Abstractive Text Summarizer using RNN and NLP

Arvind Narayan Srinivasan
School of Computer Science
University of Windsor
Windsor, Canada
srini11c@uwindsor.ca

Gundeep Singh
School of Computer Science
University of Windsor
Windsor, Canada
singh1x7@uwindsor.ca

Vigneshwaran Balasubramani
School of Computer Science
University of Windsor
Windsor, Canada
balasubv@uwindsor.ca

***Abstract*—In an era where data is generated every second, it is important to be concise about what we are trying to convey.More often we find redundant data than the abstract form of information to be conveyed. Text Summarizers can help to create an abstract summary of large set of information. There are generally two types of text summarizers. The first is the extractive summary where we get only the key features that is a set of important lines from the summary which often can dilute the essence of the summary. The next is the abstractive summary which creates a concise version that may or may not be in the original summary i.e the model tries to formulate its own sentence to create the summary.In this paper, our proposed method is to create an automatic text summarizer which follows the abstractive method and able to respond to some short length text summary. For the model, we use NLP for data processing and RNN to build a deep neural network and Long Short Term Memory for the sequence to sequence modules. We have stacked a 3 layer LSTM model with attention layer to overcome the encoder-decoder problem. Stacked LSTMs represents better sequence models. We use early stopping to avoid the overtime of the training process and getting a BLEU(Bilingual Evaluation Understudy) score of 0.29 compared to Human translation score of 0.34.**

***Index Terms*—NLP,RNN,LSTM,BLEU**

I. Introduction

The human brain is capable enough to read, observe and process information in a manner efficient enough to draw key details from the given data so as to focus or remember only the information that is of any use. An automated text summarizer is a simulation of the human approach of summarizing large information in a way that all the key information is still retained.

There are two types of Text Summarizers: *Extractive* and *Abstractive*. The summary obtained using an extractive text summarizer contains exact same sentences and instances from the original data while the abstractive summaries might or might not have the exact sentences from the original data. Clearly, Abstractive Summarizers seem to be a better approach as the summaries are drawn after a semantic understanding of the original data with required tailoring [1]. For our project we have followed the abstractive approach.

For our implementations we have used *Amazon fine food reviews dataset*. We used 10 percent of the total data for validation and rest for training. The model depending on the machine configuration takes approximately from 2 to 3 hours for training and gives satisfactory results.

The data, before it is fed to the model for training is pre-processed and cleaned extensively using various approaches involving stop word removal, contraction mapping, Regular Expression etc. For our working model we have used sequence to sequence model, the encoders are used to take the source text and converts them into fixed length vector form and the decoder further generates the output sequence. We have used many to many Long Short Term Memory Networks(LSTM) for the implementation of the model. LSTMs are special kind of Recurrent Neural Networks used for processing long sentence in which the context of a full long sentence needs to be considered. LSTMs are also excellent in mitigating the problem of vanishing gradient.

II. Related Works

There are several works taken up by machine learning enthusiasts all over the world to come up with an optimal solution for this problem. Here are some top ideas which we have refered to, to create our model.

With sequence to sequence model using RNN, Masum et al in [1] have implemented an automatic text summarizer that follows abstractive method to create a short length text summary. The author has used the amazon fine food reviews as the dataset that contains 568,454 fine food reviews, but they have used only 20,000 dataset from it. They have used bidirectional RNN with the sequence to sequence model that uses a LSTM with Bahdanau attention. For the data processing, they first convert the entire input sequence to lowercase, then add contractions and remove stop words and lemmatize all the words to give out the purified summary text. For word embedding, they use ConceptNet Numberbatch for the vocabulary size and count those words which occurred more than 20 times in the dataset. They also add to tokens to enhance the machine vocabulary. They have defined the hyperparamaters such as epoch=100, batch size=64, rnn's size=256, learning rate=0.005, keep probability=0.75.In the conclusion, the model cannot predict accurately for few reviews but for other reviews, it provides great performance and fluent summary. A major limitation in their model is the machine provides correct output only for the short text.

In [2], the author proposes a graph based method for abstractive summarization. The author shows examples that sentiments play a vital role in decision making and graphs can be used to represent these sentiment infusions. The author

forms a word graph and ensures the sentence correctness using the POS constraints. The final step is the abstractive summary by scoring of path. The author also used abstract meaning representation where each sentence is represented as a rooted, directed, acyclic graph with labels on edges (relations) and leaves (concepts). Apart from these, the author also proposes vertex constrained shortest path where he uses undirected word graph and combined score of bigram. To conclude, Abstractive Meaning representation parser is used to generate the abstractive summary with the help of semantic information and it is also used for the multi document.

In [3], author cites some significant research work done on abstractive text summarization. The author proposed commonly used model that is based on encoder-decoder architecture. It consists of two RNN/LSTM/GRU with/without attention mechanism that helps us to convert input sequence (documents) into out sequence (summary). The author cites two different methods based on a research paper. In continuous skip gram model(CSG) model,it takes the input word and predict the probability of context words. While continuous bag of words(CBOW) is just opposite it takes the context word as input and predicts a single word that fixes in that context. The author specifies the disadvantages of RNN and for the exploding gradient problem is solved by gradient clipping that is restricting the value in between maximum and minimum value. The vanishing gradient problem is solved in LSTM and GRU.

In [4], the author creates survey of Extractive and Abstractive Text Summarization Techniques. In extractive techniques, a paper proposes five features regarding to text summarization and the PSO was used to train the system to obtain the weights of each feature. In another paper, they have used a feature selection method using pseudo genetic probabilistic based summarization(PGPsum). For abstractive techniqies, a paper proposes graph based framework that generates textual graph that represents the text to be summarized. In another paper, a summarization system SUMMARIST that is based on the equation of sum of topic identification,interpretation and generation.

In [5],the author specifies certain examples of sequence to sequence model through bidirectional RNNs mainly: a text summarizer and a chatbot.We will focus more precisely on the text summarizer part. The author proposes a text summarization tool that generates complete, meaningful and consistent paraphrasing of lengthy documents based on their custom word embedding that uses word2vec algorithm. The author has built a encoder-decoder RNN model and maps it to a fixed length vector called context vector and train them with LSTM and obtained a BLEU score of 0.21.

III. Approach

A. Word Embedding and Feed Forward Neural Network

The initial approach of the model preparation was the traditional approach of creating word embedding matrix with each word being mapped with a certain number of features. In this approach a trained model tries to find the similarity between the data it already knows and the data it is given to make a prediction of the next expected word in a sequence.

Each word in this approach is a vector and the features are the dimensions of the vector. So the similarity of the words can be deduced using any of the available similarity functions.

	MAN(3000)	WOMEN(9000)	KING(4700)
GENDER	-1	1	-0.95
ROYAL	0.03	0.02	0.93
AGE	0.02	0.01	0.5
FOOD	0.01	0.00	0.03

Fig. 1. Example of Word Embedding Matrix.

In the figure(Fig 1.) the words(*Man, Women and King*) are mapped to features(*Gener, Royal and Age etc*).

It's worth noticing that this technique can be further enhanced if we use a neural network to make the predictions in a sequence. The neural network can be trained on the word Embedding matrix of the whole set of vocabulary with certain number of features.

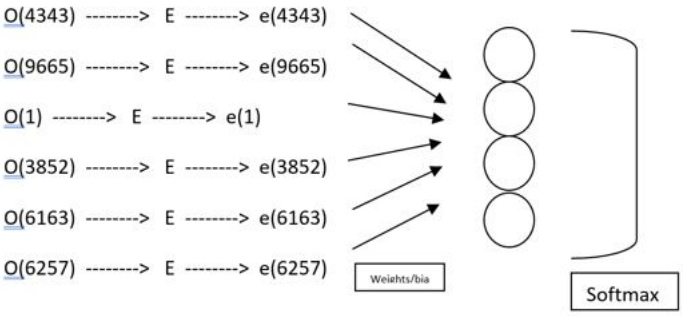

Fig. 2. Feed Forward Network.

For this approach a feed forward network is used since the predictions are based on the already obtained weights and biases while training the model and no update is made thereafter.Also to be considered every word in this approach is represented using a one hot encoding vector and a feature map for a single word can be obtained by multiplying the word vector with the word embedding matrix.

Once a word specific matrix is obtained , all the words of a sequence are fed as an input to the neural network and the weights and bias(already known) are used to predict the next expected word in a sequence. The obtained outputs can vary in number with different probabilities and hence a soft-max layer is used to get the output having maximum probability. [10]

As seen in the above figure(Fig 2.), The inputs are the words(represented in vector form) of a sentence and the

expected output is the last word in the sentence. 'E' represents the word embedding matrix. 'e' represents the single word feature representation. The final output is drawn by using softmax layer.

The above mentioned approach is good only with the short sentences as the need to remember the context of a full context is not necessary but when it comes to long sentences where the intention/subject of the sentence needs to be remembered throughout the process then this feed forward network will fail terribly.

B. RNN and LSTMs

Recurrent Neural Networks(RNN) are a form of deep neural networks that works on the principle of saving the output of a layer and feed it as an input to the next layer.

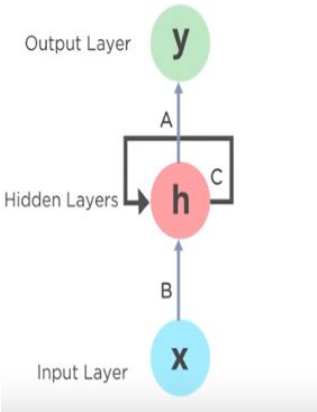

Fig. 3. RNN Structure

The above image is just a flipped version of a feed forward network. If we generalize the nodes and add a loop to the hidden layer, we form a RNN. We have X on the bottom going to the H and going to Y that is the feed forward neural network but right in the middle that has a value C, so there is a whole another process happening there that is memorizing whats going on in the hidden layers and the hidden layers as they produced data feed into the next on, so we have an output that goes off to Y but that output goes back into the next prediction coming in. Hence this allows it to handle sequential data because it considers the current input and the previously received input. Basic RNN is good at capturing the short dependencies in the sequential data. But it fails to capture the long-term dependencies [3].They have certain problems namely vanishing and exploding gradient. To overcome the vanishing gradient(as textual contexts requires long term dependencies), Long short term memory units i.e LSTMs models can be used.

LSTMs are special kind of RNN capable of learning long term dependencies remembering information long periods of time is their default behavior. All RNN have the form of a chain of repeating modules. In RNN, we would have a very simple structure, such as a single tanh layer.LSTMs also have repeating modules but the repeating modules has a different structure instead of having a single layer, there 4 interacting layer communicating in a very special way. We can generalize layers into 3 steps:

1) forget irrelevant parts of the previous state i.e to decide how much of the past to remember

$$f_t = \sigma(W_f.[h_{t-1}, x_t) + b_f \tag{1}$$

2) decides how much should this unit add to the current state

$$f_t = \sigma(w_i.[h_{t-1}, x_t) + b_i \tag{2}$$

$$c_t = tanh(w_c.[h_{t-1}, x_t) + b_c \tag{3}$$

3) decides what part of the current cell state makes to the output

$$o_t = \sigma(W_o.[h_{t-1}, x_t) + b_o \tag{4}$$

$$h_t = o_t * tanh(C_t) \tag{5}$$

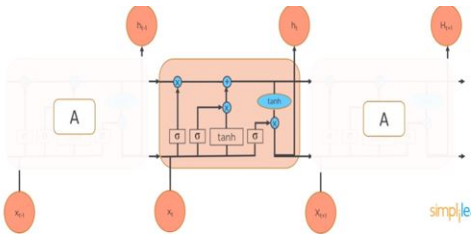

Fig. 4. LSTM Model.

IV. Architecture

We will be implementing an Abstractive text summarizer. The figure below is an overview of the architecture.

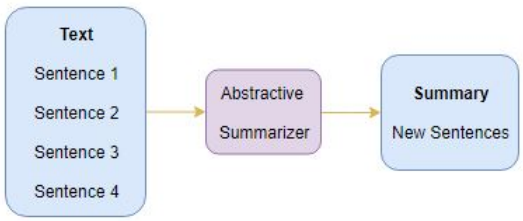

Fig. 5. Generic Abstractive Summarizer Architecture.

From the overview, Abstractive summarizer creates the summary with sentences that are not in the source documentation but conveys the same meaning.It makes use of certain components that help in predicting the words that are needed to form the summary. Lets look at a detailed architecture and the components which help with producing the summary. The architecture is constructed pertaining to a many to many sequence to sequence problem. The two major components are

- Encoder
- Decoder

The Encoder-Decoder architecture is mainly used to solve problems where input and output sequences are of different lengths. The architecture uses a variant of RNN called

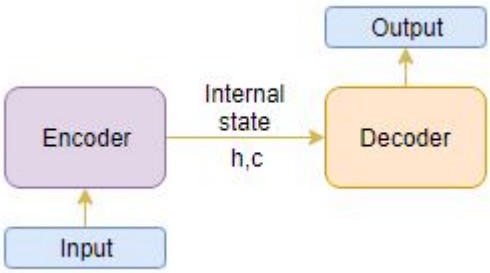

Fig. 6. Detailed Architecture.

LSTM(Long Short Term Memory) for the encoder and decoder components as they can capture long term dependencies and can overcome the problem of vanishing gradient.

A. Encoder

Encoder is a series of LSTMs connected in a sequence where the input sequence is fed as one word per each time-step and it converts them to a fixed length vector [1]. The contextual information of the input sequence is captured and stored in the internal state by the encoder. If we expand the encoder component we get the following structure The states h_i and

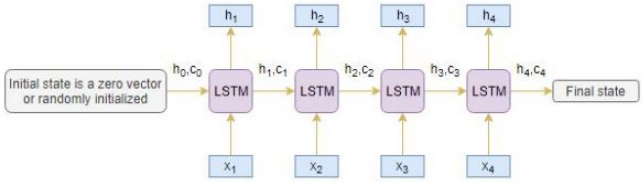

Fig. 7. Encoder expanded.

c_i are the hidden and cell states respectively. As the encoder and decoder are two different components in the Architecture the Encoder has to store the information in the cell state and initialize the hidden state in order to be passed to the decoder.

B. Decoder

The Decoder also contains a series of LSTM architecture, it reads the entire input sequence which is the output of the Encoder and predicts the offset of the sequence per time-step. Given a word the decoder is trained to predict the next word in the sequence. If we expand the decoder we get the following structure Before feeding the input to the decoder, two special tokens called <start> and <end> are added to the beginning and the end respectively. This is due to the fact that the Decoder will be unaware of the target sequence which is given a head start by the <start> token which contains the first word from the input sequence and <end> is a signal to mark the end of sequence.

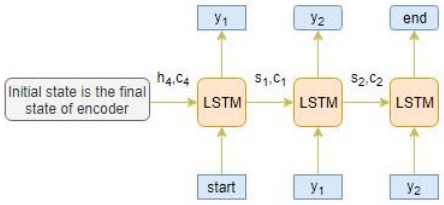

Fig. 8. Decoder expanded.

C. Inference Architecture

Inference architecture combines the Encoder and Decoder components and defines a sequence of steps which defines the working of model.

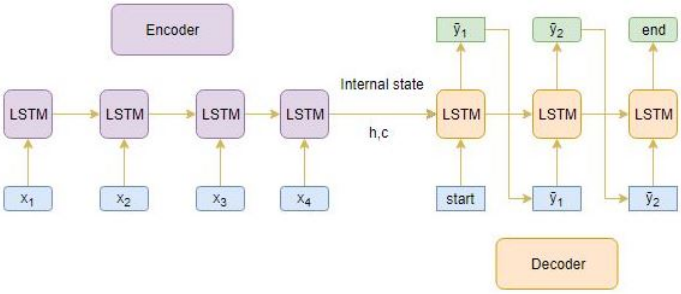

Fig. 9. Inference Architecture.

The below are the steps to predict the summary using the input sequence

1) Encode the input sequence and initialize decoder with internal states from encoder
2) Define the <start> token as input to decoder
3) For each time-step run the Decoder
4) The predicted word will be the word with maximum probability. This prediction is then passed to the subsequent nodes and the internal states are updated with the current prediction.
5) Repeat the steps 3 - 4 until the <end> token is reached

However, there is one important issue with the Encoder-Decoder approach. It is that the input sequence should be compressed to a fixed length vector which renders it difficult to cope up with long sentences. Hence the performance will be forced to depend on the length of input vector.

D. Attention Mechanism

Attention Mechanism [6] is used to predict a word by looking at some specific parts of the sequence only rather than the whole sequence. This helps with mitigating the dependency on the length of input vector. The amount of attention to be given to each word at a given time-step **t** alters the performance of the model. We calculate alignment score (e_{ij}) [8] which defines to what extent the source word is related with the target word. It is calculated as a dot product of source hidden state h_j and target hidden state s_i using a

function called score.

$$e_{ij} = score(s_i, h_j) \tag{6}$$

where i and j represents the target and source time-step respectively. The obtained score is normalized using a soft max function to retrieve the attention weight

$$a_{ij} = e^{e_{ij}} / \sum_{k=1}^{T_x} e^{e_{ik}} \tag{7}$$

Context vector is computed by linear sum of products of the attention weights a_{ij} and hidden states of the encoder h_j [7]

$$C_i = \sum_{j=1}^{T_x} a_{ij} h_j \tag{8}$$

A hidden vector is calculated using the attended context vector and the target hidden state of the decoder

$$S_i = concatenate([s_i; C_i]) \tag{9}$$

The attended hidden layer is then passed to the dense layer to produce y_i

$$y_i = dense(S_i) \tag{10}$$

The above steps are repeated to find the specific words with high scores which are then used to predict the entire summary.

V. Data Processing

Clean data is essential part of the project and preprocessing of the data is one challenging part before the actual implementation of the model. In order to understand the intention of the writer the full sentence is never required, only a few words from the sentence are enough. For example: In the sentence "*Jhon told me over the phone yesterday that his favorite game is cricket*", the words 'favorite','John', 'game' and 'cricket' are enough to understand the intention of the speaker. This arises a need to clean the data extensively to make it fit for learning.

In our model we have used various Natural Language Processing libraries like *"NLTK"*(Natural Language Tool Kit), *"re"*(Regular expression) and *"BeautifulSoup"* etc.

Given figure is a flow chart of how the data is processed in a formal sequence.

As seen in the above figure (Fig. 7), the text first as a whole is converted to lower case. We used BeautifulSoup library to remove all the unnecessary tags from the input text. The next phase is of Contraction mapping. In this phase the abbreviation/slang used are changed to the full legit words and phrases, for example: *'Ain't'* is changed to *'is not'*. Contraction mapping is a predefined set where each word/abbreviation is mapped to its actual word/phrase. Next we use the regular expression library of machine learning to find and remove certain patterns from the text. This includes removing apostrophe, periods and any other occurring special character. At the last the text is refined and all the stopping words that are of least importance to us are finally removed using the nltk library.

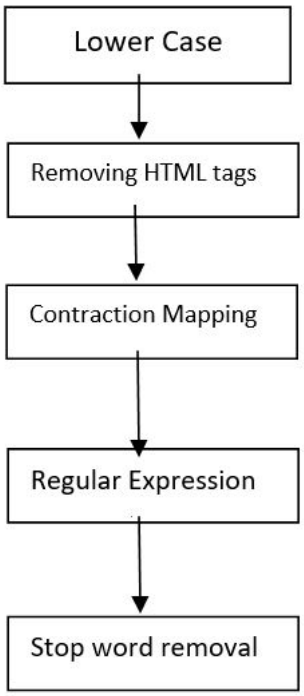

Fig. 10. Example of Word Embedding Matrix.

This step is done for both the text and the summary column of our dataset and clean text and summary are thus obtained for learning and further processing by the model.

VI. Implementation Details

We concentrate on the Text and Summary features of the dataset. Which are cleaned and preprocessed according to the above steps. They are then tokenized using text and summary tokenizers to be able to convert the word sequence to integer sequence.

A. Loss function

We use sparse categorical cross-entropy as the loss function since it is efficient in converting the integer sequence to one-hot vector. Early stopping is used as a normalisation technique to find the right epoch to stop the training of the model when the validation loss increases.

B. Training the model

The model is trained as explained in the initial set up of Encoder and Decoder. 90% of dataset is used as training set. The model is trained using one word per each time-step and the cell state and hidden states are updated for use in the decoder. The decoder then learns to predict the next word from the training data.

C. Testing the model

The model is tested using a 10% validation set.Here the Decoder will be unaware of the next word and it will predict the next word given the previous word using attention mechanism and updates it's own hidden and cell states. The prediction is further passed through a 3 stack LSTM function to create the summary.

The training and testing are performed according to the inference steps explained in the above section.

VII. Result and Conclusion

Text summarizers has become an essential part in this digital era; In this paper, we achieve this through abstractive text summarization with help of encoder-decoder architecture coupled with RNN and LSTMs model stacked up. From the graph below, we can see that as the number of epochs increases, the accuracy loss plummets to zero with the addition of early stopping:

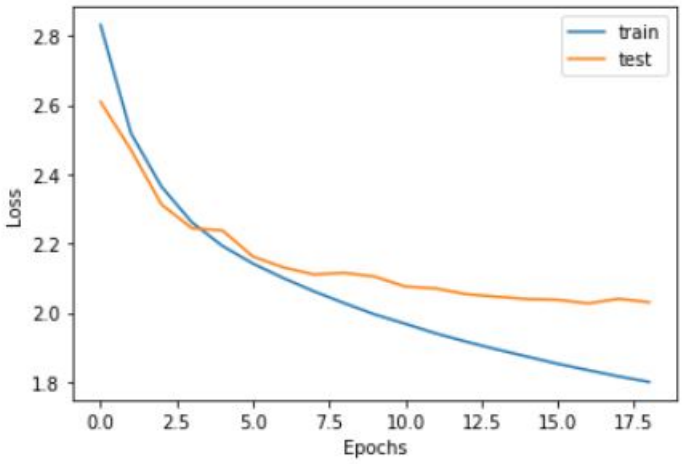

Fig. 11. Epochs vs Loss.

Although few reviews do not give accurate results, we cannot achieve hundred percent prediction rate but we get a BLEU score of 0.29 compared to human bilingual evaluation understudy score of 0.34 which is comparatively a good score for a machine to predict a summary.

```
Review: introduced number people hooked best sour gummy ever great flavors got great price
Original summary: new favorite
Predicted summary:  best candy ever


Review: new price attractive however tastes horrible maybe old zico coconut water brands might find acceptable
Original summary: do not be by the price
Predicted summary:  bad taste
```

Fig. 12. Results.

VIII. Future Works

The scope of the project is very broad and not just limited to the current efficiency. The efficiency can be improvised using various methods. Increasing the size of the dataset is one of the most basic yet common way.

Instead of the usual LSTMs, a bi-directional LSTM can be used. Bi-directional LSTM can capture context from both sides and hence can give better results. Also instead of the simple argmax(greedy approach) for decoding the test sequence , beam search strategy can be used.

Acknowledgment

We would like to thank our course instructor Dr. Alioune Ngom for imparting us all the vital knowledge of the subject Artificial Neural Networks and presenting us with an opportunity to build this application model.

Under his supreme guidance and through this project we have gained tremendous knowledge of various Natural Language Processing techniques for handling text data and application of advanced Deep Learning techniques using Recurrent Neural Networks.

The entire course curriculum constantly provided us with adequate information to make this model implementation possible.

References

[1] A. K. M. Masum, S. Abujar, M. A. I. Talukder, A. S. A. Rabby, and S. A. Hossain, "Abstractive method of text summarization with sequence to sequence RNNs," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019.

[2] C. Badgujar, V. Jethani, and T. Ghorpade, "Abstractive Summarization using Graph Based Methods," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018.

[3] N. Raphal, H. Duwarah, and P. Daniel, "Survey on Abstractive Text Summarization," 2018 International Conference on Communication and Signal Processing (ICCSP), 2018.

[4] V. Dalal and L. Malik, "A Survey of Extractive and Abstractive Text Summarization Techniques," 2013 6th International Conference on Emerging Trends in Engineering and Technology, 2013.

[5] G. Aalipour, P. Kumar, S. Aditham, T. Nguyen, and A. Sood, IEEE International Conference on Big Data, Department of Computer Science, University of Colorado Boulder, Boulder, CO, 2018.

[6] M. Luong, H. Pham, Christopher D. Manning "Effective Approaches to Attention-based Neural Machine Translation". Conference on Empirical Methods in Natural Language Processing (EMNLP 2015).

[7] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: first results. CoRR, abs/1412.1602.

[8] Long, Dang Nguyen, Minh-Tien Xuan Bach, Ngo Nguyen, Le Phuong, Tu. (2018). An Entailment-based Scoring Method for Content Selection in Document Summarization. 122-129. 10.1145/3287921.3287976.

[9] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," IBM, 2002.

[10] Pai, "Comprehensive Guide to Text Summarization using Deep Learning in Python",2019.