COMPARATIVE ANALYSIS OF AUTOMATIC TEXT SUMMARIZERS

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Computer Science and Engineering/Information Technology

by

Janvijay Singh Bisht (151308)

Under the supervision of

Mr. Nitin Kumar

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat

Solan-173234, Himachal Pradesh

`

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"COMPARATIVE ANALYSIS OF AUTOMATIC TEXT SUMMARIZERS"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2018 to December 2018 under the supervision of **Mr. Nitin Kumar,** Assistant Professor, Department of Computer Science & Engineering and Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Janvijay Singh Bisht
(151308)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Mr. Nitin Kumar
Assistant Professor
Computer Science & Engineering and Information Technology
Dated:

ACKNOWLEDGEMENT

I would like to present our heartfelt gratitude to our mentor and guide **Mr. Nitin Kumar, Assistant Professor, Computer Science & Engineering and Information Technology** for giving invaluable intellectual inputs and suggestions right from choosing the appropriate project for commencement to its fruitful culmination.

I would also like to take this opportunity to offer our sincerest thanks to **Dr. S.P. Ghrera**, **Head, Computer Science & Engineering and Information Technology** for giving us the opportunity to work under his guidance in Jaypee University of Information Technology, Waknaghat, enabling us to gain an immensely enriching professional experience. I would also like to thank him for his endless support and relentless supervision without which this could not have been possible.

I would also like extend my deep gratitude to the lab assistant, **Mr. Ravi Raina** for his endless support throughout the length of our project.

Last but not the least, the successful completion of this project would not have been possible without the consistent backing of our mentors, family and friends.

Table of Content

Table of Abbreviations

ATS	Automatic Text Summarization
CNN	Convolutional Neural Network
IDF	Inverse Document Frequency
LSTM	Long Short Term Memory
NLP	Natural Language Processing
RNN	Recursive Neural Network
TF-IDF	Term Frequency - Inverse Document Frequency
TS	Text Summarization

List of Figures

List of Graphs

List of Tables

ABSTRACT

In view of the intense time crunch faced by humans in today's time, the idea of information being presented to people in the most concise and crisp manner possible is pretty much a necessity. This project entitled "COMPARATIVE ANALYSIS OF AUTOMATIC TEXT SUMMARIZERS" aims to summarize text documents automatically using software drawing from the concepts of machine learning and neural networks.

There are two major kinds of Automatic Text Summarization (ATS) techniques, Extractive Text Summarization and Abstractive Text Summarization. Extractive ATS refers to picking out more important sentences, ones that more central to the chief idea of the document and stringing them together to give a summarised document. Abstractive ATS, on the other hand, deals with rephrasing the sentences in the original document and expressing them differently. These algorithms are generally more difficult to implement as this challenges the system to strain itself with complex skills such as paraphrasing and abstraction.

In this project, Extractive ATS is done using lexical analysis. The results and a detailed analysis has been presented in this report. The project will later use machine learning algorithms to achieve ATS both extractive and abstractive in nature along with presenting a structured study of the difference in efficiencies among the various algorithms. I will also integrate the project to a front-end user interface that shall be user friendly and responsive.

This report is a summary of the tasks accomplished by us on the project entitled "Automatic Text Summarization and Analysis" in partial fulfilment. It contains all the details pertaining to the technical, functional and non-functional aspects of the partial completion of the project aimed at summarizing documents followed by a comprehensive comparison of the efficiency of the same. It also gives an in-depth explanation of the methodology, procedure and algorithms used, where the data is being extracted from, how it is being worked upon along with test plans for the future, models and flowcharts

CHAPTER 1: INTRODUCTION

1.1. INTRODUCTION

Natural Language Processing (NLP) is the application of computational techniques to the analysis and synthesis of natural language and speech. It refers to the ability of a machine to comprehend language as I speak. The idea behind NLP is to bridge the gap between how instructions are fed into the computer (in a precise and structured manner) and how humans actually talk, with no organized linguistic structure.

In today's time, the amount of information available to us has increased multiple folds and continues multiply exponentially. Most information, academic or otherwise is available in electronic form. Now-a-days, I face an unnatural abundance of textual data and paired with a lack of disposable time, there is an urgent need to create automated ways of presenting information in the most concise yet meaningful form. The idea is to summarize a document by shortening its length or volume without losing the meaning of the original text. The summarized text must still convey the entire idea that the original text did, but in a more concise manner.

Automatic Text Summarization is the process of creating a synopsis of a text document retaining only the more central and significant phrases or sentences to give a coherent version of the original document. The objective is to produce a synopsis by reducing either the size or volume of the original text without actually changing the meaning of it.

The process of TS may be grouped into two broad categories, Extractive and Abstractive Summarization.

Extractive Text Summarization consists of picking out important sentences and piecing them together in a relevant and consequential manner. The key component of this type of summarization is to select important statements or phrases. The prominence of a sentence is evaluated by putting it through a predefined procedure that may be syntactic and/or semantic in nature. Sentences are put through a pre-established function and the ones that explain all ideas salient to the original document are extracted as a subset of those in the original document. These sentences or phrases are then concatenated together to form the summarized document.

Abstractive Text Summarization, on the other hand, refers to actually generating a synopsis by paraphrasing, not just extracting central segments word by word. This approach

generates new statements, even substituting phrases or statements for other, more relevant ones. A sentence is reconstructed based on the semantic interconnections among the words. Creating abstractive summaries are a tougher job to accomplish, as the system must incorporate capabilities like understanding of the text and the ability to rephrase it. Difficult as they may be to implement, abstractive summarization techniques are by and large free of most restrictions and confinities that extractive methods suffer from.

The pressing need to summarize textual data has become extremely prominent in the past few years. The very little time that people have to spare, they most certainly do not wish to spend it going through extensively lengthy documents only to find a small section of the text to be relevant. Hence, an interesting and probably viable solution to the above problem is the development of a program that automatically summarizes lengthy text documents into smaller versions retaining the phrases and sentences of higher centrality and relevance.

1.2. PROBLEM STATEMENT

To design a software to automatically summarize textual documents to produce a synopsis that retains relatively more central and significant information, using Lexical analysis and different machine learning algorithms, along with a thorough comparative study of the efficiencies of the same. The software must be supported by an interactive user interface so that it is easy and convenient to use. It must be stable, time efficient and considerably accurate.

1.3. OBJECTIVES

The project entitled “Comparative Study of Automated Text Summarisers” has the following objectives:

- To summarize a text document extractively using lexical analysis.
- To realise automated text summarization with the help of various machine learning and neural network algorithms.
- To conduct a detailed in-depth comparative study of the effectiveness and productivity of the different algorithms used previously.
- To create an interactive and responsive front-end user interface to make the software more accessible and usable by different classes of user.

The project will realise these objectives by using algorithms for summarizing single document and generating summaries using extractive summarization techniques and generate headlines using extractive summarization techniques.

1.4. METHODOLOGY

In this section, the procedure adopted while undertaking this project is explained in detail, step by step. Fig 1.1 shows the plan of action for completion of the project.

- Commencement of the project begins with the identification of the problem. The difficulty of tackling TS with automation and a considerable amount of efficiency is extremely apparent. As students, I face the issue of abundance of information and with the evolution of the internet, things have only become graver. I needed a system to give us a concise outlook to all the textual data I have. Hence, figuring out the objectives our project was not a hassle.
- The next step is to carve out the problem statement and a list of objectives. Requirement Analysis is also done in this step.
- Then, I review relevant and freely available literature relating to text summarization. Various research papers are examined and looked through in the process, a review of which has been presented in this report.
- In the next phase, a structured plan to implement the functionalities and achieve the goals set for the project is formulated and a rough skeleton of the process of completion of the project is drawn out.
- The data is collected and worked upon at this point. The algorithms developed or chosen are used to process the data.
- Once irrefutable results are achieved, they are analysed and a final report is filled up in detail. This report may contain a study of the efficiency of the developed software, both in terms of time and accuracy.

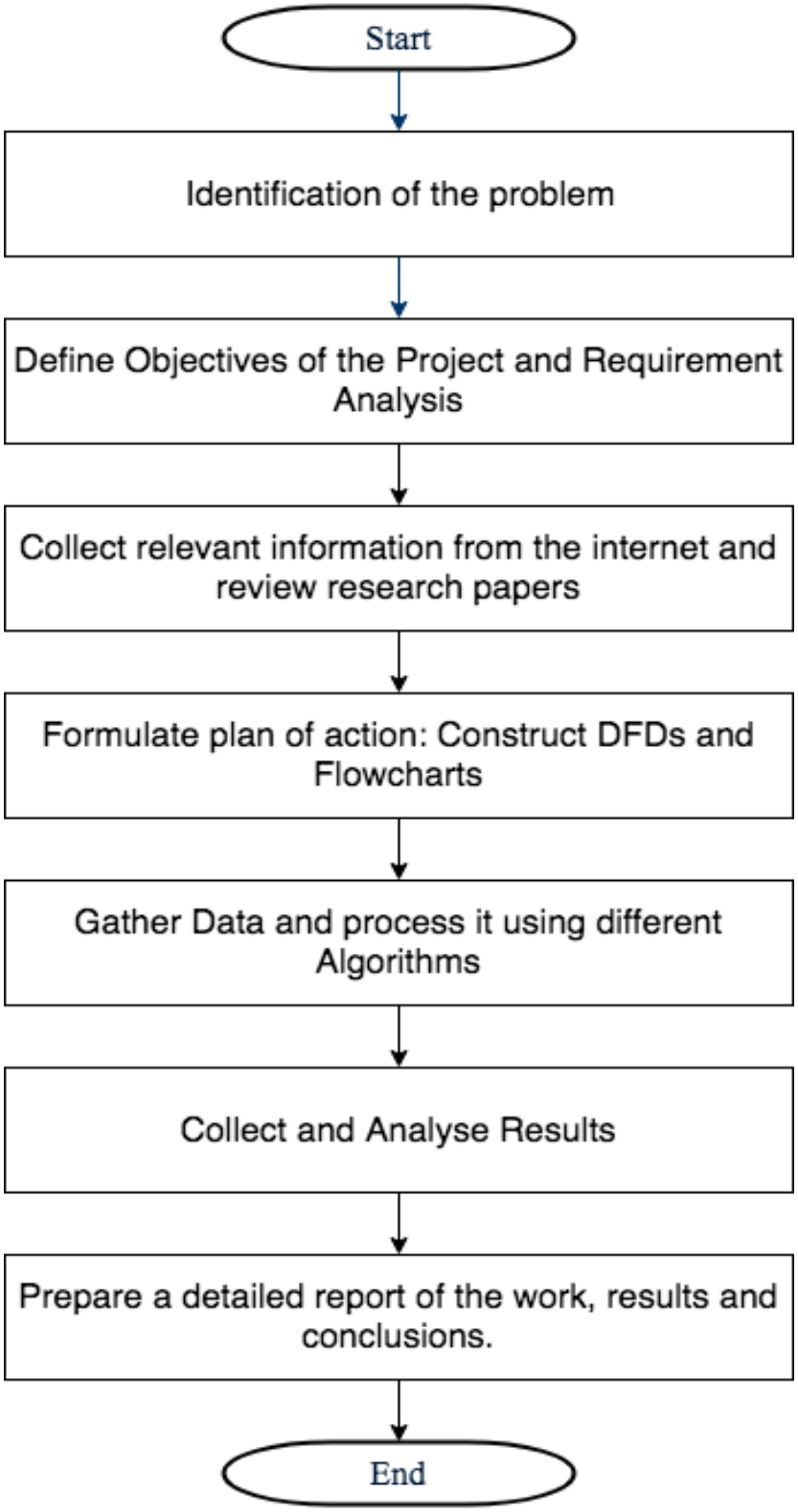


Fig 1.1 Flowchart: Project Development

The documents to be summarized will be inputted from the user and the words will be vectorised and weighted by using word embeddings and will then be used in our machine learning models to generate headlines, whereas in the case of summary generation the

document given by the user will act as the input for the algorithm words and sentences will be weighted and extractive summary will be generated using tf-idf values and weighing the words to generate vectorised form to learn the important words needed to generate the summary.

1.5. ORGANIZATION

This section gives a brief introduction of what each chapter of this report contains along with all the relevant information.

Chapter 1: Introduction

This chapter contains a brief account of what the project is about, the problem statement, objectives and the methodology adopted for the study.

Chapter 2: Literature Survey

This section reviews and examines the various research papers and freely available information that form the basis of the study and provides an insight on the relevant literature.

Chapter 3: System Development

This chapter describes the processes and steps involved in the development of an automatic text summarizer. It includes different diagrams and flowcharts to explain the development and working of the summarizer.

Chapter 4: Algorithms

This section includes various algorithms that have been used in the project. There is an explanation of what the algorithm does, the input it takes, steps included and the output that it produces.

Chapter 5: Test Plan

This segment contains information about the data sets, list of requirements for the successful functioning of the Automatic Text Summarizer and details pertaining to efficiency calculation and the metrics involved.

Chapter 6: Results and Analysis

This chapter includes a description of the results of our study so far and an analysis of the same.

Chapter 7: Conclusion and Future Work

This section contains a description of the conclusions that can be drawn from the results obtained and their analysis. It also has information about the scope of expansion of the project.

CHAPTER 2: LITERATURE SURVEY

This section reviews and examines the various research papers and freely available information that form the basis of the study and provides an insight on the relevant literature.

2.1. Early Work

Automated Text Summarization has been a popular topic of interest for researchers over the world for decades now. Extractive Text Summarization was first performed by Hans Peter Luhn [3] in 1958. Known as the father of Information retrieval, Luhn primarily aimed to summarize technical works based on the frequency and positioning of significant words. Sentences are graded and the ones with the top scores are picked. A novel approach was later provided by H.P. Edmundson [4] in 1969, where four methods namely, Cue, Title, Location and Key method Ire used to identify the centrality score of a sentence and produce a ranking in the order of importance. A neural net model used to pre-process an input data and match with a string already defined by the user was proposed by A. Das et al [5]. Featured words are extracted from the article at hand with the user defined words and in case they match, the score of the word is increased. Repetition of the process gives us an overall sentence score, which can then be used to extract a summary out of the original document. Single Document Summarization based on Extraction techniques was proposed by J Jagadeesh et al [6] in 2005. Arman Kiani and M. R. Akbarzadeh [7] proposed the use of Hybrid Fuzzy systems for text summarization in 2006.

Numerous other notable approaches have been employed in realising and improving the process of automatic text summarization, both extractive and abstractive in nature.

2.2. LexRank based Text Summarization

LexRank [1] based Text Summarization involves using graph-based methods for Natural Language Processing. It uses sentence based graphs to summarize a text document using extractive methods of summarization. Extractive summarization assess the importance of a sentence, phrase or words using various predefined methods and central sentences are taken into the summary that is produced as output.

A widely used metric for computation of the importance of a word in a sentence is IDF [1]. It is calculated as:

$$Idf = \log (N_i/N)$$

Where, N is the total number of documents and N_i is the number of documents in which the term i is present. Rare words have large IDF values while common ones like ‘a’, ‘the’, etc. have smaller IDF.

The scores of words is calculated by calculating TF-IDF values of the words (tf X idf) and if it is above a preset threshold, the word is entered into the centroid. Centroid is a pseudo document with a list of words that have their TF-IDF values above the preset threshold.

Similarity of two sentences is defined in vector space by the idf-modified cosine.

$$idf - modified - cosine(x, y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{xi \in x} (tf_{xi,x} idf_{xi})^2} \times \sqrt{\sum_{yi \in y} (tf_{yi,y} idf_{yi})^2}}$$

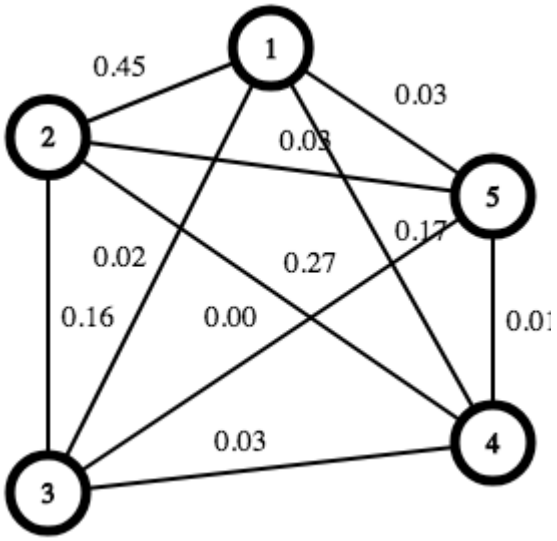
IDF-modified-cosine formula [1]

The corresponding values in a similarity matrix (N X N dimensions) is the measure of similarity between any two sentences.

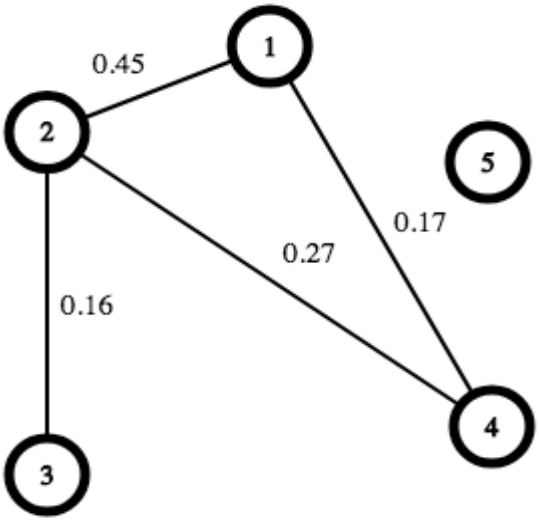
	1	2	3	4	5
1	1.0	0.45	0.02	0.17	0.03
2	0.45	1.0	0.16	0.27	0.03
3	0.02	0.16	1.0	0.03	0.00
4	0.17	0.27	0.03	1.0	0.01
5	0.03	0.03	0.00	0.01	1.0

Table 2.1 Example of Intra-sentence similarity matrix

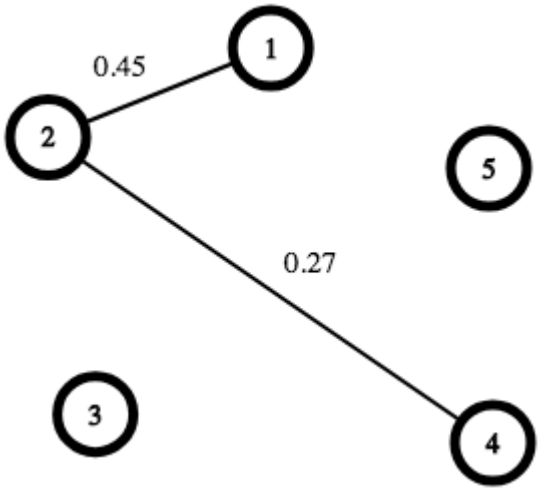
Depending on the values in the similarity matrix, weighted graph with N vertices is created (Graph 2.1) This graph is further filtered by setting different threshold values (as shown in Graph 2.2 and Graph 2.3)



Graph 2.1 Example of weighted cosine similarity graph



Graph 2.2 Example of similarity graph with threshold 0.1



Graph 2.3 Example of similarity graph with threshold 0.2

Constructing a degree centrality score matrix shows us the most central sentence in the text. Table 2.2 is the degree centrality score matrix for Graph 2.2 and 2.3.

	deg(0.1)	deg(0.2)
1	3	2
2	4	3
3	2	1
4	4	2
5	1	1

Table 2.2 Example of degree centrality score matrix

The problem with the degree centrality method is that each vote accounts for equal weightage while in reality that is seldom the case. It may happen, in this situation that, two non-important sentences support each other and move up the centrality ladder. This situation is avoided by taking a note of the centrality of the node that is supporting the node that is being examined. This is done using the following formula:

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{\deg(v)}$$

Formula for calculation of centrality of a node [1]

where p(u) is the centrality of node, adj[u] is a set of nodes that are adjacent to node u, deg(u) is the degree of node u.

Equation in Fig 2.2 is written in matrix notation and mathematical calculations on Markov chain are done with a Stochastic Matrix and after assigning equal probability of jumping to each node in graph, a modified version (called PageRank) of the equation in Fig 2.2 is obtained as shown in Fig 2.3.

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{p(v)}{\deg(v)}$$

Formula of PageRank [1]

Unlike the original PageRank[1] method, the similarity graph for sentences is undirected as cosine similarity is a symmetric relation. However, this does not change much about how the computations are made. This new metric can be called 'LexRank' and is used for centrality evaluation. The LexRank value of the sentences may be affected by the threshold that has been set.

Minor improvements can be made in the original LexRank formula can be made by looking at the similarity strength of the links. It is called continuous LexRank [1] and can be computed as:

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{idf - modified - cosine(u,v)}{\sum_{z \in adj[v]} idf - modified - cosine(z,v)} p(v)$$

Formula of Continuous LexRank [1]

Both the methods: classical LexRank [1] and continuous LexRank [1] are used in this project and the results are documented in this report.

2.3 Machine Learning Approach

Machine Learning approach for the text summarization decides the importance of words in the document by recurrent regressive training. Various pre-trained models will be used to learn the way these algorithms work and to get a comparative analysis between them to use the best learning algorithm to summarize the document.

2.3.1 Word Embeddings

I can see that the right starting point for word vector learning may be really with ratios of probabilities rather than the probabilities themselves. In GloVe's instance, the counts matrix is pre-processed by normalizing the counts and log-smoothing them. Compared to word2vec, GloVe permits for execution, meaning that it's easier to train over

more info. Both words co-occur frequently with water (as it is their shared land) and rarely --together with the unrelated word style. Almost all unsupervised procedures for studying word representations utilize the statistics of word occurrences in a corpus as the primary source of information, yet the question remains as to how I can create meaning from such figures, and how the resulting word vectors might represent that significance. On the project page it's stated that GloVe is a log-bilinear model with a Lighted least-squares objective. The design rests that ratios of probabilities have the potential for encoding some type of meaning which could be analysed as vector differences. Hence, the training objective is to learn word vectors for example their dot product equals the logarithm of the words' probability of co-occurrence. This goal and vector gaps from the term vector space associate the ratios of probabilities that are co-occurrence Since the logarithm of a ratio equals the difference of logarithms. It creates the term vectors that perform Ill on similarity tasks and on both word analogy activities and named entity recognition.

By doing reduction on a count matrix, generally speaking, vectors are learnt by count-based models. First they construct a large matrix of co-occurrence info, which includes the info on how frequently each "word" (stored in rows), can be viewed in certain "context" (the columns). The number of "contexts" needs be big, since it is basically combinatorial in size. They factorize this matrix to yield a multi-dimensional matrix of attributes and phrases, where every row creates a vector representation for each term. It is accomplished by minimizing a “reconstruction loss” which looks for multi-dimensional representations which could explain the variance from the high-dimensional data.

2.3.2 Encoder

For the Project basically 2 types of encoder were used:

Dynamic RNN

Dynamic RNN is preferred over static RNNs because of the fact that the static RNN cannot work on variable sequence lengths and variable input. The input in the project are documents and sentence length in a document cannot be fixed to a certain constant. When using static RNN the correct number of time steps in the sequence may be smaller or greater than the constant provided for the training.

Bidirectional Dynamic RNN [12]

Bidirectional Dynamic RNN can be visualised as using one RNN in general time and another one at same time but from the opposite direction and the outputs of both the RNNs are then summed up at every time step.

Using bidirectional dynamic RNNs are difficult task due to input confusion but give better results because the model now has both forward and backward information about the sequence we are learning about. The backward and forward iteration works in the same way as backward and forward propagation works in regression techniques. This in turn helps in generating the summaries as well as verifying that what we are generating is correct or not.

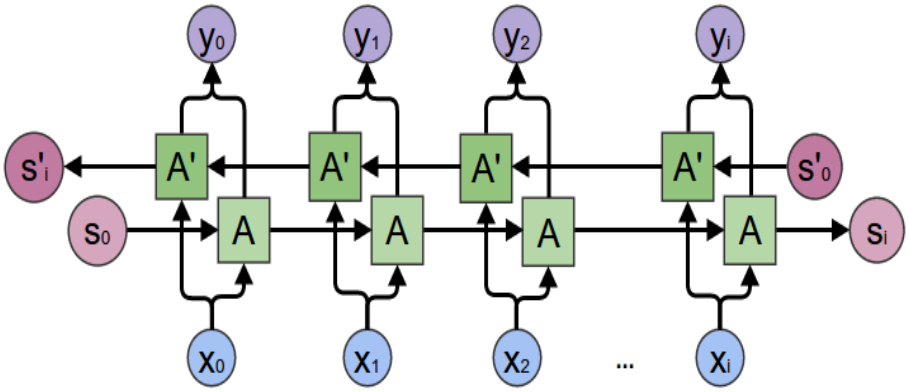


Fig 2.1 Biderectional Dynamic RNN

The major implications lie on the fact that LSTM cell provided in this RNN for every step are twice than that of the Dynamic RNN so the attention mechanism and the output are better for every document but the time taken by the biderctional dynamic RNN is more than Dynamic RNN.

2.3.3 Decoder

For the Project we used 2 decoders:

- LSTM Basic Decoder for training

  The prediction using sequence-to-sequence learning heaves more difficulties for encoding as well as decoding because of the size of input data and also their may be a considerable difference between the size of input and output sequence. Using LSTM cell in the training procedure of the decoder helps the model to retain information about the sequence inputted as text file. The LSTM cell largely improves the learning capabilities of the decoder used in the training period.

- Beam Search Decoder for inference

  Another decoder used for the project is the Beam Search Decoder and this decoder is used for inference from the trained model. Beam Search

decoder is a step ahead on the greedy search decoder and instead of greedily choosing the next step, the beam search algorithm expands the next steps in the sequence instead of choosing one step it chooses next *k* steps in the sequence. The variable *k* is user inputted. This make number of parallel beams through sequence of probabilities. The algorithm stops when on out of the *k* sequences is the output sequence, hence is used for inference.

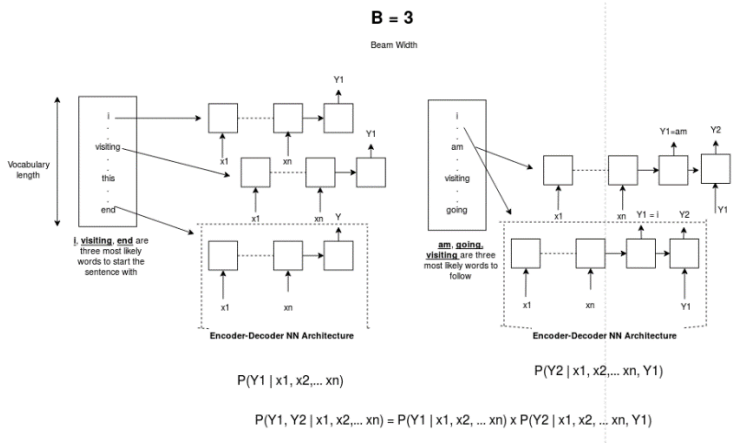


Fig 2.2 Beam Search Decoder for *k* value 3

2.3.4 Attention Mechanism

Attention Mechanism retains the knowledge gained during the training part of the mode. After each iteration attention mechanism finds the next variable.

- Bahadanau Attention [12]

  Attention is necessary for any encoder-decoder model for generalizing the test data. Using attention mechanism helps the model to learn how to efficiently minimise the cost function by looking up where to find the variables instead of just regressively training the model. This attention mechanism check how we weight hidden input vectors. Words align to their individual meaning but in reality, number of words sequenced together can give different meaning to the sentence with comparison to their individual meanings. Bahadanau Attention mechanism pairs words which occur together and also pairs words with their meanings. The two pairs are then compared with each other to justify the meaning the phrase really holds.

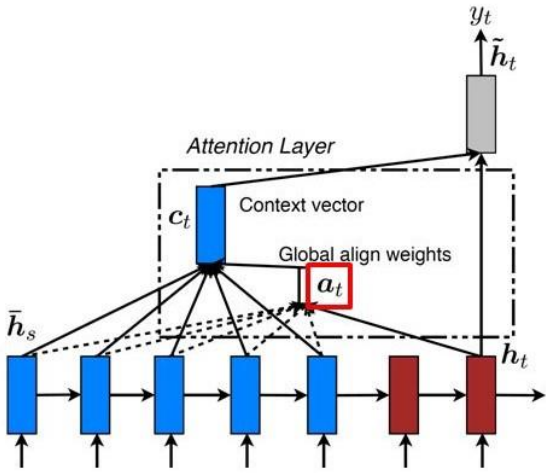


Fig 2.3 Bahadanau word attention layer. [10]

2.3.5 Machine Learning Summarization using RNN

Automatic Text Summarization using Recurrent Neural Network is one of the leading models used for giving abstractive summary of any document. Recurrent Neural Network is one of the oldest and most robust machine learning algorithms. They Ire established in the early 1980's but on account of many decades of individual research now have LSTM cell from the 1990's innovation. As a result of the memory cell RNN's are in a position to keep in memory things about the input they've received, which allows them to give accurate outputs due to the presence of RNN's. Recurrent Neural Networks generate predictive improvements in the output from consecutive statistics which other calculations cannot. This really is why they truly have been preferred algorithm for linear information such as some time collection, text, speech, monetary info, sound, movie clip, audio files since they could produce a lot deeper comprehension of an arrangement plus its own particular instance, in comparison to additional calculations. Recurrent Neural Networks (RNN) certainly really are a robust and powerful kind of neural networks and also appeal into the absolute most promising calculations out there in the present time since they're the sole ones using memory cell.

RNN works as an iterative memory cell unit, in a Recursive Neural Network the input given to node cycles through these loops. When giving an output of a cell, it takes consideration the input from the previous node and from the current input given to it by the user.

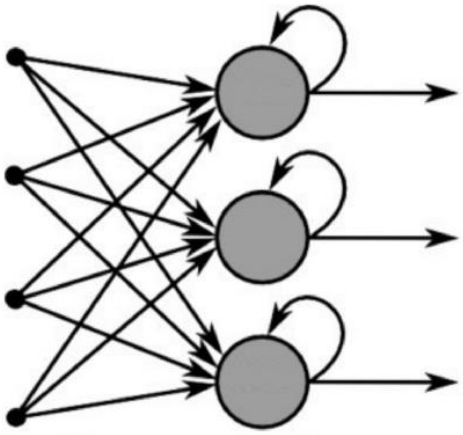


Fig 2.4 Recurrent Neural Network architecture [11]

A customary RNN includes a short-term memory. In conjunction with a LSTM that they also possess a long-term memory, however, I'll discuss this further below. Another Fantastic way to exemplify the Notion of a RNN's memory would be to describe it with an example: Imagine you get a typical feed-forward neural network and then provide it the phrase "neuron" as an input and it processes the term character by nature. A Recurrent Neural Network can recall exactly that, due to its internal memory. It generates output signal, copies that loops and output it back in the network. Hence a Recurrent Neural Network includes 2 inputs, the current and the recent past. This is vital because the arrangement of information includes crucial information about what's coming, which explains exactly why a RNN can do things other calculations cannot. A Feed-Forward Neural Network assigns, such as most of other Deep Learning boosters, a light matrix into its inputs and produces the output. Be aware that RNN's use weights to the present and to the former input.

$$h_t = f(h_{t-1}, x_t)$$

Formula for calculating the current state [2]

Where

H_t = Current State

H_{t-1} = Previous State

X_t = Input State

$$h_t = tanh(w_{hh}h_{t-1} + w_{xh}x_t)$$

Formula for applying Activation Function [2]

Where:

W_{hh} = weight at recurrent neuron

W_{xh} = weight at input neuron

$$y_t = w_{hy} h_t$$

Formula for calculating output [2]

Y_t = Output

W_{hy} = weight at output layer

Recurrent Neural Network work on the concept of forward propagation and backward propagation for error correction. In general, neural networks forward propagation is used to get the output of the input given to any particular node. To check whether the output to the given problem is correct or not I check the output of the program and if the output of the program is faulty, I do backward propagation. In backward propagation for every error at any node of the model I find the partial derivative this in turn helps to diminish the error by subtracting this value from the previously given weights. Then the given function is recursively minimized by using gradient descent tu find the local minima and absolute minima. What I do while backward propagation is use these derivatives to modify the weights of our model while training the model itself. This concept of forward and backward propagation helps in minimising the error in the model while training the model and I do not rely on iteratively checking the model and its output to get the errors I may find in the program.

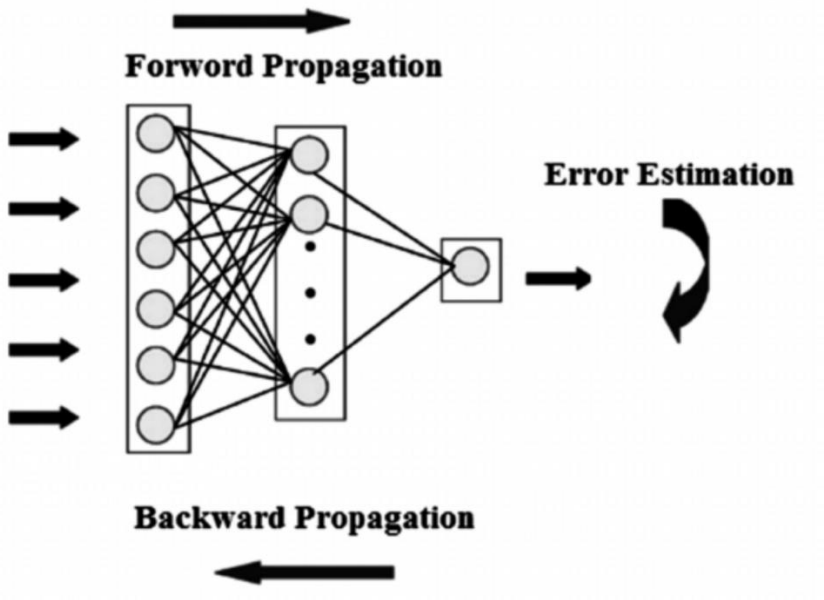


Fig 2.5 Error estimation and optimization [11]

The main reason behind using backward propagation is to minimise error in every sentence. Whereas forward propagation involves only weighing the vectors and summarizing the document, backward propagation checks the summary generated as well verifies the weight at each hidden layer node optimises it if any error occurs and reiterates the neural network.

2.3.5 Text Summarization using Seq2Seq model

Text summarization is considered as a task of creating some outline plus a headline composed of the couple paragraphs which catches the notions of the given sentences and/or the document or a short write-up. I utilize the adjective 'abstractive' to define the summary generated by the model as holding the meaning of the document the summary is not generated by splitting the sentences in the document however also a compacted paraphrasing of the principal contents of this record, most likely employing language hidden from the origin file. This task may be cast as mapping an input set of phrases at a resource file to your target chain of phrases. At the frame of sequence-to-sequence versions, an exact important model for our task would be that your attentional Recurrent Neural Network (RNN) encoder-decoder version suggested (Bahdanau et al.)[10]. Despite of the similarities, the abstractive summarization is actually an exact various problem in MT. In contrast to in MT, the aim (outline) is normally quite quick and will not rely greatly around the length of the foundation (record) from summarization. Whereas at MT, the translation is forecast to be more loss less. In addition, a barrier in summarization will be always to compress the record in a mode such as the concepts within the record are maintained. In summarization, it really is clear, although there is really just a belief of almost intervention between purpose and origin. I create the subsequent primary contributions within the job: (i) I employ the off the shelf attentional encoder-decoder RNN which has been initially produced for system interpretation into summarization, also reveal it outperforms state-of-the-art strategies on just two separate English corpora. (ii) encouraged by tangible issues from summarization which aren't satisfactorily addressed from the system interpretation established version, we suggest publication versions and reveal they supply additional advancement in operation. (iii) we suggest a new Data Set for your Job of abstractive

summarization of the record right into several paragraphs and set benchmarks.

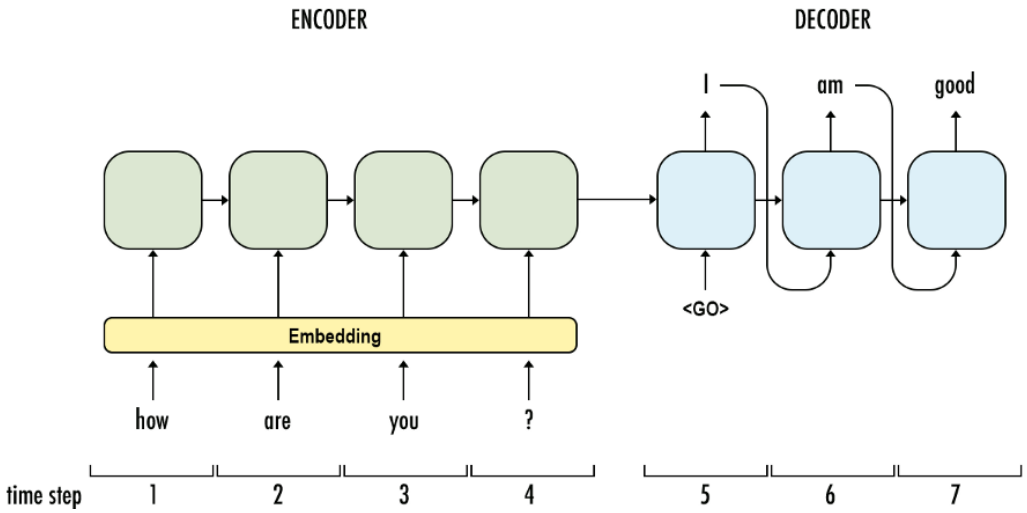


Fig 2.6 Encoding Decoding in RNN Architecture [10]

In the seq2seq model hidden layer is present in this layer I add embeddings, this embedding layer consist of vectorised vocabulary words, all the words present in the document after removing the Stopwords from the document. Each word in the embedding layer has an ID associated with it.

During the pre-processing of the data I do certain things for smooth implementation of the algorithm.

- Of every word not repeated previously in the document I add it into the vocabulary.
- Count how many numbers of time these words have occurred in the document.
- I calculate term frequency, i.e., a low frequency value to easily apply functions and calculations on the vectorised words.
- After replacing these vocabularies by their IDs, I create a copy of the document for training purposes.

Unlike the RNN, seq2seq [9] model does not have this embedding layer in between the encoder's output and decoder's input. Embeddings are a separate file where the words,

their IDs, their frequency and vectorised sentences are stored.

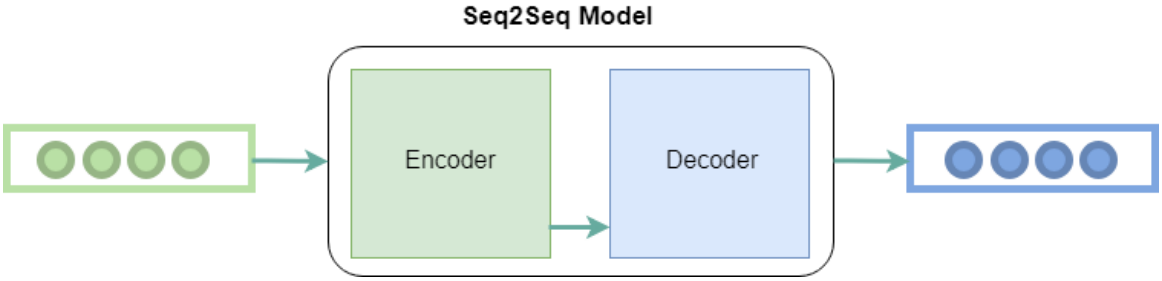


Fig 2.7 Seq2Seq Encoder Decoder Model Architecture [8]

Sequence-to-Sequence model maps input of sequence to an output of sequence, it uses two RNNs for predicting how the previous sequence of words map to the next sequence. Sequence-to-Sequence model also works perfectly fine with attention mechanism and decoding models both beam search and LSTM cell. Word embedding layer in the seq2seq model weighs the vectorised words to arrange a N*N matrix.

CONCLUSION

We learnt about different types of text summarization techniques, namely, Abstractive and Extractive Text Summarization. We learnt about how these techniques work and major algorithms used for implementing these techniques. In this project, we will apply these algorithms and by comparative analysis select best one for implementing out of the algorithms used in the project. We will implement best abstractive and extractive summarization technique for headline generation and text summarization respectively.

CHAPTER 3: SYSTEM DEVELOPMENT

This section describes the processes and steps involved in the development of an automatic text summarizer. The current work is focussed on extractive summarization of a text document using LexRank computations [1].

3.1. Software Specifications

Each the calculations are composed in python-3.6, be it in Jupyter laptops (*.ipnyb documents) in addition to the .py files. This is a conscious choice due to python interpreter-- that allows for debugging that is effortless and code implementation. Not just that, the syntax helps to ensure that the details unique to a programming language does not bogged down the reader.

Python-3.6 additionally ensures quick monitoring and instantaneous execution within Jupyter laptops with no necessity for compiling the app time and with each slight change.

In addition, the learning frame TensorFlow, as its title implies has been composed for python. Rather than being a different library TensorFlow is created using code for integration. Some Benefits of TensorFlow along with other learning frameworks have been recorded under:

- TensorFlow creates the most from CPUs and the computation may also be hastened by employing graphical processing units (GPUs).
- TensorFlow is optimized advertising incurs computation is the fact that it supports dynamic networks. This can be essential when I want the behaviour to change in run-time of our network.
- It's not hard to learn and use for learning algorithms and system learning.
- Open source.

Other outside libraries include--

Numpy: This is a high-performance processing bundle.

Matplotlib: This bundle is a library that arouses plots, high quality graphs, characters in a number of formats such as Jupyter laptops.

Pickle: This bundle indicates the progress of a loop for a progress bar, which may crucial when training word sensors to appraise the progress and classifiers.

These packages have to be installed from the machine where the calculations are analysed or to be implemented. With no bundles, the applications may not cause the behaviour that is desirable. They are vital for making sure that the modules operate explained here and as anticipated.

These packs are installed and also the code could be implemented.

3.2 Hardware Specifications

The algorithms which Ire implemented are independent, they could operate on all platforms. That having been said, that the power on those machines might be adequate to meet GPU requirements and the RAM. It is suggested that the Jupyter Notebooks to be used or the algorithms provided lest your system may crash because of inadequate memory requirements, should be implemented on Google Collaboratory.

3.3 Data Pre-processing

LexRank [1] (fig 3.2) is distinguished by PageRank technique. This strategy works right off the bat by producing a diagram, linking all the different sentences in the article, book or review. Each sentence speaks to another sentence or point in the corpus, and the edges signifies the semantic similarity between different sets pieces of lines in the corpus. In this exploration, the similarity between sentences is measured by considering each sentence as a matrix of words. It also implies similitude measure between a matrix of words is registered by recurrence of a single word event in two different matrices. The essential estimation utilizes TF-IDF plan, in which TF adds to the similitude quality of the value as the quantity of individual element in the matrix events gets higher. Then again, the backwards record recurrence respects low recurrence words conversely adds to increase in incentive to the estimation. This TF-IDF detailing is later utilized as an estimation for likeness between the word matrices by using it in the idf adjusted cosine formula (fig 3.1).

$$idf - modified - cosine(x, y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{xi \in x} (tf_{xi,x} idf_{xi})^2} \times \sqrt{\sum_{yi \in y} (tf_{yi,y} idf_{yi})^2}}$$

IDF-modified-cosine formula [1]

This computation is estimating the 'separation' between two sentences x and y, the more comparable two sentences, the less redundant their connection becomes, i.e., they hold a certain degree of similarity between them and can be used in the summarization process.

This similitude value is used to generate a closeness lattice, which is utilized as a comparability chart between different matrices. The LexRank calculation measure the significance of word matrix in the diagram by thinking about its relative significance to its adjacent matrices, where a positive commitment will raise the significance of a matrix's neighbor, while a negated value commitment will bring down the significance estimation of a matrix's neighbor.

To distinguish the most imperative sentences from the subsequent closeness grid, I apply a thresholding component. An edge esteem is utilized to sift through the connections between matrices whose value fall beneath the set threshold value. The outcome is a subset of aforementioned likeness chart, from where I can pick one aggregation that has the maximum degree. This collection is viewed as central or speaks to a rundown sentence of the text.

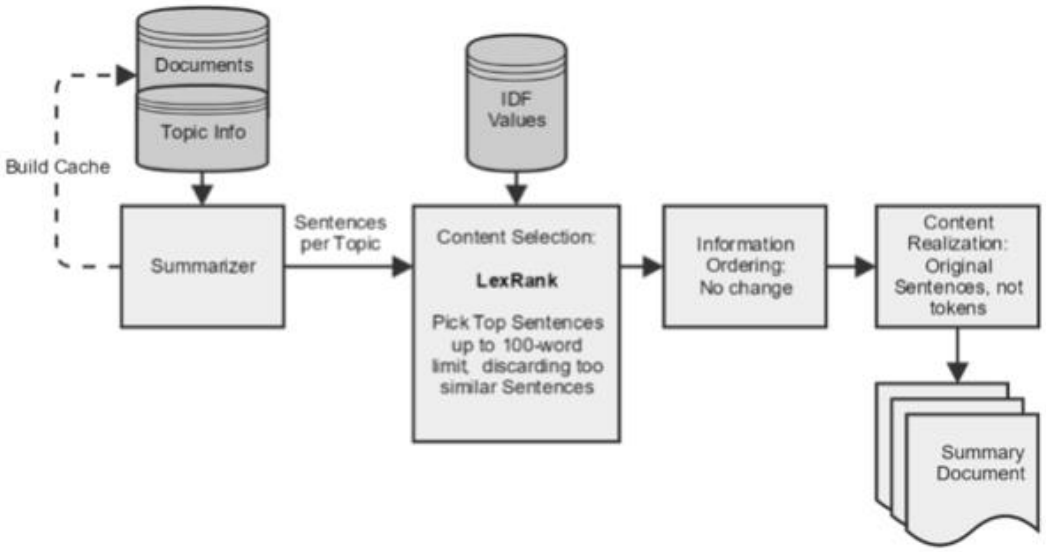


Fig 3.1 Flowchart: LexRank based Summarization [1]

LexRank picks up sentences having top most IDF values to choose the similar word matrices to form the most accurate summary of any corpus. Document is fetched from the database, which is then tokenized into words and subsequently sentences which are ranked on the basis of similarity between them and are then chained together to form the summary.

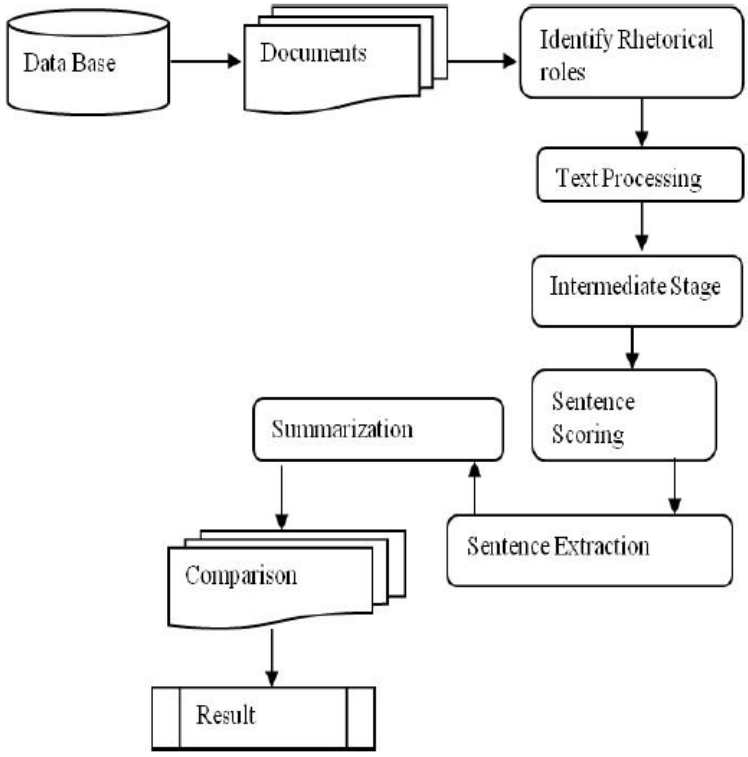


Fig 3.2 System Architecture for LexRank [1]

Documents are fetched from the database **text** pre-processing takes place. Pre-processing includes identifying rhetorical roles and initiating stop words and punctuations. The words are tokenized and given as input to the algorithm for the process of sentence scoring. A threshold value is stated for the process of sentence extraction for the summarization process to initiate. The fetched summary is then compared with its parent article to generate index score.

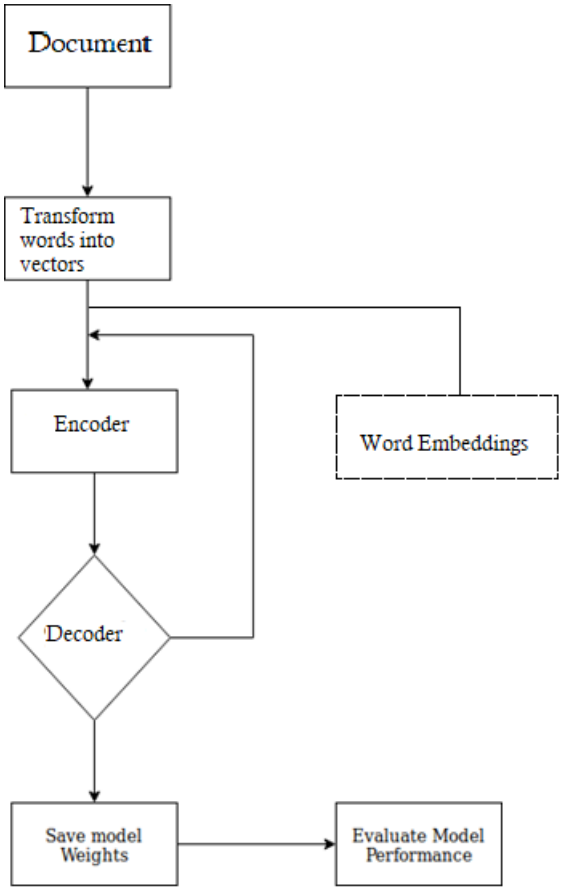


Fig 3.3 Working of Algortihm

Document are fetched from the database and/or input is taken from the user. The words are from the document are sliced and transformed into vectors, These vectors are then given as a input in the encoder layer of our encoder decoder model, where iteratively output of the encoder model is used as input of decoder model then weights on the vectorised words are put by word embeddings this process iteratively runs for each sentence formation.

3.4 Code and Sources

The work on the project has been done on python language. Various modules like *numpy*, *tensorflow*, *nltk*, *pickle*, *GloVe* and *word2vec* were used, *argparse* was used for parsing the documents. For comparision of the project different models were used. Out of the models some of the models were pre-trained others were implemented using *spyder* ide was used and for better representation of the project *jupyter notebooks* were used.

3.4.1 LexRank

LexRank algorithm uses PageRank method, summarizes documents by using a graph based method using all the sentences present in the corpus. The code for LexRank was implemented using LexRank module, specifically importing the STOPWORDS sub-module.

3.4.2 NLP Score Based Ranking

NLP score based ranking for coded in python using *BeautifulSoup, nltk, urllib* and *heapq* modules. From *nltk* module *stopwords* were imported, *BeautifulSoup* was used to convert html file to .txt file. The algorithm works by scoring sentences by the number of occurrences of words which are previously scored and then discreted by counting their number of occurence in the corpus. This model works by taking input from the users as a webpage. Parses the webpage using *BeautifulSoup* parser which converts the html document into text file and using score based ranking summarises the document. External codes were not used while implementing this model.

3.4.3 Seq2Seq Model

Sequence-to-Sequence was implemented seq2seq module with seq2seq encoder decoder model to generate headlines. The model was not coded but was trained and tested. Weights were optimised. Two models were used for word embeddings, vector wighted model and GloVe pre-trained module. The cod was used form github repository [13] .

3.4.4 Recursive RNN

Recursive RNN model was implemented using *keras, numpy* and *os* module. This model uses keras backend and RNN encoder-decoder module to train and test data. The module for trained RNN model was imported from github repository [14]. The code contains the model. It was trained and tested for different epochs and weights were optimised.

CONCLUSION

We learnt what hardware and software will be needed for smooth functioning of our system also how the system will take input and it will function once input is provided. We have taken into consideration the how precisely the data has to be preprocessed and

learnt what preprocessing actuallly does to the data and why is it necessary? We defined our system architecture and working of the algorithms.

CHAPTER 4: ALGORITHMS

4.1. Algorithms

This section includes various algorithms that have been used in the project. There is an explanation of what the algorithm does, the input it takes, steps includes and the output it produces.

4.1.1. Extractive Approach

This includes picking out sentences or phrases that are more central to the document at hand and give a synopsis that is more concise and relevant. Extractive Text Summarization is relatively easier but suffers from more restrictions than its counterpart, Abstractive Summarization. Extractive Text Summarization is realised by using LexRank[1]. It is a graph-based method used to compute the importance of text, sentences or phrases for NLP. This technique is used extensively for Extractive Text Summarization of an article, utilising the centrality of a sentence in the article to get the most relevant and significant pieces in the documents at hand.

- INPUT [1]: The input for a LexRank algorithm is a text document.
- STEPS [1]:
    - Slice the Text Document into the sentences.
    - Tokenize the words in the document into vector space.
    - Calculate the Inverse Document Frequency of the Words.
    - Divide the document into clusters.
    - Find the most central sentence in the cluster to give a general information related to main theme.
    - Centrality is defined by looking into the centrality of the words, i.e., find centroid of the cluster in the vector space.
    - The centroid of a cluster is a pseudo-document which consists of words that have tf X idf scores above a predefined threshold, where tf is the frequency of a word in the cluster, and idf values are typically computed over a much larger and similar genre data set.

```
Input: An array S of n sentences, cosine threshold t
Output: An array C of centroid scores
Hash WordHash;
Array C;
/* compute tf x idf scores for each word */
 for i -->1 to n do
        for each word w of S[i] do
                WordHash {w}{"tfidf"} = WordHash{w}{"tfidf"} + idf{w};
        end
end
 /* construct the centroid of the cluster */
/* by taking the words that are above the threshold */
for each word w of WordHash do
        if WordHash{w}{"tfidf"}>t then
                WordHash{w}{"centroid"} = WordHash{w}{"tfidf"};
        end
        else
                WordHash{w}{"centroid"} = 0;
        end
end
/* compute the score for each sentence */
for I ← 1 to n do
        C[i] = 0;
        foreach word w of S[i] do
                C[i] = C[i] + WordHash{w}{"centroid"};
        end
end
return C;
```

Algorithm to compute centroid scores [1]

- We hypothesize that the sentences that are more central (important) to the topic are present near this centroid in the vector space.
- To define similarity, I represent each sentence as an N-dimensional vector, where N is number of all possible words in the target

document. With each occurrence the, the value of the corresponding word is changed:

vector value = (the number of occurrence) * (idf value)

Then idf-modified-cosine is calculated to generate the similarity between corresponding vectors.

$$idf - modified - cosine(x,y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{xi \in x} (tf_{yi,y} idf_{xi})^2} \times \sqrt{\sum_{yi \in y} (tf_{yi,y} idf_{yi})^2}}$$

IDF-modified-cosine formula [1]

- For only getting significant sentences in the summary I set a very stringent threshold because most of the sentences in the centroid of the document will be related to the topic and the summary of the document.
- Adjacency Matrix is constructed by these idf-modified-cosine.

Input: A stochastic, irreducible and aperiodic matrix M

Input: matrix size N, error tolerance t

Output: eigen vector P

Algorithm for Power Method for computing the stationary distribution of Markov Chain.

$p_0 = \frac{1}{N} 1;$

T = 0;

repeat

T=t+1

$P_t = M^T p_{t-1};$

$\varrho = ||p_t - p_{t-1}||;$

until $\varrho < t;$

return $p_t;$

- I calculate the LexRank of by summing the elements of the Cosine Matrix/Degree.

Input: an array S of n sentences, cosine threshold t

Output: An array L of LexRank scores

Algorithm for Computing LexRank scores [1]

```
Array CosineMatrix[n][n];
Array Degree[n];
Array L[n];
for I – 1 to n do
        for j – 1 to n do
                CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
                if CosineMatrix[i][j] > t then
                        CosineMatrix[i][j] = 1;
                        Degree[i]++;
                end
                else
                        CosineMatrix[i][j] = 0;
                end
        end
end
for I←1 to n do
        for j ←1 to n do
                CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
        end
end
L = PowerMethod(cosineMatrix,n,e);
return L;
```

- OUTPUT [1]: The output of the algorithm is a summarized version of the original text document.

4.1.2 Abstractive Summary

This task includes generating a summary of a document or can generate headlines for the document by capturing the features of that passage. The summary generated from the 'Abstractive' summarization algorithm [9] does not contain the already present sentences in the source, but either spins the vocabulary or rephrase in a compressed yet effective way keeping the main contents of the source using words not used in the document.

- INPUT: Input is a text document.
- STEPS:
    - Input sequence is mapped to vectorised vocabulary in the embedding layer.
    - Embedding layer state size is similar to the vector matrix of the vocabulary.
    - The words are ranked according to the embedding layer using pre-trained model such as GloVe or Word2Vec.
    - Feature rich encoder is used to get a better preliminary output from the encoder layer of the model.
    - This output is used as the input for the decoder layer. Encoder uses POS tagging and word sense disambiguation to get better meaning for the words.
    - The encoder decoder model iteratively collects the salient ideas of that particular document.
    - LSTM cell is used as memory unit for temporary memory allocation during iteration of one single sentence in the encoder decoder model.
    - Continuous values such as TF and IDF are converted into discrete value for better use in the vectorised words embedding layer.
    - For every word we find its embedding in the feature matrix and from all of its other tags and add up all these words for a longer matrix.

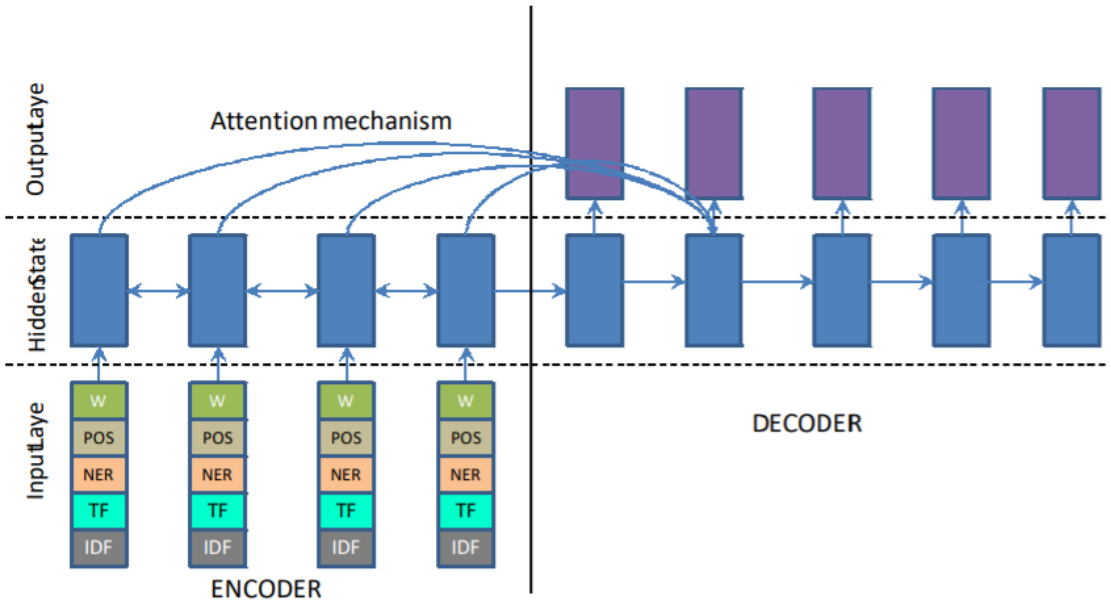


Fig 4.1 Encoder 4-Decoder Model with Attention Mechanism [10]

CONCLUSION

In this chapter we learnt about different algorithms we will use in the given project for Extractive and Abstractive Summarization respectively, how these algorithms will work and generated a sample summary for extractive summarization techniques. The algorithm used for comparison have been pre-trained and were just implemented. One algorithm each for Abstractive and Extractive approach each were implemented and executed, others were used for comparative analysis only.

CHAPTER 5: TEST PLAN

5.1. Dataset

- BBC news articles dataset

  The data set includes 2225 documents (text files) derived from BBC database of news articles corresponding to various topical areas.

- Amazon Fine Food reviews

  A data set of 568,454 food reviews, with Amazon users left upto 2012.

5.2. Results Analysis

- The resultant text must be a shorter version of the original text document.
- Sentences or phrases with higher degree of centrality must be picked to create a summary.
- The system must withstand any number of lines of data provided to it.
- It must work on alphanumeric data of all types.
- The model must be time efficient along with not losing its efficiency in the meantime.
- It must run smoothly on all platforms (Operating Systems) like Windows, UNIX based systems etc.

5.3. Performance Metrics

Two kinds of systems of measurement that are used to compute the efficiency of summarization in the project.

5.3.1. ROUGE:

ROUGE [12] is a set of metrics that is used for evaluating machine-based (Automatic) summarization of text as Ill as natural language translation.

The main two aspects that impact the evaluation are:

- Fluency of the summary, which is more prominent in the abstractive summarization.
- Adequacy of the summary

ROUGE only try to assess the adequacy of the summary not the fluency of the summary being generated by the machine, it carries out this task by counting the number of n-grams generated in the summary which match the n-grams in your

reference summary (or summaries, because ROUGE support many numbers of corpus for reference.) And for these multiple corpora references, the ROUGE scores of all references are averaged.

It cannot determine the fluency, whether the summarized content is coherent or whether the sentences flow in sensible manner because it is only based on content overlap. But some high-order n-gram to some extent judge fluency of the generated summary.

Recall: How much of the reference summary is the machine generated summary recovering. It is, in case of individual words computed as:

$$\frac{number\ of\ overlapping\ word}{total\ words\ in\ reference\ summary}$$

Precision: How much of the machine generated summary was relevant to the summary to be generated. It is, in case of individual words computed as:

$$\frac{number\ of\ overlapping\ word}{total\ words\ in\ system\ summary}$$

5.3.2. BLEU:

BLEU [11] is one of the algorithms used for evaluating text summary which has been machine generated and stands for Bilingual Evaluation Understudy. Quality is considered by cross checking the validation of the summary with that of a human. “Machine generated summary should be as close as possible to a human summary of same document to be better” – BLEU algorithm is built around this idea. BLEU can be regarded as the very first metric to be have a high correlation with quality of human summarization and judgement as such, being free to use BLEU still remains one of the most popular metric for Text Summarization.

Calculation:

- Translated segments are taken as a single unit.
- These segments are generally set of summarized sentences.
- These translated segments are compared to a set of good quality reference translations.

- To check the overall quality of the summary, these individual scores are then averaged over the whole corpus.
- Grammatical errors as well as intelligibility are not taken into account.

CONCLUSION

In this chapter we defined our test plan, our requirements and our metrics on which the summarized documents will be checked for accuracy, precision and recall. We learnt how these efficiency metrics work and calculate theses scores for summarized documents.

CHAPTER 6: RESULTS AND PERFORMANCE ANALYSIS

6.1. Results and Discussion

The output of summarization consists of two summaries, one with the classical LexRank [1] computations and other with continuous LexRank [1] computations. The following are two sets of sample outputs for two distinct articles.

Fig 6.1 has a piece of text which when processed through the summariser results in two summaries, one through classical LexRank [1] and another through continuous LexRank [1], as shown in fig 6.2.

```
['Labour plans maternity pay rise\n', '\n', 'Maternity pay for new moth-
ers is to rise by £1,400 as part of new proposals announced by the Trade
and Industry Secretary Patricia Hewitt.\n', '\n', 'It would mean paid
leave would be increased to nine months by 2007, Ms Hewitt told GMTV\'s
Sunday programme. Other plans include letting maternity pay be given to
fathers and extending rights to parents of older children. The Tories
dismissed the maternity pay plan as "desperate", while the Liberal Demo-
crats said it was misdirected.\n', '\n', 'Ms Hewitt said: "We have al-
ready doubled the length of maternity pay, it was 13 weeks when we were
elected, we have already taken it up to 26 weeks. "We are going to ex-
tend the pay to nine months by 2007 and the aim is to get it right up to
the full 12 months by the end of the next Parliament." She said new
mothers were already entitled to 12 months leave, but that many women
could not take it as only six of those months were paid. "We have made a
firm commitment. We will definitely extend the maternity pay, from the
six months where it now is to nine months, that\'s the extra £1,400."
She said ministers would consult on other proposals that could see fa-
thers being allowed to take some of their partner\'s maternity pay or
leave period, or extending the rights of flexible working to carers or
parents of older children. The Shadow Secretary of State for the Family,
Theresa May, said: "These plans were announced by Gordon Brown in his
pre-budget review in December and Tony Blair is now recycling it in his
desperate bid to win back women voters."\n', '\n', 'She said the Con-
servatives would announce their proposals closer to the General Elec-
tion. Liberal Democrat spokeswoman for women Sandra Gidley said: "While
mothers would welcome any extra maternity pay the Liberal Democrats feel
this money is being misdirected." She said her party would boost mater-
nity pay in the first six months to allow more women to stay at home in
that time.\n', '\n', 'Ms Hewitt also stressed the plans would be paid
for by taxpayers, not employers. But David Frost, director general of
the British Chambers of Commerce, warned that many small firms could be
"crippled" by the move. "While the majority of any salary costs may be
covered by the government\'s statutory pay, recruitment costs, advertis-
ing costs, retraining costs and the strain on the company will not be,"
he said. Further details of the government\'s plans will be outlined on
Monday. New mothers are currently entitled to 90% of average earnings
for the first six weeks after giving birth, followed by £102.80 a week
until the baby is six months old.\n']
```

Fig 6.1 Sample Article 1 [7]

```
CLASSICAL LexRank SUMMARY: The BBC under-
stands that as chancellor, Mr Osborne, along
with the Treasury will retain responsibility
for overseeing banks and financial regula-
tion.Before entering Parliament, he was a
special adviser in the agriculture depart-
ment when the Tories were in government and
later served as political secretary to Wil-
liam Hague.

CONTINUOUS LexRank SUMMARY: The BBC under-
stands that as chancellor, Mr Osborne, along
with the Treasury will retain responsibility
for overseeing banks and financial regula-
tion.

End of document
```

Fig 6.2 Sample Summary 1 [2]

Fig 6.3 shows ROUGE score for the article 1 and summary 1 shown in the fig 6.1 and fig 6.2 respectively.


```
[{"rouge-1": {"f": 0.49411764217577864,
              "p": 0.5833333333333334,
              "r": 0.42857142857142855},
  "rouge-2": {"f": 0.23423422957552154,
              "p": 0.3170731707317073,
              "r": 0.18571428571428572},
  "rouge-l": {"f": 0.42751590030718895,
              "p": 0.5277777777777778,
              "r": 0.3877551020408163}}]
```


Fig 6.3 ROUGE Scores (Summary 1.1) [5]


```
[{'rouge-1': {'f': 0.7142857092857143,
   'p': 0.7142857142857143,
   'r': 0.7142857142857143},
  'rouge-2': {'f': 0.4999999950000001, 'p': 0.5, 'r': 0.5},
  'rouge-l': {'f': 0.7142857142852144,
   'p': 0.7142857142857143,
   'r': 0.7142857142857143}}]
```


Fig 6.4 ROUGE Scores (Summary 1.2) [5]

Fig 6.4 has a piece of text which when processed through the summariser results in two summaries, one through classical LexRank [1] and another through continuous LexRank [1], as shown in fig 6.5.

LexRank is distinguished by PageRank technique.This strategy works right off the bat by producing a diagram made out of all sentences in the corpus.Each sentence speaks to one hub and the edges are similitude connection between sentences in the corpus.In this exploration they measure likeness between sentences by considering each sentence as pack of-words demonstrate.This implies the similitude measure between sentences is registered by recurrence of word event in a sentence.
The essential estimation is utilizing TF-IDF plan where tem recurrnce (TF) adds to the similitude quality as the quantity of word events is higher. Then again the backwards record recurrence respects low recurrence words conversely adds to higher incentive to the estimation. This TF-IDF detailing is then utilized as an estimation for likeness between sentences by utilizing it in this idf-adjusted cosine recipe.
this recipe is fundamentally estimating the 'separate' between two sentences x and y. the more comparable two sentences the more 'closer' they are to one another. This similitude measure is then used to manufacture a closeness lattice which can be utilized as a comparability chart between sentences. The LexRank calculation measure the significance of sentences in the diagram by thinking about its relative significance to its neighboring sentences where a positive commitment will raise the significance of a sentence's neighbor while a negative commitment will bring down the significance estimation of a sentence's neighbor.
This thought is essentially the equivalent with PageRank except if it is utilized in including the significance of sentence a given arrangement of sentences. To remove the most imperative sentences from the subsequent closeness grid we apply a thresholding component. An edge esteem is utilized to sift through the connections between sentences whose weights are fall beneath the edge. The outcome is a subset of the likeness chart from where we can pick one hub that has the most elevated number of degree. This hub is viewed as remarkable or speaks to a rundown sentence of the corpus.

Fig 6.5 Sample Article 2 [7]

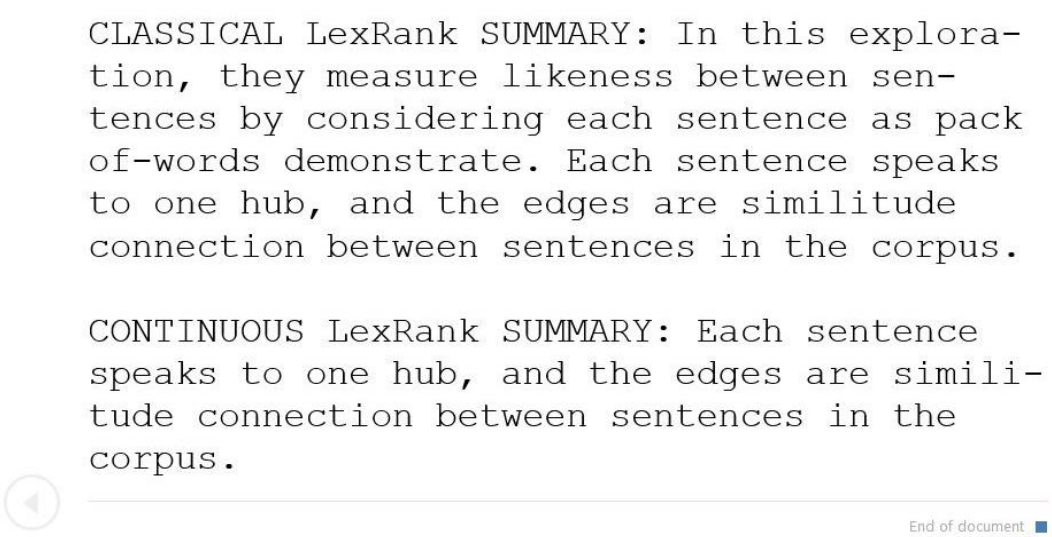


Fig 6.6 Sample Summary 2 [7]

Fig 6.6 shows ROUGE score for the article 2 and summary 2 shown in the fig 6.4 and fig 6.5 respectively.

```
Preparing documents...
Running ROUGE...
---------------------------------------------
1 ROUGE-1 Average_R: 0.28242 (95%-conf.int. 0.25721 - 0.30877)
1 ROUGE-1 Average_P: 0.30157 (95%-conf.int. 0.27114 - 0.33506)
1 ROUGE-1 Average_F: 0.28196 (95%-conf.int. 0.25704 - 0.30722)
---------------------------------------------
1 ROUGE-2 Average_R: 0.10395 (95%-conf.int. 0.08298 - 0.12600)
1 ROUGE-2 Average_P: 0.11458 (95%-conf.int. 0.08873 - 0.14023)
1 ROUGE-2 Average_F: 0.10489 (95%-conf.int. 0.08303 - 0.12741)
---------------------------------------------
1 ROUGE-L Average_R: 0.25231 (95%-conf.int. 0.22709 - 0.27771)
1 ROUGE-L Average_P: 0.26830 (95%-conf.int. 0.23834 - 0.29818)
1 ROUGE-L Average_F: 0.25142 (95%-conf.int. 0.22741 - 0.27533)

Elapsed time: 0.458 seconds
```

Fig 6.7 ROUGE Scores (Summary 2) [5]

```
Generated: 10 Amazing Halloween Costumes
Origianl: 8 Awesome Halloween Eye Makeup Ideas
Article: These makeup artists take Halloween to a new level .

Generated: The Best Of The 90s
Origianl: The ' 90s
Article: This picture sums up the ' 90s pretty nicely ... Via WTFoodge^

Generated: Man found dead in shooting
Origianl: Suspect dead after Adams Co. SWAT standoff
Article: THORTON^ , Colo. - A man wanted for a shooting at an Adams County business complex was found dead from an
apparently self-inflicted gunshot wound Wednesday afternoon .

Generated: This Is What the World 's Most Expensive Space System Looks Like
Origianl: Light Might Not Be the Fastest Thing in the Universe
Article: Italian physicists say they 've recorded sub-atomic^ particles that travel faster than the speed of light .
If true , it would disprove^ Einstein 's theory of special relativity^ . Meanwhile , in America , we 've managed to
perfect the technology to measure the speed of pizza .

Generated: The Most Annoying Tweets From The Internet
Origianl: TWItoShirt^
Article: Not into Facebook Statuses^ but wish someone would put your most brilliant thoughts ( or cleverest quips )
onto a t-shirt ? Now you can wear your tweets proudly with some help from TWItoShirt^ .
```

Fig 6.8 Sample Summary for extractive seq2seq approach [9]

"general motors corp. said wednesday its us sales fell ##.# percent in december and four percent in #### with the biggest losses coming from passenger car sales ."

> Model output: gm us sales down # percent in december

> Actual title: gm december sales fall # percent

"japanese share prices rose #.## percent thursday to <unk> highest closing high for more than five years as fresh gains on wall street fanned upbeat investor sentiment , dealers said ."

> Model output: tokyo shares close # percent higher

> Actual title: tokyo shares close up # percent

"hong kong share prices opened #.## percent higher thursday on follow-through interest in properties after wednesday 's sharp gains on abating interest rate worries , dealers said ."

> Model output: hong kong shares open higher

> Actual title: hong kong shares open higher as rate worries ease

"the dollar regained some lost ground in asian trade thursday in what was seen as a largely technical rebound after weakness prompted by expectations of a shift in us interest rate policy , dealers said ."

> Model output: dollar stable in asian trade

> Actual title: dollar regains ground in asian trade

"the final results of iraq 's december general elections are due within the next four days , a member of the iraqi electoral commission said on thursday ."

> Model output: iraqi election results due in next four days

> Actual title: iraqi election final results out within four days

Fig 6.9 Sample Summary using Recursive RNN approach [9]

In the late 19th century, scientists first argued that human emissions of greenhouse gases could change the climate.

The phrase began to come into common use, and in 1976 Mikhail Budyko's statement that "a global warming up has started" was widely reported.

Public attention increased over the summer, and global warming became the dominant popular term, commonly used both by the press and in public discourse.

They challenged the scientific evidence, argued that global warming would have benefits, and asserted that proposed solutions would do more harm than good.

In 1986 and November 1987, NASA climate scientist James Hansen gave testimony to Congress on global warming.

Fig 6.10 Sample Summary for Global Warming Wikipedia page using NLP Score Based Ranking [4]

Method	Document			Comment		
	RG-1	RG-2	RG-W	RG-1	RG-2	RG-W
Sentence Lead	0.495	0.420	0.214	-	-	-
LexRank	0.506	0.432	0.219	0.348	0.198	0.127
RNN	0.497	0.440	0.218	0.374	0.212	0.140
Seq2Seq	0.422	0.357	0.172	0.111	0.062	0.041
Seq2Seq-Glove	0.562	0.519	0.234	0.455	0.291	0.165
Score-based ranking	0.577	0.524	0.246	0.462	0.309	0.176
CNN	0.576	0.523	0.246	0.467	0.319	0.178

Tab 6.1 ROUGE Score Comparison

Above mentioned screenshots are the summaries of the documents using different algorithms learned and implemented during the project period. We learned that the extractive summaries are easier to implement and quite nicely upholds the information given the text document provided. Extractive approach gives better results for larger documents because data provided for scoring the words and sentences is quite big hence the scoring is more legitimate. For smaller documents, extractive approach doesn’t heed good results. The values calculated are not dependable because due to small size of corpus non-essential words may get a good rating also. For smaller documents extractive summary mostly gives better results. Extractive summary is more difficult to implement and need

regressive training to improve its memory cell and attention mechanism. Training data and test data was split into 80:20. Keras and TensorFlow were used for training the models. Abstractive summaries work better for smaller documents, for larger documents the training time is more and the memory cells are not able to efficiently retain data. Although for smaller document the abstractive approach works better than the extractive approach. Also, for headline generation extractive approach works very poorly compared to abstractive approach.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1. Conclusion

LexRank and continuous LexRank can be calculated using Eigenvector and Markov chain computations based on TF-IDF and IDF-modified-cosine to determine the importance of a sentence in the text document. More important sentences from the original document are retained in the summary. Efficiency of LexRank has been calculated using ROUGE metrics and it can be concluded that LexRank can be used effectively for Extractive Text Summarization of a document. Different Machine Learning Models have been used to compute the summary of any given document using GloVe and Word2Vec pre-trained word embeddings. Efficiency of these algorithm has been calculated using ROGUE metrics and it can be concluded that RNN with LSTM cell can be used effectively for the Abstractive Text Summarization.

We learnt about different types of text summarization techniques, namely, Abstractive and Extractive Text Summarization. We learnt about how these techniques work and major algorithms used for implementing these techniques. In this project, we will apply these algorithms and by comparative analysis select best one for implementing out of the algorithms used in the project. We will implement best abstractive and extractive summarization technique for headline generation and text summarization respectively.

7.2. Scope

- The scope of our project is confined to summarising a single text document and can be expanded to working on multiple documents that are related to one another.
- The project can also be extended to include features like the ability to read from images and other formats.
- The functionalities of the project can be applied to various domains where time is of the essence and documents need to be skimmed through quickly and efficiently (for example, legal or police affairs.)

REFERENCES

[1] G. Erkan, D. Radev, “LexRank: Graph-based Lexical Centrality as Salience in Text Summarization”, Journal of Artificial Intelligence Research 22, 2004.

[2] Selvani Deepthi Kavila, Dr. Radhika Y, “Extractive Text Summarization Using Modified Iighing and Sentence Symmetric Feature Methods”, I.J. Modern Education and Computer Science, 2015.

[3] H.P.Luhn, “The Automatic Creation of Literature Abstracts”. IBM Journal of Research and Development, 1958.

[4] H.P. Edmundson, “New Methods in Automatic Extracting”, Journal of the Association for Computing Machinery, April 1969.

.[5] A.Das, M.Marko, A.Probst, M.A.Portal, C.Gersheson ―Neural Net Model For Featured Word Extraction‖, 2002.

[6] Jagadeesh J, Prasad Pingali, Vasudeva Varma, “Sentence Extraction based single Document Summarization”, Workshop on Document Summarization, 19th and 20th March, 2005, IIIT Allahabad.

[7] Arman Kiani B, M. R. Akbarzadeh ―Automatic Text Summarization Using: Hybrid Fuzzy GA-GP‖, IEEE International Conference on Fuzzy Systems. Juky 16-21, 2006.

[8] R. Mihalcea, and P. Tarau, "Textrank: Bringing order into texts,". In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004.

[9] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond,". In Computational Natural Language Learning, 2016.

[10] D. Bahadanau, K. Cho Yoshua Bengio “Neural Machine translation by jointly to align and translate”, Conference paper at ICLR, 2015

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu “ BLEU: a Method for Automatic Evaluation of Machine Translation”, 40th Annual meeting of the Association for Computer Linguistics, 2002

[12] Chin-Yew Lin “ ROUGE: A Package for Automatic Evaluation of Summaries” DUC Conference, 2004

[13] Lee, Dongjun (2018) TensorFlow Text Summarization using seq2seq. Retrieved from, *https://github.com/dongjun-Lee/text-summarization-tensorflow*

[14] Chen, Xianshun (2018) Keras Text Summarization. Retrieved from, *https://github.com/chen0040/keras-text-summarization*