn cease-to-cease and scales to a massive quantity of schooling data. By using Tensorflow neural network library and with the help of Amazon Fine Food Reviews as a dataset we have implemented the same. We have also implemented the same for our input. At the end we examine the reconstructed paragraph the use of standard metrics, by which we can display that neural networks models can encode texts in a manner that hold syntactic, semantic, and discourse coherence.

1. INTRODUCTION

With the dramatic increase of the Internet, humans are crushed via way of means of the splendid quantity of on-line data and files. This increasing availability of files has demanded exhaustive studies in automated textual content summarization. According to a it's described as “a text that is produced from one or more texts, that conveys valuable information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that”.
Generating a condensed model of a passage whilst retaining its which means is referred to as textual content summarization. Tackling this assignment is a important step in the direction of important language understanding. Summarization structures may be extensively categorised into categories. Extractive fashions generate summaries via way of means of cropping essential segments from the authentic textual content and placing them collectively to shape a coherent abstract or summary. Abstractive fashions generate summaries from scratch without being confined to reuse of terms from the authentic textual content.
Automatic textual content summarization may be very challenging, due to the fact while we as human beings summarize a chunk of textual content, we generally examine it totally to expand our understanding, after which write a precis highlighting its most important points. Since computer systems lack human information and language capability, it makes automated textual content summarization a totally tough and non-trivial assignment.
Summaries also can be of types: typical or question-targeted. Topic-targeted or user-targeted summaries are the opposite names for question-targeted summaries. Such a precis consists of the question associated content material while a popular experience of the data gift withinside the record is supplied in a typical precis.
Summarization assignment may be both supervised or unsupervised. Training information is wanted in a supervised device for choosing essential content material from the files. Large

quantity of labelled or annotated information is wanted for studying techniques. These kinds are addressed at sentence step as elegance category hassle where sentences belonging to the summary are described as high-quality samples and sentences which are no longer unique withinside the summary are named as bad input. In sentence abstraction, general based techniques are assigned along with SVMs and neural networks. On the opposite hand, unsupervised structures do now no longer require any schooling information. They generate the summary via way of means of gaining access to most effective the goal files. Thus, they're appropriate for any newly determined information with none superior modifications. Such structures practice heuristic guidelines to extract fantastically applicable sentences and generate a precis. The approach hired in unsupervised device is clustering.

Based at the fashion of output, there are styles of summaries: indicative and informative summaries. Indicative summaries inform what the record is approximately. They deliver data approximately the subject of the record. Informative summaries, whilst overlaying the topics, deliver the complete data in elaborated shape.

Based on language, there are 3 styles of summaries: multi-lingual, mono-lingual and cross-lingual summaries. When language of supply and goal record is same, it is a mono-lingual summarization device. When supply record is in lots of languages like English, Hindi, Punjabi and precis is likewise generated in those languages, then it's miles termed as a multi-lingual summarization device. If supply record is in English and the precis generated is in Hindi or some other language apart from English, then it's miles referred to as a cross-lingual summarization device.

Before we pass directly to in addition matters we should understand the simple concepts, which are, Deep Learning, Recurrent Neural Network(RNN), Long Short Term Memory(LSTM) Cells, Encoders-Decoders, Attention Mechanism and Word Embedding.

- **Deep Learning (In context to Neural Networks)**

In this project we're going to use the idea of Deep Learning for Text summarizer primarily based on food review dataset. So before developing any model for the same, we shall acknowledge the concept of deep learning. The basic structures of neural network with its hidden layer. Neural networks are capable enough to solve any machine learning problem. Parameters required in defining the architecture of neural network (NN) are number of hidden layers to be used, number of hidden units to be present in each layer, activation function for each node, error threshold, interconnections type , etc. neural networks can recognize very complex features of data without any proper involvement of manual participation as opposed to the machine learning systems. Deep learning uses deep neural networks to study precise representations of the data, which can then be used to perform specific tasks.

- **Recurrent Neural Networks (RNNs)**

In Recurrent Neural Networks in short RNNs, the latest output is purely dependent on past output. Because of this property RNNs, we will try to abstract out our textual content as more like as human like as possible. RNNs generally use Back Propagation Algorithm, in fact it is applied at every time stamp which is known as Back Propagation through time(BTT).

Further we will be using Sequence to Sequence Model which consists Encoders and Decoders for summarizing our long textual reviews into a proper summary. The Encoders and Decoders are further discussed with LSTMs which is extended part of RNNs.

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

Long Short Term Memory Network in short LSTM is a special kind of Recurrent Neural Network which is capable of learning long term dependency because of solving the issue of short term dependency of a normal Recurrent Neural Network. It is not viable to apprehend the context of input's background while trying to achieve the same with the use of LSTM.

- **Encoders and Decoders**

As we are implementing sequence to sequence models. So one of the approach to sequence-to-sequence prediction which has been effective in a way is Encoders-Decoder LSTM network. This architecture contains two models i.e. First one is for reading the input sequence and reading the same into vector of fixed length and Second one is for decoding the vector and predict the sequence as a result. The use of the models in implementation justifies its name of Encoder-Decoder LSTM which means it is designed particularly for seq2seq issues.

- **Attention Mechanism**

In this we will specifically the role of Attention Mechanism about how it would be implemented. Well now rather visualizing at all the words in the sequence of the source , we should be able to increase the necessity of particular parts of the source sequence which results in the target sequence. So, it is basic idea behind the attention mechanism. It is classified in two different types dependent of the derived context vector in which one is Global Attention where the attention is located on all the source or starting positions. Second one is Local Attention where the attention is located on only few starting or source positions.

- **Word Embedding**

Word Embedding is a learning representation technique for text in which words having same meaning further have similar representation. It is a class of technique where every word is represented as vectors with real value in predefined vector space. Each word is mapped to single vector and values of the vectors are hence learned in a way that resembles a neural networks and as a result the method is frequently lumped into the area of deep structured learning. In this project we are using Concept Net Number batch for word embedding. It contains semantic vectors which are precisely be used for word embedding.

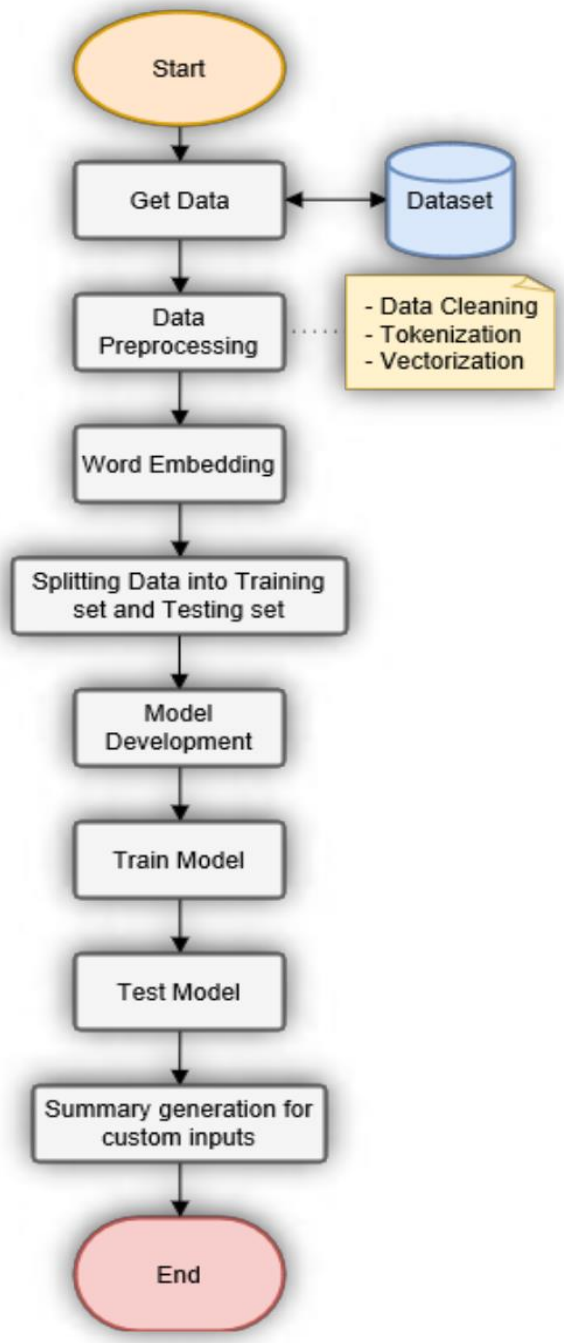
2. Setting up Environment for Text Summarizer

The following components and modules are required to download and installed properly

- Download and Install Anaconda 3
- Make sure Jupyter Notebook is Pre Installed as we are implementing the project in it.
- Download and Install Python v3.5 or Above.
- Download and Install Pandas
- Download and Install Numpy
- Download and Install NLTK Library
- Download and Install Tensorflow (v1.5.0 or less is preferred)
- Download the "Amazon Fine Food Reviews" Dataset from Kaggle.
- Download the "Concept Net Number Batch" *.txt extension from Github repository for word embedding.
- .txt files in which any random text is saved by which we can test our inputs rather than the original dataset

3. Methodology

The approach we have applied, uses machine learning as well as deep learning concepts. So the flowchart explaining our approach is as below:



So in the above flowchart shows the approach for the project. First step is to acquire the data according to our need that how would we train the dataset for the respective models. After setting it up to the requirements we have 10 columns and 50,000+ rows present. The data includes products and user information, ratings, and the plain text review with its summary.

Dataset

After downloading the dataset , we get the CSV file. In order to implement the dataset we first have to import it our code and it is read by the code and then the further steps are proceeded. The preview of data can be viewed as below :

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1.0	1.0	5.0	1.303862e+09	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0.0	0.0	1.0	1.346976e+09	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1.0	1.0	4.0	1.219018e+09	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3.0	3.0	2.0	1.307923e+09	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0.0	0.0	5.0	1.350778e+09	Great taffy	Great taffy at a great price. There was a wid...

Data Cleaning

The cleaning phase consists of removing stop words, removing null entries, replacing contractions with their longer forms, and removing unwanted characters like symbols and emoji's. Stop words are only removed from Text part of the reviews, they are not removed from Summaries to sound like more natural phrases. After processing the data, it will have only two columns i.e. text and summary. The below shows the cleaned data

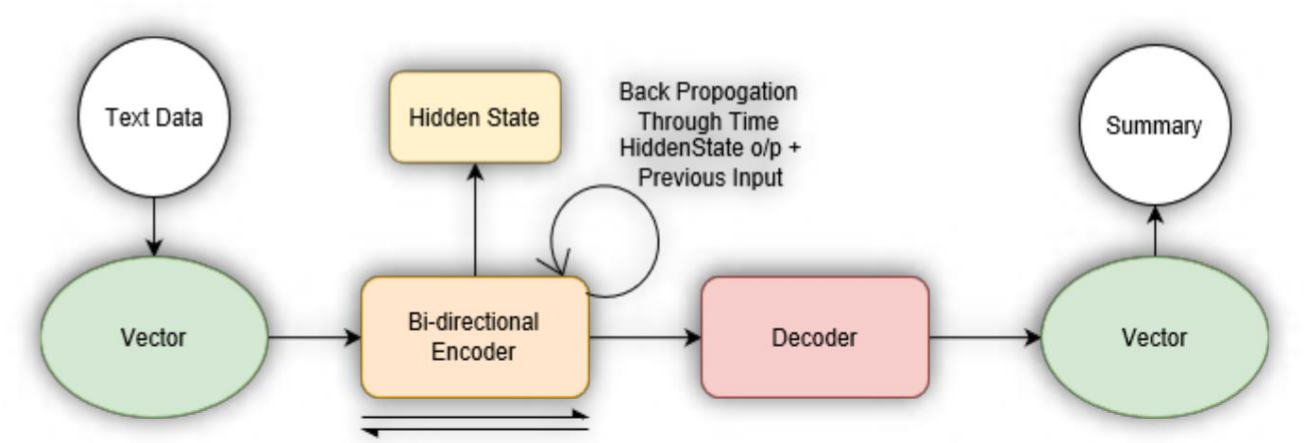
```
# Inspecting some of the reviews
for i in range(5):
    print("Review #",i+1)
    print(reviews.Summary[i])
    print(reviews.Text[i])
    print()
```

```
Review # 1
Good Quality Dog Food
I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The prod
uct 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.

Review # 2
Not as Advertised
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".

Review # 3
"Delight" says it all
This is a confection that has been around a few centuries.  It is a light, pillowy citrus gelatin with nuts - in this
case Filberts. And it is cut into tiny squares and then liberally coated with powdered sugar.  And it is a tiny mouthf
ul of heaven.  Not too chewy, and very flavorful.  I highly recommend this yummy treat.  If you are familiar with the
story of C.S. Lewis' "The Lion, The Witch, and The Wardrobe" - this is the treat that seduces Edmund into selling out
his Brother and Sisters to the Witch.

Review # 4
Cough Medicine
If you are looking for the secret ingredient in Robitussin I believe I have found it.  I got this in addition to the R
oot Beer Extract I ordered (which was good) and made some cherry soda.  The flavor is very medicinal.

Review # 5
Great taffy
Great taffy at a great price.  There was a wide assortment of yummy taffy.  Delivery was very quick.  If your a taffy
lover, this is a deal.
```

Words Embedding

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 ConceptNet Numberbatch 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 <unk> token for those which words are not known and we add <pad> in the end of line to indicate that it's the end of line. Finally, we sort the summary and text by the length of text, i.e. in ascending order.

Now we will design our Model which will be used while training. Since we are using TensorFlow, our major effort goes into building graphs. We will create few place holders to hold our data and other parameters. Then we will create our bi-directional encoder, which will generate its own representation of input data. The Basic Architecture used for this approach can be seen as below:



Applying Attentive RNNs Architecture

We will implement Bahdanau Attention. So,

1. **Attentive Encoder:** The participation of encoder is to make sure that it generates annotation or a context vector for every time stamp of each words in an input sentence of fixed length
2. **Attentive Decoder:** The participation of decoder is to produce the desired or target words by prioritizing on the most latest information or update contained in the source sentence. For this task it makes worth of attention mechanisms. It is implement below.

```
def decoding_layer(dec_embed_input, embeddings, enc_output, enc_state, vocab_size, text_length, summary_length,
                   max_summary_length, rnn_size, vocab_to_int, keep_prob, batch_size, num_layers):
    '''Create the decoding cell and attention for the training and inference decoding layers'''

    for layer in range(num_layers):
        with tf.variable_scope('decoder_{}'.format(layer)):
            lstm = tf.contrib.rnn.LSTMCell(rnn_size,
                                           initializer=tf.random_uniform_initializer(-0.1, 0.1, seed=2))
            dec_cell = tf.contrib.rnn.DropoutWrapper(lstm,
                                                     input_keep_prob = keep_prob)

    output_layer = Dense(vocab_size,
                         kernel_initializer = tf.truncated_normal_initializer(mean = 0.0, stddev=0.1))
```

Now in general Attention Mechanism is implemented as follows:

```
    #Attention Mechanism
    attn_mech = tf.contrib.seq2seq.BahdanauAttention(rnn_size,
                                                     enc_output,
                                                     text_length,
                                                     normalize=False,
                                                     name='BahdanauAttention')

    dec_cell = tf.contrib.seq2seq.DynamicAttentionWrapper(dec_cell,
                                                          attn_mech,
                                                          rnn_size)

    initial_state = tf.contrib.seq2seq.DynamicAttentionWrapperState(enc_state[0],
                                                                    _zero_state_tensors(rnn_size,
                                                                                        batch_size,
                                                                                        tf.float32))
    with tf.variable_scope("decode"):
        training_logits = training_decoding_layer(dec_embed_input,
                                                  summary_length,
                                                  dec_cell,
                                                  initial_state,
                                                  output_layer,
                                                  vocab_size,
                                                  max_summary_length)
    with tf.variable_scope("decode", reuse=True):
        inference_logits = inference_decoding_layer(embeddings,
                                                    vocab_to_int['<GO>'],
                                                    vocab_to_int['<EOS>'],
                                                    dec_cell,
                                                    initial_state,
                                                    output_layer,
                                                    max_summary_length,
                                                    batch_size)

    return training_logits, inference_logits
```

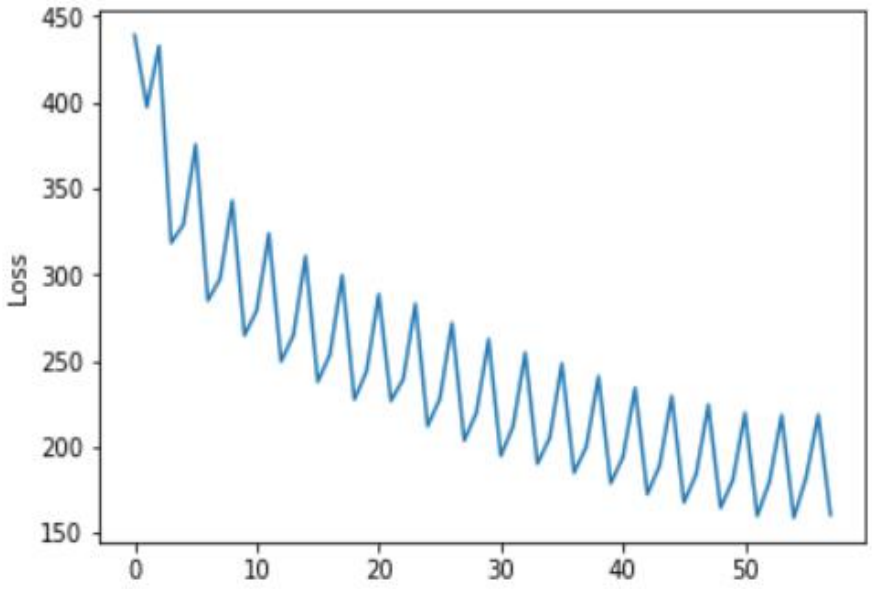
Training Snippet

Here is one of the snippet of random epoch :

```
Epoch   5/50 Batch  325/485 - Loss:  1.912, Seconds: 8.33
Epoch   5/50 Batch  330/485 - Loss:  1.601, Seconds: 10.88
Epoch   5/50 Batch  335/485 - Loss:  1.818, Seconds: 10.37
Epoch   5/50 Batch  340/485 - Loss:  1.770, Seconds: 9.85
Epoch   5/50 Batch  345/485 - Loss:  1.828, Seconds: 9.95
Epoch   5/50 Batch  350/485 - Loss:  1.893, Seconds: 10.15
Epoch   5/50 Batch  355/485 - Loss:  1.774, Seconds: 10.84
Epoch   5/50 Batch  360/485 - Loss:  1.765, Seconds: 11.50
Epoch   5/50 Batch  365/485 - Loss:  1.804, Seconds: 11.64
Epoch   5/50 Batch  370/485 - Loss:  1.884, Seconds: 11.15
Epoch   5/50 Batch  375/485 - Loss:  1.905, Seconds: 10.50
Epoch   5/50 Batch  380/485 - Loss:  1.908, Seconds: 11.44
Epoch   5/50 Batch  385/485 - Loss:  1.870, Seconds: 10.79
Epoch   5/50 Batch  390/485 - Loss:  1.715, Seconds: 11.66
Epoch   5/50 Batch  395/485 - Loss:  1.961, Seconds: 11.04
Epoch   5/50 Batch  400/485 - Loss:  1.934, Seconds: 11.74
Epoch   5/50 Batch  405/485 - Loss:  1.864, Seconds: 11.97
Epoch   5/50 Batch  410/485 - Loss:  1.836, Seconds: 12.26
Epoch   5/50 Batch  415/485 - Loss:  1.980, Seconds: 13.25
Epoch   5/50 Batch  420/485 - Loss:  1.984, Seconds: 13.53
Epoch   5/50 Batch  425/485 - Loss:  1.891, Seconds: 12.38
Epoch   5/50 Batch  430/485 - Loss:  1.960, Seconds: 13.12
Epoch   5/50 Batch  435/485 - Loss:  2.046, Seconds: 13.40
Epoch   5/50 Batch  440/485 - Loss:  2.091, Seconds: 13.74
Epoch   5/50 Batch  445/485 - Loss:  2.050, Seconds: 13.97
Epoch   5/50 Batch  450/485 - Loss:  2.056, Seconds: 13.71
Epoch   5/50 Batch  455/485 - Loss:  1.940, Seconds: 15.09
Epoch   5/50 Batch  460/485 - Loss:  2.230, Seconds: 14.83
Epoch   5/50 Batch  465/485 - Loss:  2.151, Seconds: 16.89
Epoch   5/50 Batch  470/485 - Loss:  2.210, Seconds: 17.93
Epoch   5/50 Batch  475/485 - Loss:  2.178, Seconds: 19.54
Epoch   5/50 Batch  480/485 - Loss:  2.295, Seconds: 19.87
Average loss for this update: 1.941
No Improvement.
```

```
Epoch   6/50 Batch    5/485 - Loss:  1.934, Seconds: 6.25
Epoch   6/50 Batch   10/485 - Loss:  1.448, Seconds: 6.91
Epoch   6/50 Batch   15/485 - Loss:  1.462, Seconds: 5.89
Epoch   6/50 Batch   20/485 - Loss:  1.376, Seconds: 7.58
Epoch   6/50 Batch   25/485 - Loss:  1.501, Seconds: 5.38
Epoch   6/50 Batch   30/485 - Loss:  1.397, Seconds: 7.69
Epoch   6/50 Batch   35/485 - Loss:  1.483, Seconds: 7.04
Epoch   6/50 Batch   40/485 - Loss:  1.570, Seconds: 5.47
Epoch   6/50 Batch   45/485 - Loss:  1.497, Seconds: 7.14
Epoch   6/50 Batch   50/485 - Loss:  1.485, Seconds: 7.66
Epoch   6/50 Batch   55/485 - Loss:  1.485, Seconds: 4.91
Epoch   6/50 Batch   60/485 - Loss:  1.505, Seconds: 4.95
Epoch   6/50 Batch   65/485 - Loss:  1.447, Seconds: 6.04
Epoch   6/50 Batch   70/485 - Loss:  1.375, Seconds: 6.62
Epoch   6/50 Batch   75/485 - Loss:  1.594, Seconds: 7.12
Epoch   6/50 Batch   80/485 - Loss:  1.473, Seconds: 7.09
Epoch   6/50 Batch   85/485 - Loss:  1.452, Seconds: 6.88
Epoch   6/50 Batch   90/485 - Loss:  1.485, Seconds: 6.29
Epoch   6/50 Batch   95/485 - Loss:  1.478, Seconds: 6.73
Epoch   6/50 Batch  100/485 - Loss:  1.399, Seconds: 8.50
Epoch   6/50 Batch  105/485 - Loss:  1.430, Seconds: 8.53
Epoch   6/50 Batch  110/485 - Loss:  1.433, Seconds: 7.55
Epoch   6/50 Batch  115/485 - Loss:  1.538, Seconds: 8.19
Epoch   6/50 Batch  120/485 - Loss:  1.412, Seconds: 7.49
Epoch   6/50 Batch  125/485 - Loss:  1.510, Seconds: 6.96
Epoch   6/50 Batch  130/485 - Loss:  1.440, Seconds: 6.44
Epoch   6/50 Batch  135/485 - Loss:  1.460, Seconds: 8.11
Epoch   6/50 Batch  140/485 - Loss:  1.602, Seconds: 7.12
Epoch   6/50 Batch  145/485 - Loss:  1.436, Seconds: 7.71
Epoch   6/50 Batch  150/485 - Loss:  1.491, Seconds: 7.11
Epoch   6/50 Batch  155/485 - Loss:  1.512, Seconds: 7.75
Epoch   6/50 Batch  160/485 - Loss:  1.432, Seconds: 6.71
Average loss for this update: 1.486
New Record!
```

Loss Graph



Testing Snippet

```
Cleaned Text [328, 7836, 523, 13, 1153, 611, 4044, 61, 3292, 333, 113, 3233, 3, 2487, 812, 468, 113, 13, 2138, 2139, 7
60, 4888, 1619, 113, 3233, 3, 1411, 1381, 61, 460]
INFO:tensorflow:Restoring parameters from ./best_model.ckpt
Index Value:  2805

Original Text: This arrived earlier than expected, which was great! However, for some reason, this particular product
smelled a little like "fish food".... but no worries, still tasted like great cashew butter, just the smell was kinda
weird - like fish food. Also, I wish this product was organic.
File Written to text.001.txt

Orignal Summary:  I love cashew butter!

Text
  Word Ids:    [328, 7836, 523, 13, 1153, 611, 4044, 61, 3292, 333, 113, 3233, 3, 2487, 812, 468, 113, 13, 2138, 2139,
760, 4888, 1619, 113, 3233, 3, 1411, 1381, 61, 460]
  Input Words: arrived earlier expected great however reason particular product smelled little like fish food worries
still tasted like great cashew butter smell kinda weird like fish food also wish product organic

Summary
  Word Ids:       [96, 113, 105, 523, 25386]
File Written to text.A.001.txt
  Response Words:  tastes like i expected bajillion
Time:  45.358612298965454
```

4. Conclusion

We have implemented Attentive RNN Model for Text Summarizer. We have used model which simplified version of the encoder-decoder framework for learning. The model is trained on the Amazon Fine Food Review to generate summaries of reviews based on the sentences and lines of each review. There are few limitations which we faced as the approach we used kind of outdated which can worked out and improved in the further works. First limited is that sometimes it generates repeated words in the summary, the other problem is that when input text size is large its takes too much time. Other Issue, for large text input sometimes it misinterpret the context and generates exactly reverse context summary. Also it has issue implementing on Tensorflow of version 2.0 or above although it can optimized using other alternatives.