TEXT SUMMARIZATION USING DEEP LEARNING – LSTM APPROACH

Thesis

Submitted In Partial Fulfillment Of The Requirement For The Degree of

MASTER OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

BY

DIGANTA KUMAR DAS

University Roll Number: 002010502021

Examination Roll Number: M4CSE22021

Registration Number: 154145 of 2020-2021

Under The Guidance Of

PROF. DIGANTA SAHA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY

JADAVPUR UNIVERSITY, KOLKATA

132, Raja Subodh Chandra Mallick Road

Jadavpur, Kolkata, West Bengal, 700032

AUGUST, 2022

CERTIFICATE OF RECOMMENDATION

This is to certify that the dissertation titled TEXT SUMMARIZATION USING DEEP LEARNING – LSTM APPROACH was completed by DIGANTA KUMAR DAS, University Roll No: 002010502021, Examination Roll Number: M4CSE22021, University Registration No: 154145 of 2020-21, under the guidance and supervision of Prof. Diganta Saha, Department of Computer Science and Technology, Jadavpur University. The findings of the research detailed in the thesis have not been incorporated into any other work submitted for the purpose of earning a degree at any other academic institution.

Prof. Diganta Saha
Department of Computer Science & Engineering
Jadavpur University

COUNTERSIGNED BY

Prof. Nandini Mukhopadhyay
Head of The Department
Department of Computer Science and Engineering
Jadavpur University

COUNTERSIGNED BY

Prof. Chandan Mazumdar
Dean, FET
Department of Computer Science and Engineering
Jadavpur University

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

This is to certify that the thesis entitled TEXT SUMMARIZATION USING DEEP LEARNING – LSTM APPROACH is a bonafide record of work carried out by DIGANTA KUMAR DAS in partial fulfilment of the requirements for the award of the degree Master of Engineering in Department of Computer Science and Engineering, Jadavpur University during the period of June 2021 to August 2022 (3rd& 4th Semester). It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for the purpose for whichit has been submitted.

Signature of Examiner

Date:

Signature of Supervisor

Date:

DECLARATION

I certify that,

a) The work TEXT SUMMARIZATION USING DEEP LEARNING – LSTM APPROACH contained in this report hasbeen done by me under the guidance of my supervisor.

b) The work has not been submitted to any other Institute for any degree or diploma.

c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Diganta Kr Das
Master of Engineering
Roll No.: 002010502021
Registration No.: 154145 of 2020-21
Department of Computer Science and Engineering
Jadavpur University, Kolkata

ACKNOWLEDGEMENT

First and foremost, I want to express my gratitude to God Almighty for providing me with the strength, wisdom, and capability to go on this amazing adventure and to continue and successfully finish the embodied research work. I'd like to thank Professor Diganta Saha of Department of Computer Science and Engineering at Jadavpur University for his excellent assistance, consistent support, and inspiration during the course of my dissertation. I owe Jadavpur University a great debt of gratitude for providing me with the chance and facilities to complete our thesis.
I am grateful to every one of the teaching and non-teaching personnel whose assistance has made the trip during my research time much easier. Special thanks to our HOD Prof. (Dr.) Nandini Mukhopadhyay ma'am. I would like to thank my seniors, and my friends for providing me with regular encouragement and mental support throughout our effort.
Last but not the least, my family deserves great recognition. There are no words to express my gratitude to my mother and father for all of the sacrifices they have made on my behalf. Their prayers for me have kept me going this far.

Diganta Kr Das
Master of Engineering
Roll No.: 002010502021
Registration No.: 154145 of 2020-21
Department of Computer Science and Engineering
Jadavpur University, Kolkata

ABSTRACT

Text summarization is a process of extracting the context of a large document and summarize it into a smaller paragraph or a few sentences. Machine learning and deep learning, as we know, have started ruling over almost every field in the computing industry and so, has revolutionized the process of text summarization too. Automatic text summarization is an advancing realm of the natural language processing research in which concise textual summaries are generated from lengthy input documents. Neural network models have been provided a new feasible approach for abstractive text summarization. Abstractive means generating new sentences from the original text which might not be present in the original text. These neural network models have two defects: They are likely to reproduce factual details inaccurately and they tend to repeat themselves. Extensive research has been carried out on how automatic summarization can be prosecuted through various extractive and abstractive techniques. It is also used in many bigger project implementations of classification of documents or in search engines. This presentation presents a method of achieving text summaries accurately using deep learning methods, mainly focusing on LSTMs(Long Short Term Memory) for prediction accuracy.

Key Words: Text Summarization, Deep Learning, Long Short Term Memory, Natural Language Processing, Abstractive summary, Extractive summary.

List of Tables:

List of Figures:

List of Abbreviations:

1. LSTM – Long Short Term Memory
2. LDA - Latent Dirichlet Allocation
3. NLP – Natural Language Processing
4. LSA – Latent Semantic Analysis
5. RNN – Recurrent Neural Network
6. ROUGE - Recall-Oriented Understudy for Gisting Evaluation
7. NLTK - Natural Language Toolkit
8. POK- Position of a Keyword
9. HMM -Hidden Markov Model
10. DT -Decision Trees
11. ME -Maximum Entropy
12. NN -Neural Networks
13. NB -Naïve Bayes
14. DLCSS -Direct lexical chain span score
15. DLCS -Direct lexical chain score
16. LCSS -Lexical chain span score
17. LCS -Lexical chain score
18. RST -Rhetorical Structure Theory
19. LC -Lexical chain
20. GPR -Google's Pagerank
21. HITS -Hyperlinked Induced Topic Search

22.MLSA -Meta Latent Semantic Analysis

23.SNMF -Symmetric nonnegative matrix factorization

24.SLSS -Sentence level semantic analysis

25.NMF -Non-Negative Matrix factorization

26. SVD -Singular Value Decomposition

27.SDD -Semi-Discrete Decomposition

28.TF -Term Frequency

29.IF-IDF - Term Frequency-inverse document frequency

CONTENTS

Chapter 1
INTRODUCTION

1.1 Definition:

Text summarization is a process of producing brief and concise summary by capturing the vital information and the comprehensive meaning. Text summarization is achieved by natural language processing techniques by using algorithms like page rank algorithms etc. While these algorithms fulfil the objective of text summarization, they cannot generate new sentences which are not in the document like humans. Deep learning models are usually trained to understand documents and distil the useful information before outputting the required summarized texts. They can also have grammatical errors. This is where Deep Learning comes to our rescue. The use of deep learning builds an efficient and fast model for text summarization. The use of deep learning methods helps us generate summaries which can be formed with new phrases and sentences and also which are grammatically correct.

Application/Use Cases in the industry : Social media marketing, Question answering bots, Medical cases, E-learning and class assignments, Automated content creation.

1.2 Types of Text Summarization :

1. **Extractive Text Summarization :**

The extractive text summarization involves pulling key phrases from the source document and combining them to make a summary. We identify important words or phrases from the text and extract only those for the summary. It uses subset of words or phrases existing in the given data and apply computations to find out a summary.

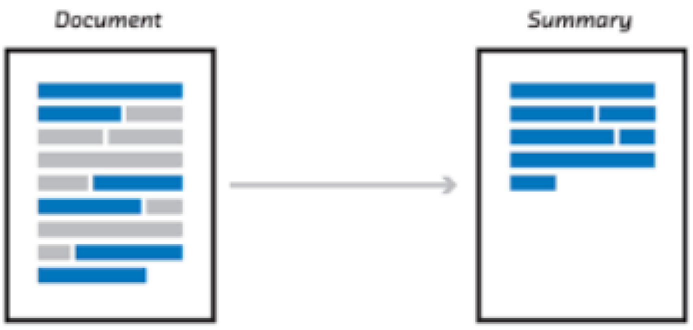


Fig -1: Extractive Text Summarization

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. This is where Artificial Intelligence swoops in, the process uses attention methods to find out the words/phrases or meanings/the general idea of words/phrases and make them up into meaningful summaries through Machine Learning.

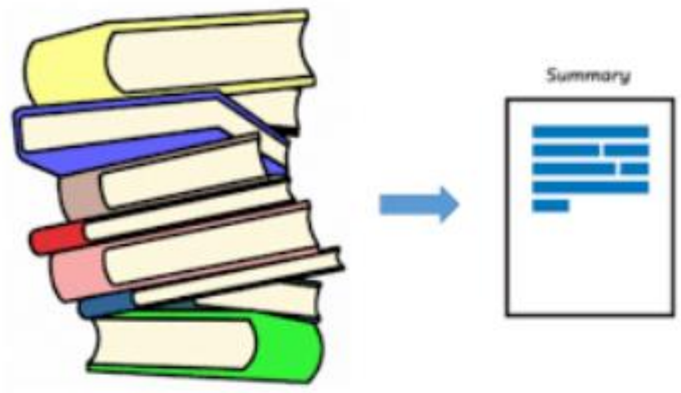


Fig -2: Abstractive Text Summarization

1.3 Problem Statement and Challenges:

We use Abstractive text summarization to predict summaries of our dataset, we are using a newspaper report dataset of the Hindustan Times from 2015-2017. It's an extensive dataset with around 1 lakh rows of data.

Key challenges in text summarization include topic identification, interpretation, summary generation, and evaluation of the generated summary. Most practical text summarization systems are based on some form of extractive summarization.

1.1 **Topic Identification** :

Topic Identification is a method for identifying hidden subjects in enormous amounts of text. The Latent Dirichlet Allocation (LDA) technique is a common topic modeling algorithm that has great implementations in Python's Gensim package. The problem is determining how to extract high-quality themes that are distinct, distinct, and significant. This varies depending on the text preparation quality and the approach for determining the ideal number of subjects.

1.2 **Topic Interpretation** :

Data interpretation refers to the process of using diverse analytical methods to review data and arrive at relevant conclusions. The interpretation of data helps researchers to categorize, manipulate, and summarize the information in order to answer critical questions.

The importance of data interpretation is evident and this is why it needs to be done properly. Data is very likely to arrive from multiple sources and has a tendency to enter the analysis process with haphazard ordering. Data analysis tends to be extremely subjective. That is to say, the nature and goal of interpretation will vary from business to business, likely correlating to the type of data being analyzed. While there are several different types of processes that are implemented based on individual data nature, the two broadest and most common categories are "quantitative analysis" and "qualitative analysis".

1.3 **Summary Generation**

Summarization is the task of condensing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content. Since manual text summarization is a time expensive and generally laborious task, the automatization of the task is gaining increasing popularity and therefore constitutes a strong motivation for academic research.

There are important applications for text summarization in various NLP related tasks such as text classification, question answering, legal texts summarization, news summarization, and headline generation. Moreover, the generation of summaries can be integrated into these systems as an intermediate stage which helps to reduce the length of the document.

1.4 Evaluation of generated summary

The summarization systems are ranked according to the similarity of the main topics of their summaries and their reference documents. Results show a high correlation between human rankings and the LSA-based evaluation measure. The measure is designed to compare a summary with its full text. It can compare a summary with a human written abstract as well; however, using a standard ROUGE measure gives more precise results. Nevertheless, if abstracts are not available for a given corpus, using the LSA-based measure is an appropriate choice.

Proposed Approach :

We approach we have taken is by using Bidirectional LSTMs in 3 different layers in a Sequence to Sequence model following encoder-decoder architecture where the output of the first LSTM layer is treated as the input for the second. The Attention Layer used in this is a custom, open source Attention Layer.

Detailed explanation of all of these follows later.

The rest of the thesis is organised as follows:

Chapter 2: This chapter reviews the related works on the topic of text summarization, all the different approaches, their characteristics, advantages and disadvantages.

Chapter 3: This is the main chapter discussing all about the proposed work, explanation of the entire flow of the project, details of datasets, models used, functions/methods used.

Chapter 4: This chapter discusses the results obtained using the proposed model and analyses the results.

Chapter 5: This chapter provides a brief conclusion, discusses about the challenges faced and future scope of the project.

Chapter 2
LITERATURE SURVEY

Text Summarization is a broad dimension in its own aspect, let us dive deep into different approaches towards the problem taken by researchers over all these years.

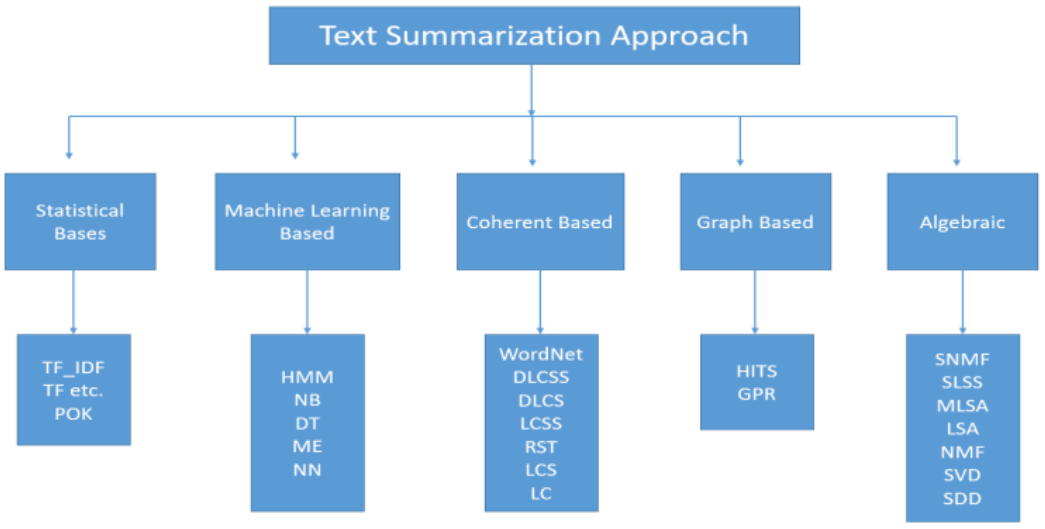


FIG. 3. TEXT SUMMARIZATION APPROACHES

2.1 Machine Learning Approaches:

Machine learning based approaches learn from the data. They can be supervised, unsupervised or semi-supervised. In supervised approach, there is a collection of documents and their respective human-generated summaries such that useful features of sentences can be learnt from them. Supervised or trainable summarizers classify each sentence of the test document either into “summary” or “non-summary” class with the help of a training set of documents. Large amount of labeled or annotated data is needed for learning purpose. Naïve Bayes classification (Fattah 2014)[1], Support Vector machine (SVM) (Fattah 2014)[1], Decision trees, Neural networks (Multilayer Perceptron) (Fattah and Ren 2009)[2], Mathematical Regression (Fattah and Ren 2009)[2] etc. are some of the supervised learning algorithms. On the other hand, unsupervised systems do not require any training data. They generate the summary by accessing only the target documents. They try to discover hidden structure in the unlabelled data. Thus, they are suitable for any newly observed data without any advanced modifications. Such systems apply heuristic rules to extract highly relevant sentences and generate a summary. Genetic algorithms (GA) (Mendoza et al. 2014)[3] is also a type of machine learning approach. Genetic algorithm, being a search heuristic works on the process of natural selection. Belonging to the category of evolutionary algorithms, they solve the optimization problems by using approaches based on natural evolution like mutation, inheritance, crossover and selection. Clustering (Yang et al. 2014), Hidden Markov Model, etc. are some of the examples of unsupervised learning techniques. Semi-supervised learning techniques require labeled and unlabeled data both to generate an appropriate function or classifier.

2.2 Coherent Approach :

Basically, a coherent based approach deals with the cohesion relation between the words. There are various cohesion relations among elements in a text like reference, ellipsis, substitution, conjunction, and lexical cohesion [4]. We can see the classification of this approach in figure 3.

2.3 Graph Based Approach:

In a graph, text elements i.e sentences and words are represented by nodes and edges connect the related text elements together. Baralis et al. (2013)[5] proposed GRAPHSUM, a novel and general-purpose summarizer based on graph model which represents correlations among multiple terms by discovering association rules. Erkan and Radev (2004)[6] proposed LexRank which is a summarization system for multiple documents where those selective sentences are shown in the graph which are expected to be a part of the summary. If similarity among two sentences lies above a given limit, then there is a connection between them in the graph. After the network is made, important sentences are selected by the system by carrying out a random walk on the graph.

2.4 Algebraic Approach:

In algebraic approach we use algebraic theories like, matrix, Eigen vectors, transpose of matrix, etc. There are various algorithms used for text summarization using this approach like SNMF Symmetric nonnegative matrix factorization[7], LSA- Latent Semantic Analysis[8][9][10], SLSS-Sentence level semantic analysis[7], MLSA-Meta Latent Semantic Analysis[11][12], NMF-Non Negative Matrix factorization[13], SDD Semi Discrete Decomposition[14], SVD-Singular Value Decomposition[15],etcetera.

2.5 Statistical Approach:

These approaches deal with some statistical features which help to extract important sentences and words from source text. These techniques are independent of any language, such that if the summarizer is developed using these techniques, then it can summarize text in any language. So, these techniques do not require any additional linguistic knowledge or complex linguistic processing (Ko and Seo 2008)[16]. Also, they require less processor and memory capacity. Some of the statistical features (Fattah and Ren 2009)[2] are position of sentence, positive keyword (based on frequency count), negative keyword (based on frequency count), centrality of sentence (i.e. similarity with other sentences), resemblance of sentence to the title, relative length of the sentence, presence of numerical data in the sentence, presence of proper noun (name entity) in the sentence, node's (sentence's) bushy path, summation of similarities for each node (aggregated similarity), etc. So, for each sentence in the document, a score is computed and highly scored sentences are chosen for generating the summary. Some other features that can discover important words are TF*IDF (Term Frequency–Inverse Document Frequency), information gain, mutual information and residual inverse document frequency. Each of the above features assigns some weight to the words. Based on these weights, the scores are assigned to the sentences and then highly scored sentences are chosen to generate the summary. Figure 1 above displays the block diagram of automatic text summarization system based on statistical approach. Firstly, preprocessing of the source document is done in which linguistic techniques are applied which includes segmentation of sentences, removal of stop-words, removal of punctuation marks, stemming, etc. Segmentation process deals with dividing the text into sentences. Then, elimination of stop-words is done. Words that occur frequently in the text but have no contribution in selecting the important sentences, for example prepositions, articles, pronouns, etc are termed as stop-words. They are considered as noisy terms within the text. So, their removal would be very helpful before a natural processing task executes. Then, stemming is performed. Stemming is the process of reducing the words with the same root or stem to a common form, thus removing the variable suffixes (Manning et al. 2008)[17]. A few popular and efficient stemming algorithms are of Porter and Lovins. Then some features are selected which will help in extraction of important sentences. These features may be statistical or linguistic or a combination of both. For each sentence, all the selected features' scores are computed and then added together to obtain the score of each sentence. Highly scored sentences are then chosen to form the summary while preserving the original order of sentences in the document. Summary's length depends on the compression rate desired.

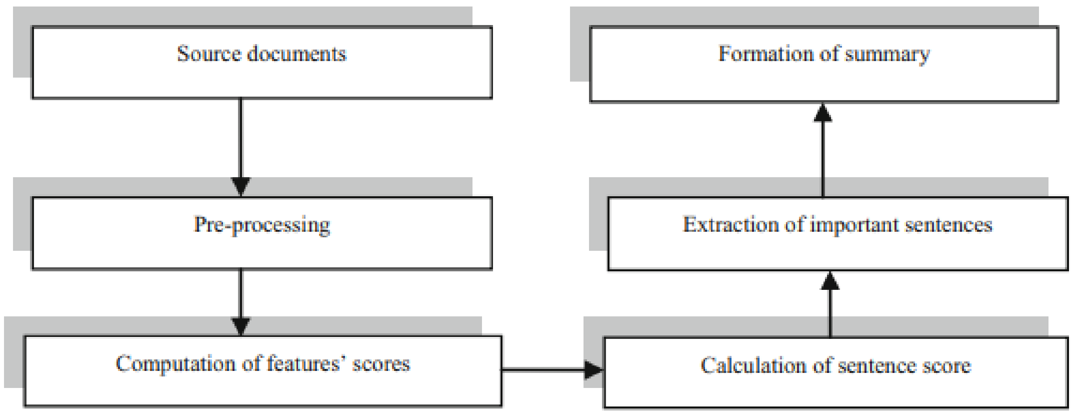

Source documents
Pre-processing
Computation of features' scores
Calculation of sentence score
Extraction of important sentences
Formation of summary

Fig. 4 Block diagram of automatic extractive text summarization system by using statistical techniques

Name	Description	Evaluation (ROUGE-1)
OCDsum-SaDE(Alguliev et al. 2013)	This is an optimization approach for generic summarization of documents. This approach uses an algorithm named as self-adaptive DE (Differential Evolution)	0.4990
Unified Rank(Wan 2010)	This is a graph based approach in which summarization of both single and multiple documents is done simultaneously.	0.4849
BSTM(Wang et al. 2009)	Bayesian Sentence based Topic Model (BSTM) employs both term-sentence and term document associations for summarizing multiple documents	0.4881
FGB(Wang et al. 2011)	Factorization with Given Bases (FGB) is a language model where sentence bases are the given bases and it utilizes document-term and sentence term matrices. This approach clusters and summarizes the documents simultaneously	0.4851
Sum-Coh(Parveen and Strube 2015)	This is graph-based unsupervised technique for extractive summarization of single documents which considers three important properties of summarization, i.e. importance, non-redundancy and local coherence	0.4850

Table 1 : Notable Text Summarization Works and their ROUGE scores

2.6 Research Gaps and Motivation:

The work we are targeting is, as it’s already said falls in the Machine Learning category, moreover specifically we are eyeing a Deep Learning approach using LSTMs and attention layer.

The research gaps in this area have been pretty existent over the years, at first RNN approach was used, the vanishing gradient problem came into play and then the LSTM approach was taken into consideration.

My objective and motivation for this project is to get better evaluation scores like ROUGE-1 scores, to improve the current summarization accuracies achieved,

Chapter 3
PROPOSED WORK

3.1 Basic Idea and Algorithm:

In the current days, we are trying to create algorithms which can help us replicate the human brain and achieve its functionalities. This has been achieved by the neural networks. Neural Networks are the set of algorithms that an recognize patterns in the data. They closely resemble the human brain and have the capability to create models that can work or function like a human brain. Recurrent Neural Network (RNN) are a type of neural networks. They are feedforward neural networks which have an internal memory. In a traditional neural network, the input and the output sequences are independent of the each other. But in order to predict a sequence or a sentence, we need to know the previous words to predict the next word. Hence, we need internal memory. RNN helps us store the previous memory with the help of hidden states which remembers information about previous sequences. The RNN is named so as it recurrently performs the same function on all input of data and the hidden layers. This reduces the complexity of having to store various parameters for each of the layers in the network thus saving the memory. The output of the current input depends on the past outputs too. After the output is produced, it is sent back to the same network so that it can be stored and used for the processing of next output in the same sequence. In order to generate an output in RNN, we consider the current input and the output that was stored from the previous input. RNNs work perfectly when it comes to short contexts. But when we want to create a summary of a complete article, we need to capture the context behind the complete input sequence and not just the output of the previous input. Hence, we need a network that can capture the complete context like a human brain. Unfortunately, simple RNN fails to capture the context or the long term relation of the data that is it cannot remember or recall data in the input that occurred long before and hence cannot make an effective prediction. RNN can remember data or context only for a short term. This is called vanishing gradient problem. This issue can be resolved by a slightly different version of RNN - The Long Short Term Memory Networks Long ShortTerm Memory (LSTM) networks are a better version of RNN. They can remember the past data easily by resolving the vanishing gradient problem. LSTM uses back propagation to train the model. LSTM is well-suited for predictions and classifications of data sequences of unknown durations. They can also be used in language translation and text summarization methods.

3.2 Usage of Model :

The model used here is sequence to sequence model. Sequence-to-sequence learning is a training model that can convert sequences of one input domain into the sequences of another output domain. It is generally used when the input and output of a model can be of variable lengths. It is a method of encoder-decoder based machine translation that maps an input of sequence to an output of sequence with a tag and attention value. The idea is to use two LSTMs that will work together with a special token and try to predict the next state sequence from previous sequence.

3.2.1 RNNs and the need for LSTMs :

RNNs work perfectly when it comes to short contexts. But when we want to create a summary of a complete article, we need to capture the context behind the complete input sequence and not just the output of the previous input. Hence, we need a network that can capture the complete context ,simple RNN fails to capture the context or the long term relation of the data that is it cannot remember or recall data in the input that occurred long before and hence cannot make an effective prediction.

3.2.2 VANISHING GRADIENT PROBLEM IN RNN :

The vanishing gradient problem occurs when the backpropagation algorithm moves back through all of the neurons of the neural net to update their weights. The nature of recurrent neural networks means that the cost function computed at a deep layer of the neural net will be used to change the weights of neurons at shallower layers. The mathematics that computes this change is multiplicative, which means that the gradient calculated in a step that is deep in the neural network will be multiplied back through the weights earlier in the network. The actual factor that is multiplied through the Neural network system in the backpropagation algorithm is referred to by the mathematical variable Wrec. This Wrec creates a vanishing gradient and an exploding gradient problem, when its very small or very large respectively. To summarize, the vanishing gradient problem is caused by the multiplicative nature of the backpropagation algorithm. To solve this we move towards the LSTM approach.

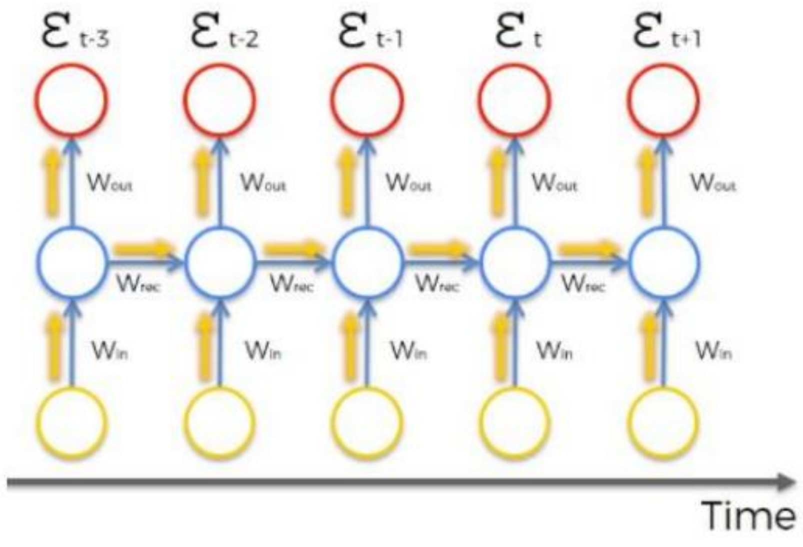


Fig 5. Vanishing Gradient Problem in RNNs

<u>3.2.3 LSTM Propagation</u>:

Long Short-Term Memory (LSTM) networks are a better version of RNN. They can remember the past data easily by resolving the vanishing gradient problem. LSTM uses back propagation to train the model, its well-suited for predictions and classifications of data sequences of unknown durations. The Sequence-to-sequence learning of LSTM is a training model that can convert sequences of one input domain into the sequences of another output domain. It is generally used when the input and output of a model can be of variable lengths. It is a method of encoder-decoder based machine translation that maps an input of sequence to an output of sequence with a tag and attention value.

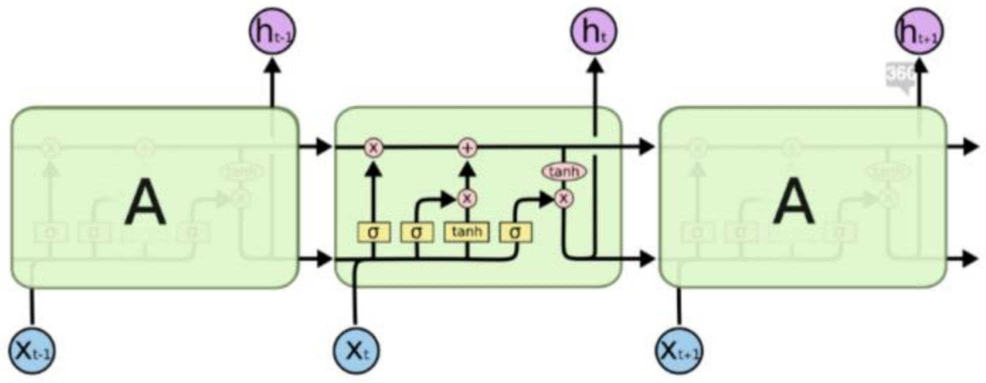


Fig 6 : LSTM Propagation

3.2.4 Forward and Backward Propagation in LSTM:

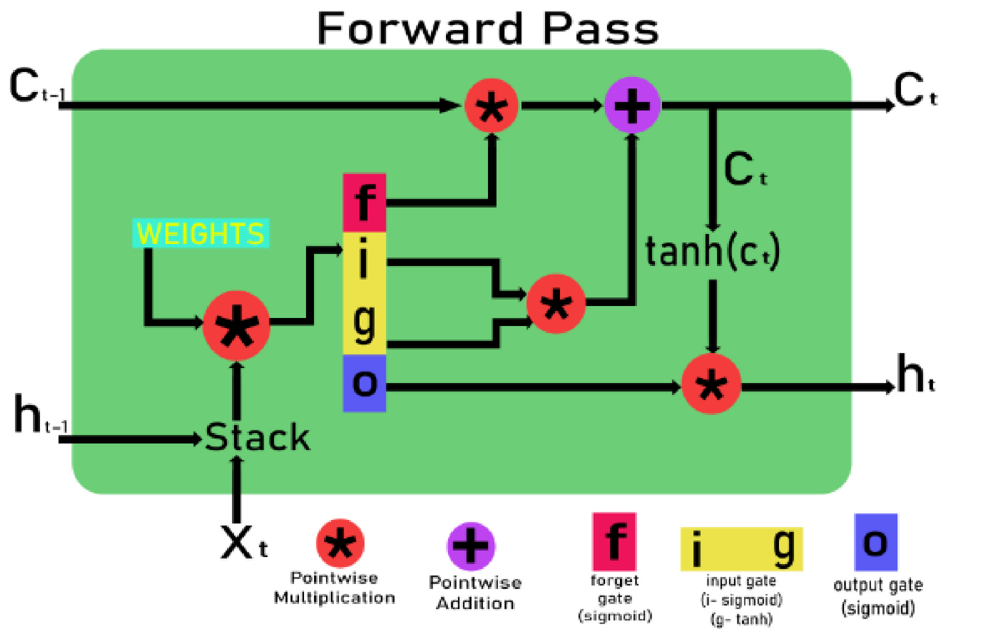


Fig 7 : Forward Pass of LSTM propagation

Let the input at time t in the LSTM cell be x_t, the cell state from time t-1 and t be c_{t-1} and c_t and the output for time t-1 and t be h_{t-1} and h_t . The initial value of c_t and h_t at t = 0 will be zero.

Step 1 : Initialization of the weights .

Weights for different gates are :

Input gate : w_{xi}, w_{xg}, b_i, w_{hj}, w_g , b_g

Forget gate : w_{xf}, b_f, w_{hf}

Output gate : w_{xo}, b_o, w_{ho}

Step 2 : Passing through different gates .

Inputs: x_t and h_{t-i} , c_{t-1} are given to the LSTM cell

Passing through input gate:

$Z_g = w_{xg} * x + w_{hg} * h_{t-1} + b_g$
$g = \tanh(Z_g)$
$Z_j = w_{xi} * x + w_{hi} * h_{t-1} + b_i$
$i = \text{sigmoid}(Z_i)$

Input_gate_out = g*i

Passing through forget gate:

$Z_f = w_{xf} * x + w_{hf} * h_{t-1} + b_f$
$f = \text{sigmoid}(Z_f)$

Forget_gate_out = f

Passing through the output gate**:**

$Z_o = w_{xo} * x + w_{ho} * h_{t-1} + b_o$
$o = \text{sigmoid}(z_O)$

Out_gate_out = o

Step 3 : Calculating the output h_t and current cell state $c_{t.}$
Calculating the current cell state c_t :
$c_{t\,=}$ (c_{t-1} * forget_gate_out) + input_gate_out

Calculating the output gate ht:

h_t=out_gate_out * tanh(ct)

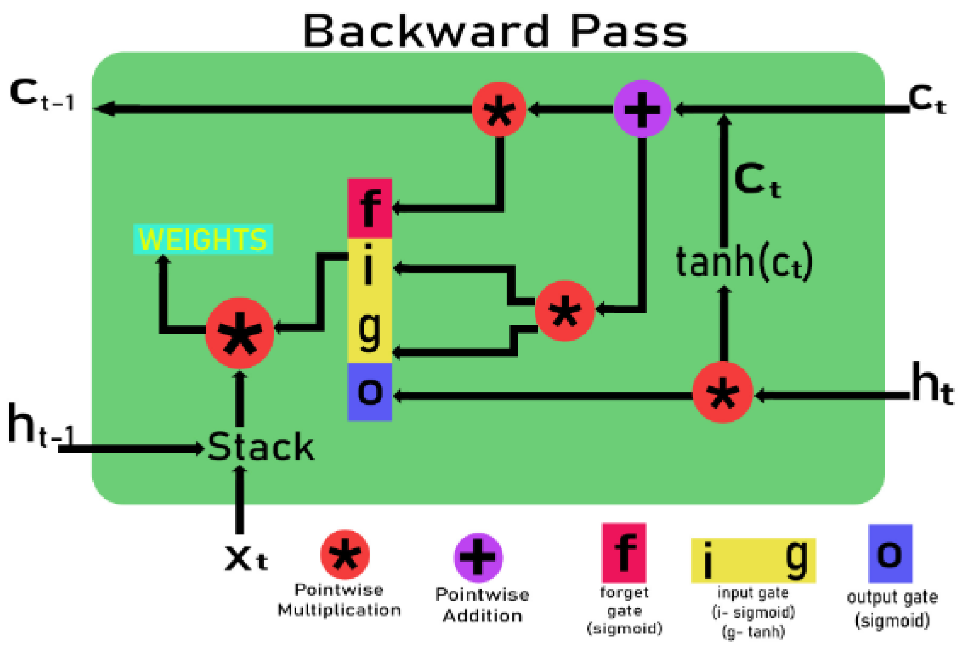


Fig 8 : Backward Pass of LSTM Propagation

Let the gradient pass down by the above cell be:

E_delta = dE/dh_t

If we are using MSE (mean square error)for error then,

E_delta=(y-h(x))

Here y is the original value and h(x) is the predicted value.

Gradient with respect to output gate

$dE/do = (dE/dh_t) * (dh_t/do)$ = E_delta * (dh_t / do)

dE/do = E_delta * $\tanh(c_t)$

Gradient with respect to c_t

$dE/dc_t = (dE / dh_t)*(dh_t/dc_t)$= E_delta *$(dh_t/dc_t)$

dE/dc_t = E_delta * o * $(1-\tanh^2(c_t))$

Gradient with respect to input gate dE/di, dE/dg dE/di = (dE/di) * (dc_t / di)

dE/di = E_delta * o * (1-$\tanh^2$ (c_t)) * g

Similarly,

dE/dg = E_delta * o * (1-$\tanh^2$ (c_t)) * i

Gradient with respect to forget gate

dE/df = E_delta * (dE/dc_t) * (dc_t / dt) t

dE/df = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1}

Gradient with respect to c_{t-1}

dE/dc_t = E_delta * (dE/dc_t) * (dc_t / dc_{t-1})

dE/dc_t = E_delta * o * (1-$\tanh^2$ (c_t)) * f

Gradient with respect to output gate weights:

dE/dw_{xo} = dE/d_o *(d_o/dw_{xo}) = E_delta * tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o) * x_t

dE/dw_{ho} = dE/do *(do/dw_{ho}) = E_delta * tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o) * h_{t-1}

dE/db_o = dE/do *(do/db_o) = E_delta * tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)

Gradient with respect to forget gate weights:

dE/dw_{xf} = dE/df *(df/dw_{xf}) = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * x_t

dE/dw_{hf} = dE/df *(df/dw_{hf}) = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * h_{t-1}

dE/db_o = dE/df *(df/db_o) = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)

Gradient with respect to input gate weights:

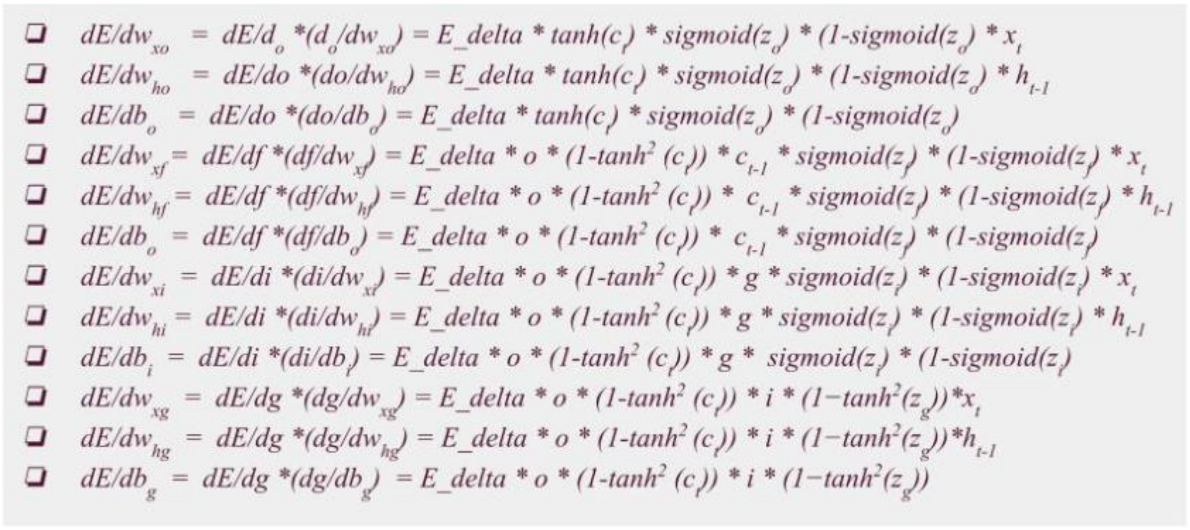
dE/dw_{xi} = dE/di *(di/dw_{xi}) = E_delta * o * (1-$\tanh^2$ (c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * x_t

dE/dw_{hi} = dE/di *(di/dw_{hi}) = E_delta * o * (1-$\tanh^2$ (c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * h_{t-1}

dE/db_i = dE/di *(di/db_i) = E_delta * o * (1-$\tanh^2$ (c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)

dE/dw_{xg} = dE/dg *(dg/dw_{xg}) = E_delta * o * (1-$\tanh^2$ (c_t)) * i * (1?$\tanh^2$(z_g))*x_t
dE/dw_{hg} = dE/dg *(dg/dw_{hg}) = E_delta * o * (1-$\tanh^2$ (c_t)) * i * (1?$\tanh^2$(z_g))*h_{t-1}
dE/db_g = dE/dg *(dg/db_g) = E_delta * o * (1-$\tanh^2$ (c_t)) * i * (1?$\tanh^2$(z_g))

Finally the gradients associated with the weights are,



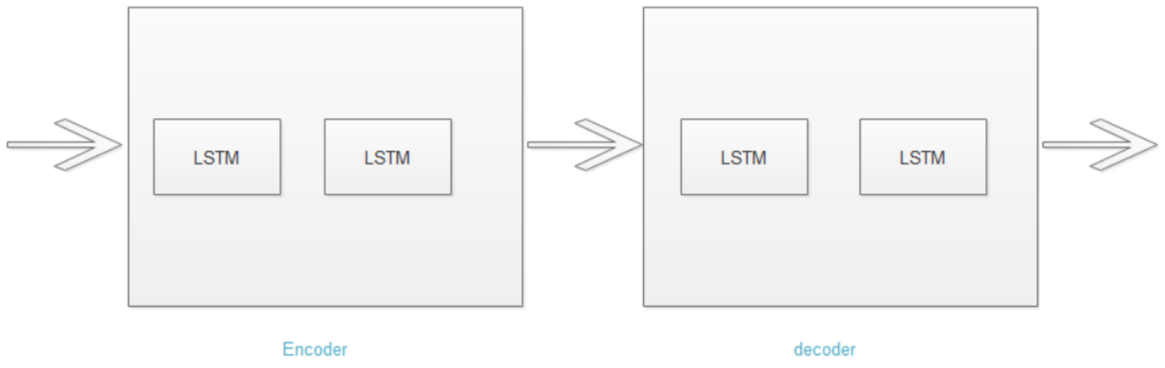
- ❑ dE/dw_{xo} = dE/d_o *(d_o/dw_{xo}) = E_delta * tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o) * x_t
- ❑ dE/dw_{ho} = dE/do *(do/dw_{ho}) = E_delta * tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o) * h_{t-1}
- ❑ dE/db_o = dE/do *(do/db_o) = E_delta * tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)
- ❑ dE/dw_{xf}= dE/df *(df/dw_{xf}) = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * x_t
- ❑ dE/dw_{hf}= dE/df *(df/dw_{hf}) = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * h_{t-1}
- ❑ dE/db_o = dE/df *(df/db_o) = E_delta * o * (1-$\tanh^2$ (c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)
- ❑ dE/dw_{xi} = dE/di *(di/dw_{xi}) = E_delta * o * (1-$\tanh^2$ (c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * x_t
- ❑ dE/dw_{hi}= dE/di *(di/dw_{hi}) = E_delta * o * (1-$\tanh^2$ (c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * h_{t-1}
- ❑ dE/db_i = dE/di *(di/db_i) = E_delta * o * (1-$\tanh^2$ (c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)
- ❑ dE/dw_{xg} = dE/dg *(dg/dw_{xg}) = E_delta * o * (1-$\tanh^2$ (c_t)) * i * (1−$\tanh^2$(z_g))*x_t
- ❑ dE/dw_{hg} = dE/dg *(dg/dw_{hg}) = E_delta * o * (1-$\tanh^2$ (c_t)) * i * (1−$\tanh^2$(z_g))*h_{t-1}
- ❑ dE/db_g = dE/dg *(dg/db_g) = E_delta * o * (1-$\tanh^2$ (c_t)) * i * (1−$\tanh^2$(z_g))

Fig 9 : Gradients with weights LSTM Backpropagation

Using all gradient, we can easily update the weights associated with input gate, output gate, and forget gate.

3.2.5 Encoder-Decoder Architecture:

The Sequence to Sequence model uses a method of encoder-decoder based machine translation. Encoder-Decoder architecture is used in predicting sequences when the length of output and input data may vary. The input sequence is read entirely by the encoder and a fixed length internal representation is generated. The internal representation captures the entire context of the input data sequence. The decoder network uses this internal representation to predict the output words until the end of the sequence token is reached.



Encoder:

An encoder is an LSTM network which reads the entire input sequence. At each time step, one word from the input sequence is read by the encoder. It then processes the input at each time step and captures the context and the key information related to the input sequence. It takes each word of input(x) and generates the hidden state output (h) and the cell state which is an internal state(c). The hidden state(hi) and cell state(ci) of the last time step are the internal representation of the complete input sequence which will be use to initialize the decoder.

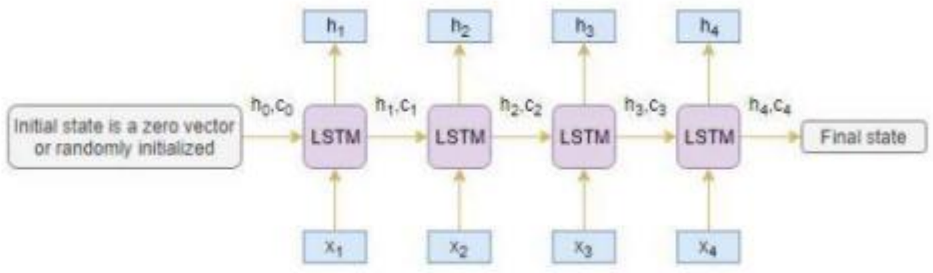


Fig 10 : Encoder Architecture

Decoder :

The decoder is also an LSTM network. It reads the entire internal representation generated by the encoder one word at a time step. It then predicts the same sequence offset by one time step. The decoder is trained to predict the next word in the output sequence given the previous word based on the contextual memory stored by the LSTM architecture. Two special tokens and are added at the beginning and at the end of the target sequence before feeding it to the decoder. We start predicting the target sequence by passing one word at a time. The first word of output of the decoder is always token. The end of the output sequence is represented by token.

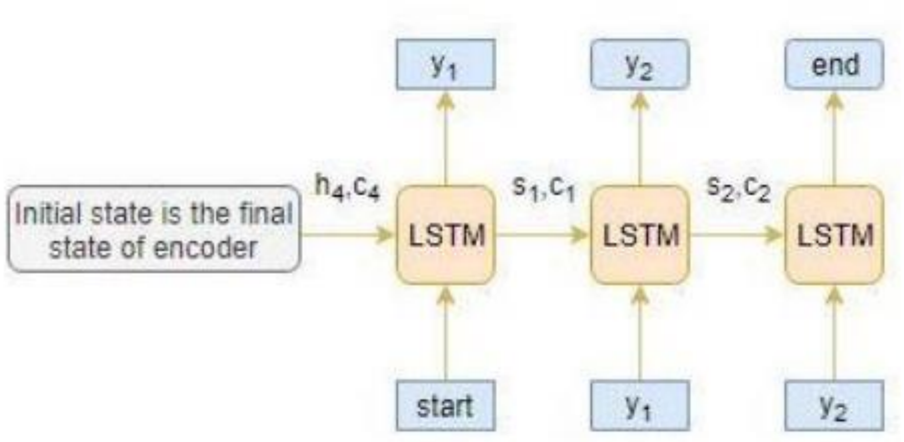


Fig 11 : Decoder Architecture

Combined Structure :

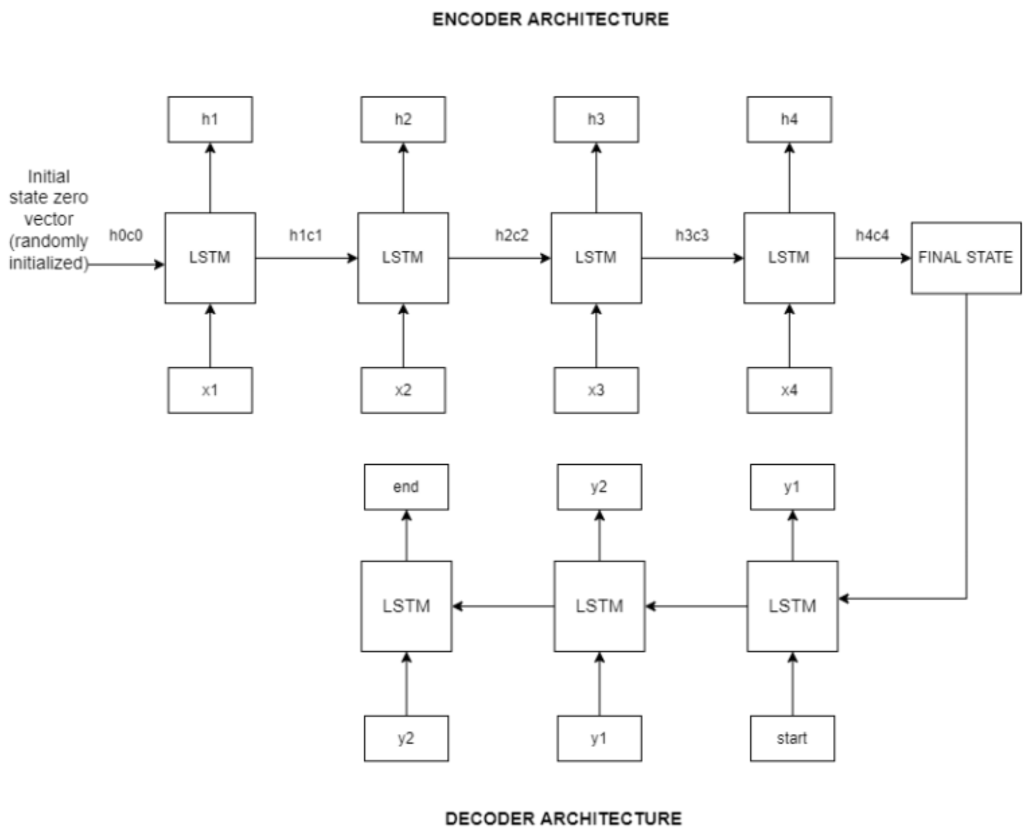


Fig 12 : Combined Encoder-Decoder Architecture [Seq2Seq model]

3.2.6 Attention Mechanism:

A Sequence to Sequence model with an attention mechanism consists of encoder, decoder and an attention layer. Attention mechanism is used to secure individual parts of the input which are more important at that particular time. It can be implemented by taking inputs from each time steps and giving weightage to time steps. The weightage depends on the contextual importance of that particular time step. It helps pay attention to the most relevant parts of the input data sequence so that the decoder can optimally generate the next word in the output sequence.

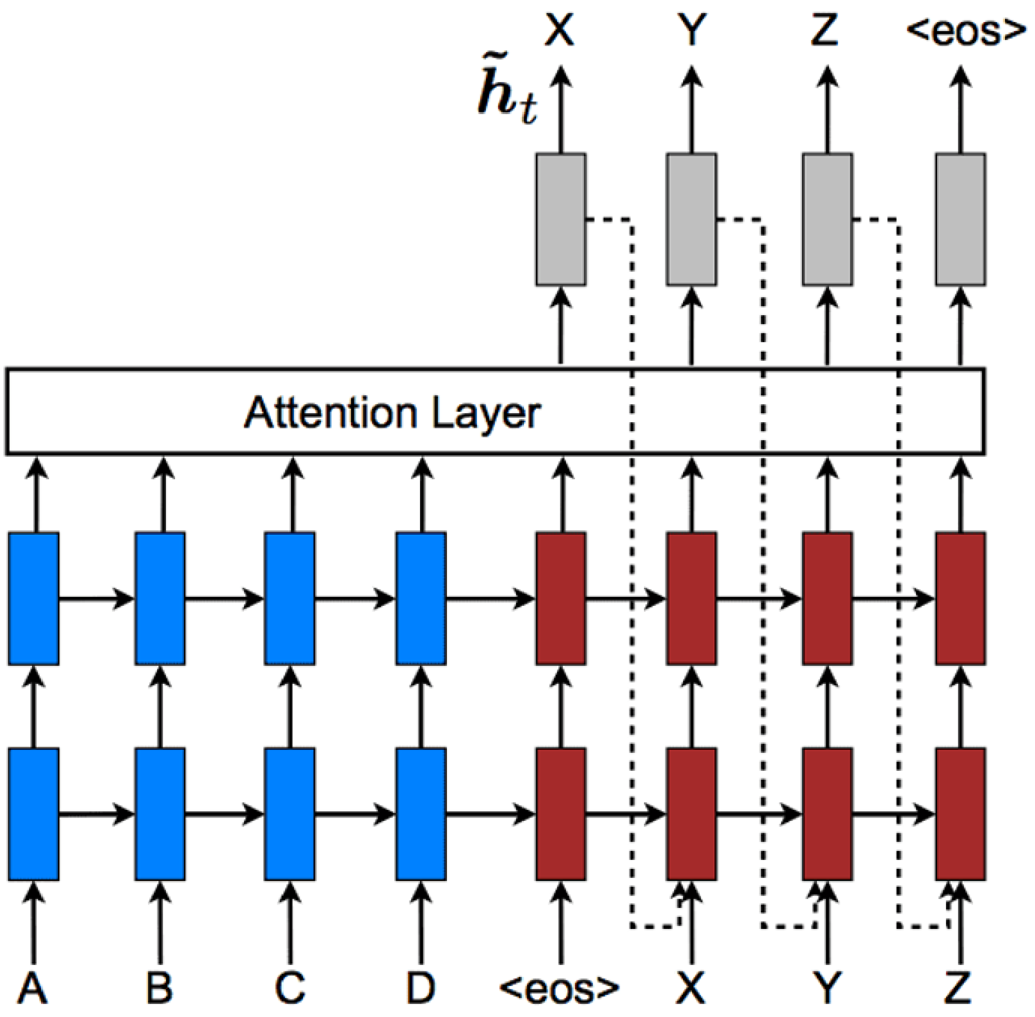


Fig 13 : Attention Layer Architecture

3.3 Implementation and Procedure:

Let us now dive deep into the entire procedure for setting up our dataset and training our model and then testing it using test set.

3.3.1 Data Preprocessing:

Dataset Used : Hindustan Times News Reports along with headlines(and other details) from 2015-2017.

Performing basic pre-processing steps is very important before we get to the model building part. Using messy and uncleaned text data is a potentially disastrous move. So, we will drop all the unwanted symbols, characters, etc. from the text that do not affect the objective of our problem.

We will perform the below preprocessing tasks for our data:

• Convert everything to lowercase
• Remove apostrophes (‘s)
• Remove any text inside the parenthesis ( ),{},[].
• Eliminate punctuations and special characters
• Remove stopwords
• Remove short words, abbreviations, colloquial words which have no impact on the summary.

3.3.2 Methodology:

- Summary Cleaner and Text Cleaner are two functions being used to clear out the text file and the summary file for clean and feedable data for Deep Learning model.
- We use plots ,wordclouds and percentile coverage for max lengths and frequencies, and demarcate max lengths as needed.
- We do test-train split and tokenize our training and testing data, tokenized data fed into encoder layer which again goes into subsequent layers for finally be decoded.
- We concatenate attention layer with decoder layer output, setup dense layer, feed our training data and start training.
- **For training** :
  Total Trainable Params = 92,013,409 , Batch Size = 128, Epochs = 20

- We use seq2summary and seq2text functions to convert the sequences to predicted summaries.
- We analyze our prediction through graphs and ROUGE scores.

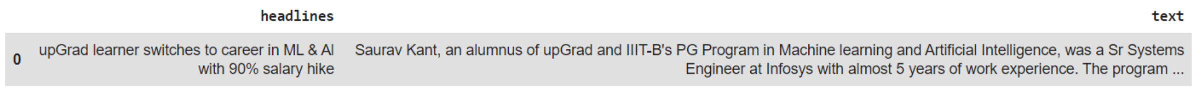
Chapter 4
RESULTS AND ANALYSIS

This section will go on to explain the datasets, data tuning processes, training-testing regimes, results and analysis of findings.

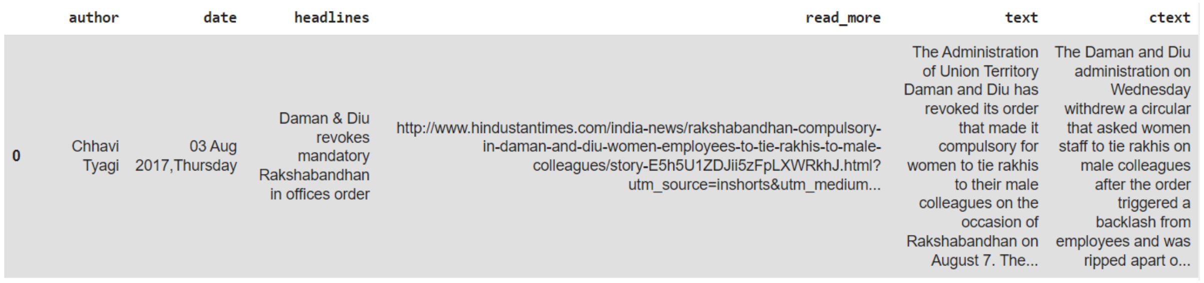
4.1 Dataset:

The dataset we use to train the said model, is made as a combination of two CSV files ,comprising news articles and their headlines which is a summary of the article from Hindustan Times newspaper from 2015-2017. The dataset is taken from Kaggle, an open source website for databases.

Out of the two databases, one of them only has the articles and headlines and the other one has headlines, article, writer, link etc. We merge the article details all together to get comparable datasets.

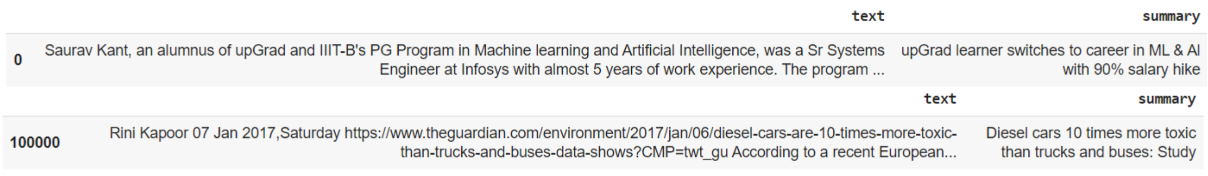


	headlines	text
0	upGrad learner switches to career in ML & Al with 90% salary hike	Saurav Kant, an alumnus of upGrad and IIIT-B's PG Program in Machine learning and Artificial Intelligence, was a Sr Systems Engineer at Infosys with almost 5 years of work experience. The program ...



	author	date	headlines	read_more	text	ctext
0	Chhavi Tyagi	03 Aug 2017,Thursday	Daman & Diu revokes mandatory Rakshabandhan in offices order	http://www.hindustantimes.com/india-news/rakshabandhan-compulsory-in-daman-and-diu-women-employees-to-tie-rakhis-to-male-colleagues/story-E5h5U1ZDJii5zFpLXWRkhJ.html?utm_source=inshorts&utm_medium...	The Administration of Union Territory Daman and Diu has revoked its order that made it compulsory for women to tie rakhis to their male colleagues on the occasion of Rakshabandhan on August 7. The...	The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie rakhis on male colleagues after the order triggered a backlash from employees and was ripped apart o...

Table 2 : One row of data from both csv files before merging



	text	summary
0	Saurav Kant, an alumnus of upGrad and IIIT-B's PG Program in Machine learning and Artificial Intelligence, was a Sr Systems Engineer at Infosys with almost 5 years of work experience. The program ...	upGrad learner switches to career in ML & Al with 90% salary hike

	text	summary
100000	Rini Kapoor 07 Jan 2017,Saturday https://www.theguardian.com/environment/2017/jan/06/diesel-cars-are-10-times-more-toxic-than-trucks-and-buses-data-shows?CMP=twt_gu According to a recent European...	Diesel cars 10 times more toxic than trucks and buses: Study

Table 3 : Row 0th and Row 100000th after merging into same dataset

Number of rows in dataset before cleaning : 1,02,916

After Cleaning, Parameterization, Tokenization and test-train split the total number of trainable params becomes 92,013,409.

4.2 PreProcessing and Analysis:

4.2.1 Machine Specification and Software Used:

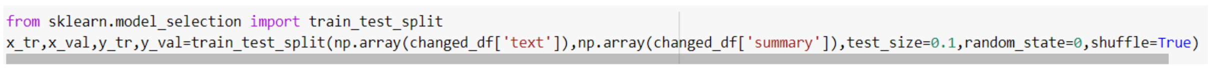
The coding is done in Python 3.7 , Google collab notebooks were used that run in the cloud and are highly integrated with google drive for easy access and sharing ofresources and faster implementation. Google Collabs also has artificial CPU and GPU provided by google, collab has this special feature of enabling special GPU so it fastens the training procedure.

All the software/model extensions needed for the code :

- Numpy -Dealing with numbers
- Pandas – Creating Dataframes for huge dataset
- Tensorflow – Keras – For dealing with model, feeding data
- Beautiful Soup – For Data cleaning
- NLTK – Getting stopwords library and cleaning purposes
- Matplotlib – Graph plots for analysis
- WordCloud – Wordclouds for frequency analysis
- SkLearn – For Test-Train split
- Keras – Backend – Making the Model ready for training

4.2.2 Training-Testing Regime:

The train test split is done using sklearn module, The split size is 90-10 for training and testing data.

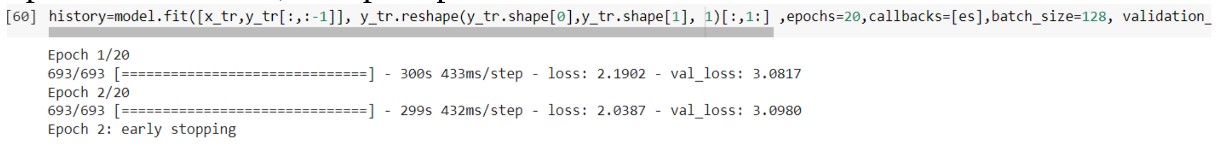


```
from sklearn.model_selection import train_test_split
x_tr,x_val,y_tr,y_val=train_test_split(np.array(changed_df['text']),np.array(changed_df['summary']),test_size=0.1,random_state=0,shuffle=True)
```

Fig 14: Test-Train split

Epochs : The basic epoch value was taken at 20, but the training didn't take 20 epochs to reach lowest loss value and got early stoppage because of a function in Keras module.

<u>Batch-Size :</u> The Batch-size was taken as 512 per epoch but the training took quite an amount of time and by reducing it by 2 times everytime we reach our optimal batch size, 128 per epoch.



```
[60] history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=20,callbacks=[es],batch_size=128, validation_

     Epoch 1/20
     693/693 [==============================] - 300s 433ms/step - loss: 2.1902 - val_loss: 3.0817
     Epoch 2/20
     693/693 [==============================] - 299s 432ms/step - loss: 2.0387 - val_loss: 3.0980
     Epoch 2: early stopping
```

Fig 15 : Fixing Epoch and Batch-size

<u>4.2.3 Graph Analysis :</u>

A frequency distribution is a representation, either in a graphical or tabular format, that displays the number of observations within a given interval. The interval size depends on the data being analyzed and the goals of the analyst. The intervals must be mutually exclusive and exhaustive.

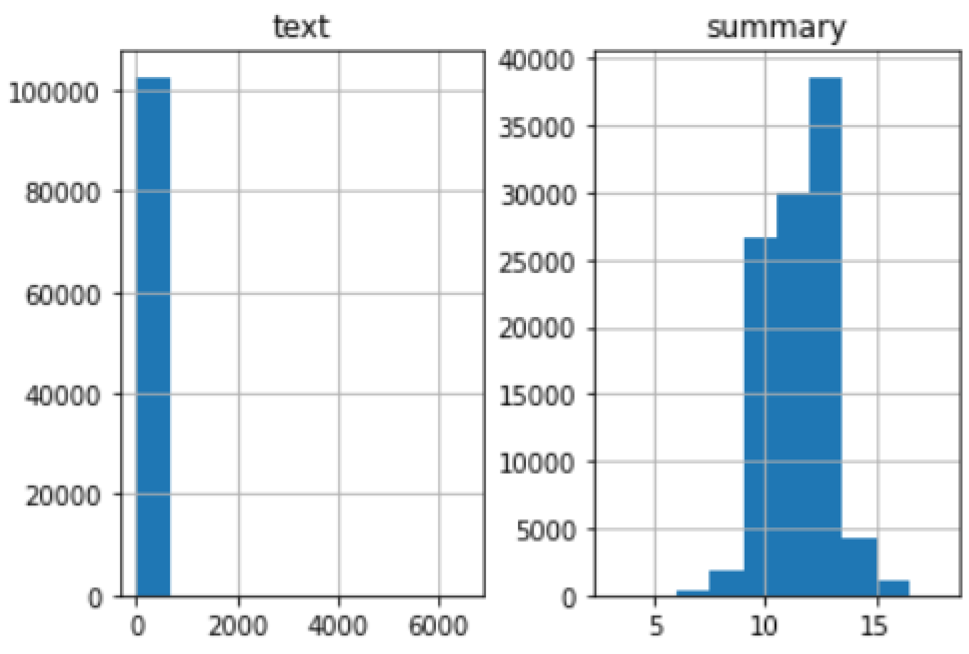


Fig 16 : Frequency distribution of text and summary in database

We also did a wordcloud to analyse frequency distribution.

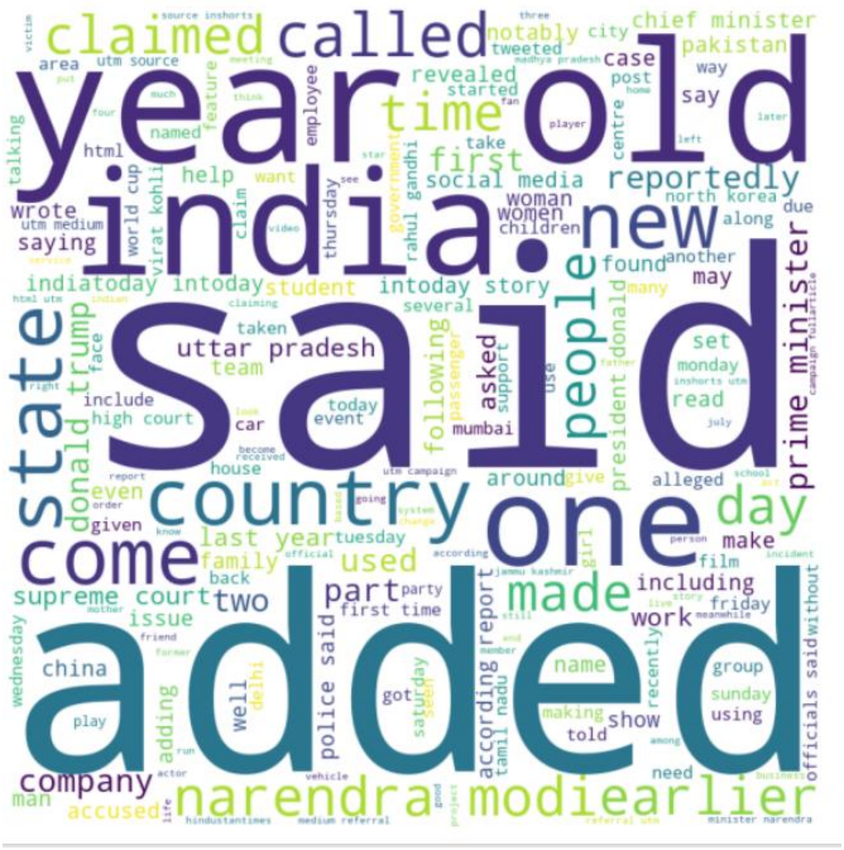


Fig 17 **:** Wordcloud for Cleaned text file in database

A learning curve is a plot of model learning performance over experience or time. Learning curves are a widely used diagnostic tool in machine learning for algorithms that learn from a training dataset incrementally. The model can be evaluated on the training dataset and on a hold out validation dataset after each update during training and plots of the measured performance can created to show learning curves.

Reviewing learning curves of models during training can be used to diagnose problems with learning, such as an underfit or overfit model, as well as whether the training and validation datasets are suitably representative.

```
] from matplotlib import pyplot
  pyplot.plot(history.history['loss'], label='train')
  pyplot.plot(history.history['val_loss'], label='test')
  pyplot.legend()
  pyplot.show()
```

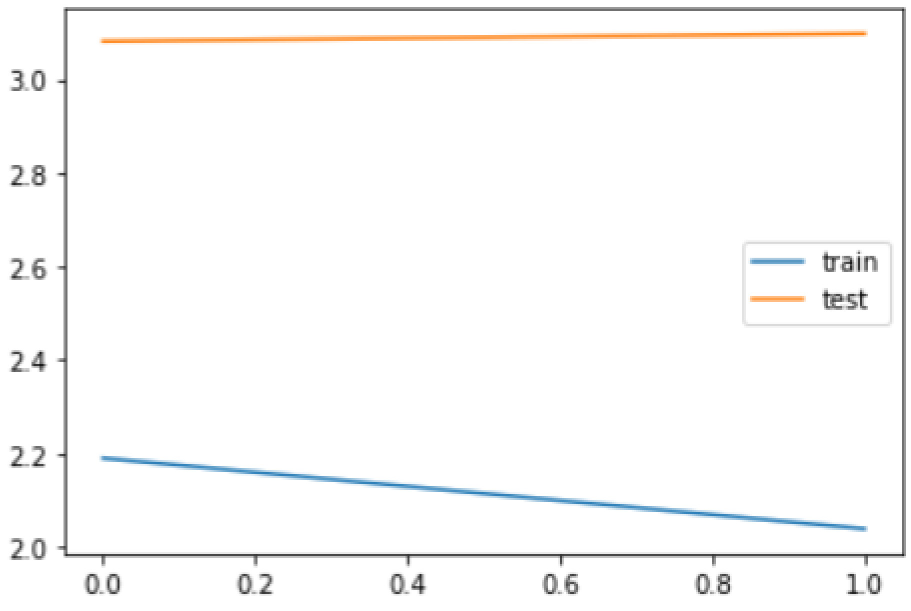


Fig 18 : Test-Train Plot

Results :

```
Review: first indian budget presented founder economist magazine james wilson february w
Original summary: who was the man behind india first ever budget in
Predicted summary:  who is the first indian to receive budget


Review: former executive uber travis vanderzanden scooter startup bird raised million se
Original summary: ex uber exec scooter startup bird raises million
Predicted summary:  ex uber exec startup gets mn from startup
```

Fig 19 : Summarization results examples

4.3 Analysing the Results:

In a general text summarization model, we use ROUGE as the validation metric. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is used for evaluating automatic text summarizations and machine translations. It consists of a set of metrics which are used for evaluation. It compares the result automatically produced using the model by the machine against human produced results. In the text summarization model, the human produced summaries which are used for reference are reference summaries and the summary generated using the model are called system summaries. The model is evaluated based on the extent to which the system summaries are similar to the reference summaries. This is captured by recall. Here, Recall means how much the system summary is able to capture or recover the essence of reference summary.

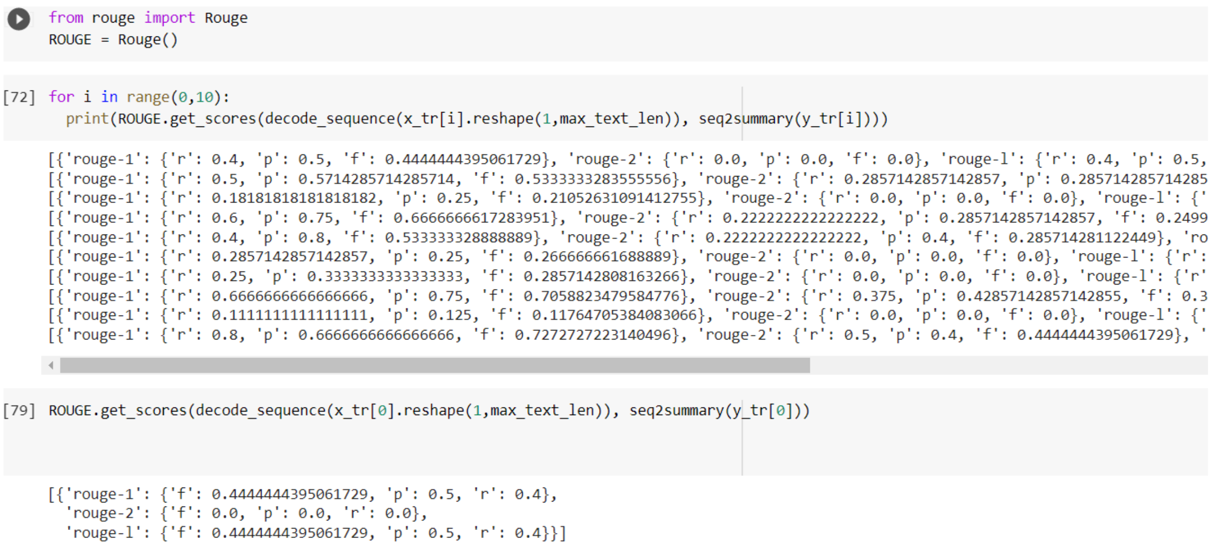
If we are just considering the individual words, it can be computed as: (Number_of_overlapping_words) ÷ (Total_words_in_reference_summary)

Also there's precision value which comes as : (Number_of_overlapping_words) ÷ (Total_words_in_predicted_summary)

ROUGE-1 refers to overlap of unigrams between the system summary and reference summary.

ROUGE-2 refers to the overlap of bigrams between the system and reference summaries.

ROUGE-N measures unigram, bigram, trigram and higher order n-gram overlap.



```
from rouge import Rouge
ROUGE = Rouge()
```

```
[72] for i in range(0,10):
       print(ROUGE.get_scores(decode_sequence(x_tr[i].reshape(1,max_text_len)), seq2summary(y_tr[i])))
```

```
[{'rouge-1': {'r': 0.4, 'p': 0.5, 'f': 0.4444444395061729}, 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0}, 'rouge-l': {'r': 0.4, 'p': 0.5,
[{'rouge-1': {'r': 0.5, 'p': 0.5714285714285714, 'f': 0.5333333283555556}, 'rouge-2': {'r': 0.2857142857142857, 'p': 0.285714285714285
[{'rouge-1': {'r': 0.18181818181818182, 'p': 0.25, 'f': 0.21052631091412755}, 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0}, 'rouge-l': {'
[{'rouge-1': {'r': 0.6, 'p': 0.75, 'f': 0.6666666617283951}, 'rouge-2': {'r': 0.2222222222222222, 'p': 0.2857142857142857, 'f': 0.2499
[{'rouge-1': {'r': 0.4, 'p': 0.8, 'f': 0.533333328888889}, 'rouge-2': {'r': 0.2222222222222222, 'p': 0.4, 'f': 0.285714281122449}, 'ro
[{'rouge-1': {'r': 0.2857142857142857, 'p': 0.25, 'f': 0.266666661688889}, 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0}, 'rouge-l': {'r':
[{'rouge-1': {'r': 0.25, 'p': 0.3333333333333333, 'f': 0.2857142808163266}, 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0}, 'rouge-l': {'r'
[{'rouge-1': {'r': 0.6666666666666666, 'p': 0.75, 'f': 0.7058823479584776}, 'rouge-2': {'r': 0.375, 'p': 0.42857142857142855, 'f': 0.3
[{'rouge-1': {'r': 0.1111111111111111, 'p': 0.125, 'f': 0.11764705384083066}, 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0}, 'rouge-l': {'
[{'rouge-1': {'r': 0.8, 'p': 0.6666666666666666, 'f': 0.7272727223140496}, 'rouge-2': {'r': 0.5, 'p': 0.4, 'f': 0.4444444395061729}, '
```

```
[79] ROUGE.get_scores(decode_sequence(x_tr[0].reshape(1,max_text_len)), seq2summary(y_tr[0]))
```

```
[{'rouge-1': {'f': 0.4444444395061729, 'p': 0.5, 'r': 0.4},
  'rouge-2': {'f': 0.0, 'p': 0.0, 'r': 0.0},
  'rouge-l': {'f': 0.4444444395061729, 'p': 0.5, 'r': 0.4}}]
```

Fig 20 : ROUGE scores for first 10 summarizations

The average F-score ROUGE value for first 10 summarizations is 0.449

The ‘f’, ‘p’ and ‘r’ written above symbolizes to F-score, Precision and Recall of the results.

4.3.1 Error Case Analysis:

The results we received from the model isn’t perfect, in many cases the ROUGE score is around 0-10% which is very less to be considered as a summary. The accuracy of the model is not very high considerably but that’s where we would be working more in the future.

Chapter 5
CONCLUSION

5.1 Inference:

The International Data Corporation (IDC) projects that the total amount of digital data circulating annually around the world would sprout from 4.4 zettabytes in 2013 to hit 180 zettabytes in 2025.
The increasing growth of the Internet has made a huge amount of information available. It is difficult for humans to summarize large amounts of text. Thus, there is an immense need for automatic summarization tools in this age of information overload. That’s a huge amount of data circulating in the digital world. There is a dire need of algorithms which can be used to automatically shorten the amount of data with accurate summaries that capture the essence of the intended messages.
Furthermore, applying text summarization reduces reading time and accelerates the process of researching for information playing a major role in current era of rapid development and digitization. Humans are generally quite good at this task as we have the capacity to understand the meaning of a text document and extract salient features to summarize the documents using our own words. However, automatic methods for text summarization are crucial in today’s world where there is an over-abundance of data and lack of manpower as well as time to interpret the data.

5.2 Future Work:

In case to make this model more effective i.e of higher ROUGE scores will have to study in detail and implement Text Summarization using B.E.R.T(Bidirectional Encoder Representations from Transformers) and compare it with G.P.T 2(Generative Pretrained Transformer 2) model. Have to search for a new bigger Data-set to train the model with the help of our data set and check the accuracy of the results. Study and try to implement about new machine translation models like B.E.R.T & GPT-2 model.

5.3 REFERENCES:

1. Fattah MA (2014) A hybrid machine learning model for multi-document summarization. 592–600. doi:10. 1007/s10489-013-0490-0
2. Fattah MA, Ren F (2009) GA, MR, FFNN, PNN and GMM based models for automatic text summarization. Comput Speech Lang 23:126–144. doi:10.1016/j.csl.2008.04.002
3. Mendoza M, Bonilla S, Noguera C et al (2014) Extractive single-document summarization based on genetic operators and guided local search. Expert Syst Appl 41:4158–4169. doi:10.1016/j.eswa.2013.12.042
4. 32. J. Morris, G. Hirst, "Lexical cohesion computed by thesaural relations as an indicator of the structure of text," Journal of Computational Linguistics, vol. 17(1), ACM, 1999, pp. 21–48.
5. Baralis E, Cagliero L, Mahoto N, Fiori A (2013) GRAPHSUM : discovering correlations among multiple terms for graph-based summarization. Inf Sci 249:96–109. doi:10.1016/j.ins.2013.06.046
6. Erkan G, Radev D (2004) LexRank: graph-based lexical centrality as salience in text summarization. J Artif Intell Res 22:457–479
7. D. Wang, T. Li, S. Zhu, C. Ding, "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization," In Proceedings of the 31th Annual International ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval, 2008. ACM, pp. 307–314.
8. T. K. Landauer, P. W. Foltz, D. Laham, "Introduction to latent semantic analysis," Journal of Discourse Processes, vol. 25(2–3), 1998, pp. 259– 284.
9. B. Rosario, "Latent semantic indexing: An overview," Technical Report of INFOSYS 240, University of California, 2000.
10. T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," Journal of Machine Learning, vol. 42(1–2), 2001, pp. 177– 196.
11. M. Simina, C. Barbu, "Meta latent semantic analysis" In IEEE International conference on Systems, Man and Cybernetics, IEEE, 2004, pp. 3720–3724
12. J. T. Chien. "Adaptive Bayesian latent semantic analysis" IEEE Transactions on Audio, Speech, and Language Processing, vol. 16(1), 2008, pp.198–207
13. J. H. Lee, S. Park, C. M. Ahn, D. Kim, "Automatic generic document summarization based on non-negative matrix factorization" Journal on Information Processing and Management, vol. 45(1), 2009, pp. 20–34.
14. V. Snasel, P. Moravec, J. Pokorny, "Using semi-discrete decomposition for topic identification," In Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications, IEEE, 2008, pp. 415–420.
15. T. G. Kolda, D. P. O'Leary, "Algorithm 805: Computation and uses of the semi-discrete matrix decomposition," Transactions on Mathematical Software, vol. 26(3), 2000, pp. 415–435.
16. Ko Y, Seo J (2008) An effective sentence-extraction technique using contextual information and statistical approaches for text summarization. Pattern Recognit Lett 29:1366–1371. doi:10.1016/j.patrec.2008.02. 008
17. Manning CD, Raghavan P, Schtze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge.

18. Text Summarization using Deep Learning by Kasimahanthi Divya, Kambala Sneha, Baisetti Sowmya, G Sankara Rao- International Research Journal of Engineering and Technology (IRJET) - Volume: 07 Issue: 05 | May 2020.

19. Machine Learning Approach for Automatic Text Summarization Using Neural Networks by Meetkumar Patel , Adwaita Chokshi , Satyadev Vyas , Khushbu Maurya- International Journal of Advanced Research in Computer and Communication Engineering ISO 3297:2007 Certified Vol. 7, Issue 1, January 2018.

20. Recent automatic text summarization techniques: a survey by Mahak Gambhir · Vishal Gupta - Published online: 29 March 2016 © Springer Science+Business Media Dordrecht 2016.

21. U. Hahn, I. Mani, "of Automatic Researchers are investigating summarization tools and methods that", IEEE Computer33. 11, pp. 29- 36, November 2000.

22. E. Baralis, L. Cagliero, A. Fiori, P. Garza, "MWI-Sum: A Multilingual Summarizer Based on Frequent Weighted Itemsets", ACM Transactions on Information Systems, vol. 34, no. 1, pp. 1-35, 2015

23. K. Spärck Jones, "Automatic summarizing: The state of the art", Information Processing & Management, vol. 43, pp. 1449-1481, nov 200