TEXT SUMMARIZATION USING NEURAL NETWORK

A Project report submitted in partial fulfilment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Submitted by

SAI TEJA P	**(316126510111)**
SHAIK AMEEN	**(316126510114)**
G RAVI VARMA	**(316126510078)**
M MOUNISHA	**(316126510092)**

Under esteemed guidance of

Dr. K. Selvani Deepthi
Associate Professor
Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES
(UGC AUTONOMOUS)
(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA &NAAC with 'A' Grade)
SANGIVALASA, Bheemili mandal VISAKHAPATNAM – 531162

MARCH 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

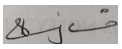
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES (UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA &NAAC with 'A' Grade)

SANGIVALASA, Bheemili mandal VISAKHAPATNAM – 531162

BONAFIDE CERTIFICATE

This is to certify that this project report entitled "TEXT SUMMARIZATION USING NEURAL NETWORK" submitted by "**SAI TEJA POLISETTY (316126510111), SHAIK AMEEN (316126510114), GUNTAMUKKALA RAVI VARMA (316126510078), MAMILAPALLI MOUNISHA (316126510092)**" in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering of Anil Neerukonda Institute of Technology and Sciences (A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.



SIGNATURE

Dr. R. Sivaranjani

HEAD OF THE DEPARTMENT

COMPUTER SCIENCE AND ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES

(AUTONOMOUS)

SIGNATURE

Dr. K Selvani Deepthi

SUPERVISOR

ASSOCIATE PROFESSOR

COMPUTER SCIENCE AND ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES

(AUTONOMOUS)

DECLARATION

We **SAI TEJA P, SHAIK AMEEN, G RAVI VARMA, M MOUNISHA** students of fourth year B.Tech, in Computer Science and Engineering from ANITS, Visakhapatnam, hereby declare that the project work entitled "**TEXT SUMMARIZATION USING NEURAL NETWORK**" is carried out by us and submitted in partial fulfilment of the requirements for the award of **Bachelor of Technology in Computer Science and Engineering**, under Anil Neerukonda Institute of Technology and sciences during the Academic year 2019-2020 and has not been submitted to any other university for the award of any kind of degree.

SAI TEJA P (316126510111)
SHAIK AMEEN (316126510114)
G RAVI VARMA (316126510078)
M MOUNISHA (316126510092)

ACKNOWLEDGEMENT

An endeavour over a long period can be advice and support of many well- wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We owe our tributes to **Dr.R. Sivaranjani, Head of the Department, Computer Science & Engineering** for her valuable support and guidance during the period of project implementation.

We wish to express our sincere thanks and gratitude to our project guide **Dr. K. Selvani Deepthi, Associate Professor, department of computer science and engineering, ANITS**, for the simulating discussions, in analysing problems associated with our project work and for guiding us throughout the project. Project meeting were highly informative. We express our sincere thanks for the encouragement, untiring guidance and the confidence they had shown in us. We are immensely indebted for their valuable guidance throughout our project.

We also thank all the **staff members** of CSE department for their valuable advices.

We also thank **Principal** and supporting staff for providing resources as and when required.

We would like to thank Mrs. B.V. Udaya Lakshmi of the Department of CSE, ANITS for providing great assistance in accomplishment of our project.

SAI TEJA P	316126510111
SHAIK AMEEN	316126510114
G RAVI VARMA	316126510078
M MOUNISHA	316126510092

ABSTRACT

The past decade has endorsed a great rise in Artificial Intelligence. Text summarization which comes under AI has been an important research area that identifies the relevant sentences from a piece of text. By Text Summarization, we can get short and precise information by preserving the contents of the text. This project presents an approach for generating a short and precise extractive summary for the given document of text. A cumulative method using statistical methods and neural network model for extractive text summarization of sports articles using extraction of various features is discussed in this project. The features taken are TF-ISF, Sentence Length, Sentence Position, Sentence to Sentence cohesion, Proper noun, Pronoun. Each sentence is given a score known as the predictive score is calculated and the summary for the given document of text is given based on the predictive score or also known as the rank of the sentence. The accuracy is checked using the DUC dataset and sports articles of various newspapers like the New York Times, CNN. The precision of 73% is acquired when compared with System Generated Summary (SGS) and manual summary, on an average.

Keywords- Artificial Intelligence, Natural Language Processing, Term frequency inverse sentence frequency, cosine similarity, System Generated Summary (SGS).

CONTENTS

TITILE	PAGE NO
ABSTRACT	i
LIST OF FIGURES	iv
LIST OF TABLES	v
1. Introduction	1
1.1 Text Mining	1
1.1.1Infromation Extraction	1
1.1.2 Information Retrieval	2
1.1.3 Categorisation	2
1.1.4 Clustering	3
1.1.5 Summarisation	3
1.2 Natural Language Processing	4
1.2.1 Text Embedding	5
1.2.1.1 Skip-Gram	7
1.2.1.2 Continuous Bag of words	7
1.2.2 Neural Machine Translation	9
1.3 Deep Learning	10
1.3.1 Biological Neural Network	11
1.3.2 Artificial Neural Network	14
1.3.2.1 Single layered perceptron	14
1.3.2.2 Multi layered perceptron	15
1.3.2.3 Recurrent Neural Network	17
1.3.2.4 Convolutional Neural Network	20
1.3.3 Activation Function	22
1.3.3.1 Sigmoid	23
1.3.3.2 Tanh	24
1.3.3.3 ReLU	24
1.3.3.4 Leaky ReLU	25
1.4 Motivation of the work	25
1.5 Problem Statement	26
1.6 Organization of thesis	26
2. Literature Survey	27
2.1 Summarization method using Fuzzy rules	27
2.2 Method using KNN for feature similarity	28
2.3 Deep learning approaches for summarization of legal documents	29
2.4 Method using local scoring and ranking	29
2.5 Summarization of multi-documents using RNN	30
2.6 Summarization using Multi-layered network	31
2.7 Method using sentence ranking	32
2.8 Method using Deep learning	33
2.9 Method using Unsupervised Deep learning techniques	34
2.10 Method using word-embedding	35
2.11 Existing methods for Text Summarization	36
2.12 Different stemming techniques	38

3. Methodology 40
3.1 Proposed System 40
3.1.1Architecture 40
3.1.2 Detailed Understanding of architecture 41
4. Experimental Analysis and results 47
4.1 System Configuration 47
4.1.1Software Requirements 47
4.1.2 Hardware Requirements 50
4.2 Sample Code 53
4.3 Screenshots 61
4.4 Experimental Analysis 64
5. Conclusion and Future work 67
5.1 Conclusion 67
5.2 Future work 67

REFERENCES

LIST OF FIGURES

Fig.No	Topic Name	Page No.
1.1	Comparison of Text Mining Techniques	4
1.2	Text embedding	6
1.3	Text embedding techniques	8
1.4	A Recurrent neural network for Machine Translation	10
1.5	Block diagram of BNN	12
1.6	Biological Neural Network	13
1.7	Single Layered Perceptron	15
1.8	Architecture of multi layered perceptron	17
1.9	Architecture of LSTM	20
1.10	Architecture of CNN	22
1.11	Sigmoid activation function	23
1.12	Comparison of tanh and sigmoid activation function	24
1.13	ReLU and Leaky ReLU activation functions	25
3.1	Neural network architecture	46
4.1	Input text document	61
4.2	Lower casing	61
4.3	Stop word removal	61
4.4	Stemming	62
4.5	TF-ISF values of sentences	62
4.6	Sentence length values of sentences	62
4.7	Proper noun present of not vector	62
4.8	Pronoun vector	62
4.9	Cosine similarity matrix	63
4.10	Sentence containing scores vector	63
4.11	Summary generated by Statistical method	63
4.12	Summary generated by single layered perceptron	63
4.13	Summary after set operation	63
4.14	Comparison between SGS and Online summarizer tool	66

LIST OF TABLES

Table No.	Topic Name	Page No.
CHAPTER 4		
1.	Comparison of performance between manual w.r.t SGS and manual w.r.t online summarizer tool	65
2.	Shows number of sentences in document as well as SGS	66
3.	The average of the Precision(P), recall(R), f-measure(F) for all the 10 documents	66

1. INTRODUCTION

1.1 TEXT MINING

Text mining can also be referred as Text analysis, which uses different Artificial Intelligence technologies. Text mining can be referred to as deriving high-quality information by drawing patterns and identifying the important keywords from the unstructured data. Text mining tasks include text classification which is the classification of the text for example genre, the sentimental analysis which tells about how the author wrote the sentences that is in which tone, document clustering refers to as clustering the documents using unsupervised learning, text summarization for obtaining a short and precise text. Each task is different from each other and has its own probe and methodologies. The purpose is too unstructured information, extract meaningful numeric indices from the text. Thus, make the information contained in the text accessible to the various algorithms. Information can be extracted to derive summaries contained in the documents. Hence, you can analyse words, clusters of words used in documents. In the most general terms, text mining will "turn text into numbers". Such as predictive data mining projects, the application of unsupervised learning methods.

1.1.1 Information Extraction

This is the most famous text mining technique [11]. Information exchange refers to the process of extracting meaningful information from vast chunks of textual data. This text mining technique focuses on identifying the extraction of entities, attributes, and their relationships from semi-structured or unstructured texts. Whatever information is extracted is then stored in a database for future access and retrieval. The efficacy and relevancy of the outcomes are checked and evaluated using precision and recall processes. In most of the cases, this activity includes processing human language texts by means of NLP.

1.1.2 Information Retrieval

Information Retrieval (IR) [11] refers to the process of extracting relevant and associated patterns based on a specific set of words or phrases. In this text mining technique, IR systems make use of different algorithms to track and monitor user behaviours and discover relevant data accordingly. Information retrieval is regarded as an extension to document retrieval. That the documents that are returned are processed to condense. Thus, document retrieval follows by a text summarization stage. That focuses on the query posed by the user. IR systems help in to narrow down the set of documents that are relevant to a problem. As text mining involves applying very complex algorithms to large document collections. Also, IR can speed up the analysis significantly by reducing the number of documents. Google and Yahoo search engines are the two most renowned IR systems.

1.1.3 Categorisation

This is one of those text mining techniques that is a form of "supervised" learning wherein normal language texts are assigned to a predefined set of topics depending upon their content. Categorization in text mining means sorting documents into groups. Automatic document classification uses a combination of natural language processing (NLP) and machine learning to categorize customer reviews, support tickets, or any other type of text document based on their contents. Thus, categorization [11] or rather Natural Language Processing (NLP) is a process of gathering text documents and processing and analysing them to uncover the right topics or indexes for each document. The co-referencing method is commonly used as a part of NLP to extract relevant synonyms and abbreviations from textual data. Today, NLP has become an automated process used in a host of contexts ranging from personalized

commercials delivery to spam filtering and categorizing web pages under hierarchical definitions, and much more.

1.1.4 Clustering

Clustering [11] is one of the most crucial text mining techniques. It seeks to identify intrinsic structures in textual information and organize them into relevant subgroups or 'clusters' for further analysis. A significant challenge in the clustering process is to form meaningful clusters from the unlabelled textual data without having any prior information on them. Cluster analysis is a standard text mining tool that assists in data distribution or acts as a pre-processing step for other text mining algorithms running on detected clusters.

1.1.5 Summarisation

Text summarisation [11] refers to the process of automatically generating a compressed version of a specific text that holds valuable information for the end-user. The aim of this text mining technique is to browse through multiple text sources to craft summaries of texts containing a considerable proportion of information in a concise format, keeping the overall meaning and intent of the original documents essentially the same. Text summarisation integrates and combines the various methods that employ text categorization like decision trees, neural networks, regression models, and swarm intelligence.

There are broadly two different approaches that are used for text summarization:

Extractive text summarization:

The name itself gives what this approach does. We identify the important sentences or phrases from the original text and extract only those from the text. Those extracted sentences would be our summary.

Abstractive summarization:

This is a very interesting approach. Here, we generate new sentences from the original text. This contrasts with the extractive approach we saw earlier where we used only the sentences that were present. The sentences generated through abstractive text summarisation might not be present in the original text.

Technique	Characteristics	Tools
Retrieval	Retrievals valuable information from unstructured text	Intelligent Miner, Text Analyst
Extraction	Extract information from structured database	Text Finder, Clear Forest Text
Summarization	Reduce length by keeping its main points and overall meaning as it is	Tropic Tracking Tool, Sentence Ext Tool
Categorization	Document based categorization	Intelligent Miner
Cluster	Cluster collection of documents, Clustering, classification and analysis of text document	Carrot, Rapid Miner

Fig 1.1: Comparison of Text mining techniques

1.2NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP)is all about computer and human languages interactions. A finest procedure that helps the system in reading and understanding the given text. Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software. Data generated from conversations, declarations or even tweets are examples of unstructured data. Unstructured data doesn't fit neatly into the traditional row and column structure of relational databases and represent most data available in the actual world. It is messy and hard to manipulate. Nevertheless, thanks to the advances in disciplines like machine learning a big revolution is going on regarding this topic. Nowadays it is no longer about trying to interpret a text

or speech based on its keywords (the old-fashioned mechanical way), but about understanding the meaning behind those words (the cognitive way). This way it is possible to detect figures of speech like irony, or even perform sentiment analysis. It is a discipline that focuses on the interaction between data science and human language and is scaling to lots of industries. Today NLP is booming thanks to the huge improvements in the access to data and the increase in computational power, which are allowing practitioners to achieve meaningful results in areas like healthcare, media, finance and human resources, among others.

With the rapid increase in data the text summarization manually consumes a lot of time. That resulted in development of various summarization methods. Using them, the system should deliver a précis with the main essence of the document in reduced size. Text summarization is of two types: i) extractive ii) abstractive. Firstly, the extractive text summarization where important sentences and words from the given text document are identified and those are combined into the summary in a meaningful way. Finally, in abstractive text summarization, new sentences are fabricated, and the summary is given without the loss of information which is understandable. This can be done using different methods that include statistical measures, deep learning techniques with supervised or unsupervised learning and word vector embedding.

1.2.1 Text Embedding

Text Embeddings [13] are real valued vector representations of strings. We build a dense vector for each word, chosen so that it's similar to vectors of words that appear in similar contexts. Word embeddings are considered a great starting point for most deep NLP tasks. They allow deep learning to be effective on smaller datasets, as they are often the first inputs to a deep learning architecture and the most popular way of transfer learning in NLP. The most popular names in word embeddings are Word2vec by Google (Mikolov) and GloVe by Stanford (Pennington, Socher and Manning). Let's delve deeper into these word representations.

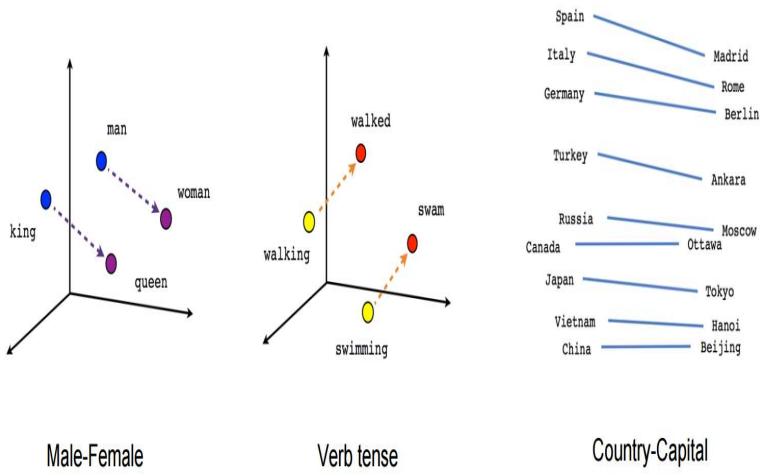


Fig 1.2 Text embedding

In Word2vec**,** we have a large corpus of text in which every word in a fixed vocabulary is represented by a vector. We then go through each position *t* in the text, which has a centre word *c* and context words *o*. Next, we use the similarity of the word vectors for *c* and *o* to calculate the probability of *o* given *c* (or vice versa). We keep adjusting the word vectors to maximize this probability.

For efficient training of Word2vec, we can eliminate meaningless (or higher frequency) words from the dataset (such as a**,** the**,** of**,** then…). This helps improve model accuracy and training time. Additionally, we can use negative sampling for every input by updating the weights for all the correct labels, but only on a small number of incorrect labels.

Word2vec has 2 model variants worth mentioning:

- Skip-gram and Continuous bag of words

1.2.1.1 Skip-Gram [13]:

Skip-gram is one of the unsupervised learning techniques used to find the most related words for a given word. Skip-gram is used to predict the context word for a given target word. It is the reverse of CBOW algorithm. Here, target word is input while context words are output. As there is more than one context word to be predicted which makes this problem difficult. We consider a context window containing *k* consecutive terms. Then we skip one of these words and try to learn a neural network that gets all terms except the one skipped and predicts the skipped term. Therefore, if 2 words repeatedly share similar contexts in a large corpus, the embedding vectors of those terms will have close vectors.

1.2.1.2 Continuous Bag of words [13]:

We take lots and lots of sentences in a large corpus. Every time we see a word, we take the surrounding word. Then we input the context words to a neural network and predict the word in the centre of this context. When we have thousands of such context words and the centre word, we have one instance of a dataset for the neural network. We train the neural network and finally the encoded hidden layer output represents the embedding for a word. It so happens that when we train this over many sentences, words in similar context get similar vectors.

One grievance with both Skip-Gram and CBOW is such that they are both window-based models, meaning the co-occurrence statistics of the corpus are not used efficiently, resulting in suboptimal embeddings. The GloVe model seeks to solve this problem by capturing the meaning of one

word embedding with the structure of the whole observed corpus. To do so, the model One grievance with both Skip-Gram and CBOW is that they're both window-based models, meaning the co-occurrence statistics of the corpus are not used efficiently, resulting in suboptimal embeddings.

The GloVe model seeks to solve this problem by capturing the meaning of one word embedding with the structure of the whole observed corpus. To do so, the model trains on global co-occurrence counts of words and makes a enough use of statistics by minimizing least-squares error and, as a result, produces a word vector space with meaningful substructure. Such an outline sufficiently preserves words' similarities with vector distance.

Besides these 2 text embeddings, there are many more advanced models developed recently, including Fast Text, Poincare Embeddings, sense2vec, Skip-Thought, Adaptive Skip-Gram. I highly encourage you to check them out.

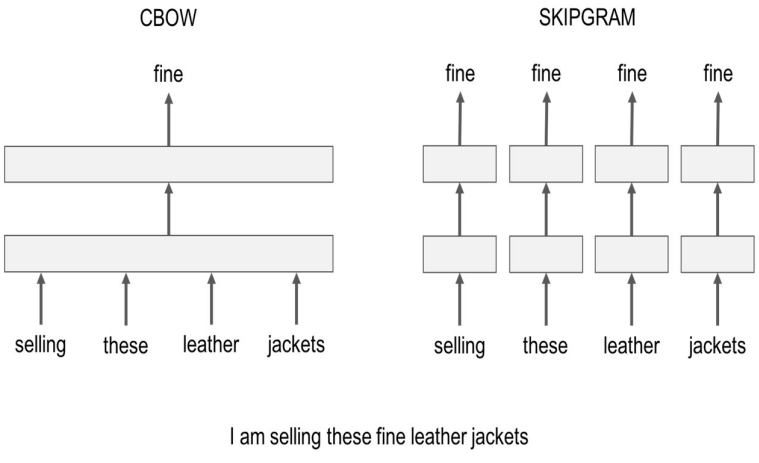


Fig 1.3 Text embedding techniques

1.2.2 Neural Machine Translation

Neural Machine Translation[13] is the approach of modelling this entire process via one big artificial neural network, known as a Recurrent Neural Network (RNN). RNN is a stateful neural network, in which it has connections between passes, connections through time. Neurons are fed information not just from the previous layer but also from themselves from the previous pass. This means that the order in which we feed the input and train the network matters: feeding it “Donald” and then “Trump” may yield different results compared to feeding it “Trump” and then “Donald”.

Neural Machine Translation is a machine translation approach that applies a large artificial neural network toward predicting the likelihood of a sequence of words, often in the form of whole sentences. Unlike statistical machine translation, which consumes more memory and time, neural machine translation, NMT, trains its parts end-to-end to maximize performance. NMT systems are quickly moving to the forefront of machine translation, recently outcompeting traditional forms of translation systems.

Machine translation is the task of automatically converting source text in one language to text in another language. Given a sequence of text in a source language, there is no one single best translation of that text to another language. This is because of the natural ambiguity and flexibility of human language. This makes the challenge of automatic machine translation difficult, perhaps one of the most difficult in artificial intelligence. Classical machine translation methods often involve rules for converting text in the source language to the target language. The rules are often developed by linguists and may operate at the lexical, syntactic, or semantic level. This focus on rules gives the name to this area of study: Rule-based Machine Translation,

or RBMT. The key limitations of the classical machine translation approaches are both the expertise required to develop the rules, and the vast number of rules and exceptions required.

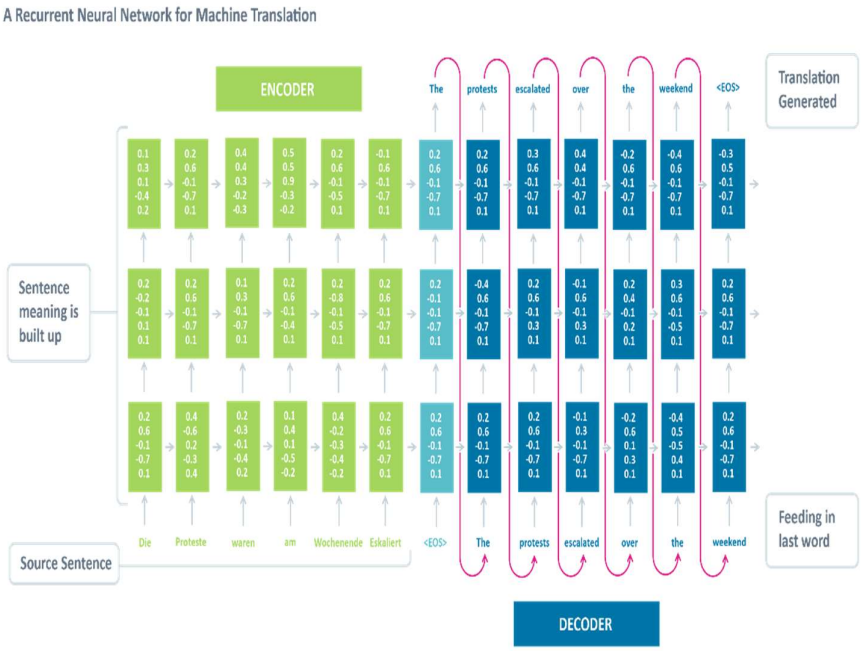


Fig 1.4 A Recurrent neural network for Machine Translation

1.3DEEP LEARNING

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network. It has been evolved from Biological neural network.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free

speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

There are two main reasons it has only recently become useful:

- Deep learning requires large amount of labelled data. For example, driverless car development requires millions of images and thousands of hours of video.
- Deep learning requires substantial computing power. High performance GPU have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

There are many applications of deep learning in different fields:

- Automatic driving
- Aerospace and defence
- Medical research
- Industrial automation

1.3.1 Biological neural network

Computational modelling and theoretical analysis of biological neural networks are integral parts of computational neuroscience. This field's association with cognitive and behavioural modelling is derived from the fact that biological neural systems maintain very close relationships to this modelling. The hope is that through this modelling a link can be established between the data obtained from the biological processes, the probable working mechanism of the biological

network, and learning through statistics and theory of information.

The neural system of the human body consists of three stages: receptors, a neural network, and effectors. The receptors receive the stimuli either internally or from the external world, then pass the information into the neurons in a form of electrical impulses. The neural network then processes the inputs then makes proper decision of outputs. Finally, the effectors translate electrical impulses from the neural network into responses to the outside environment. Figure 2.1 shows the bidirectional communication between stages for feedback.

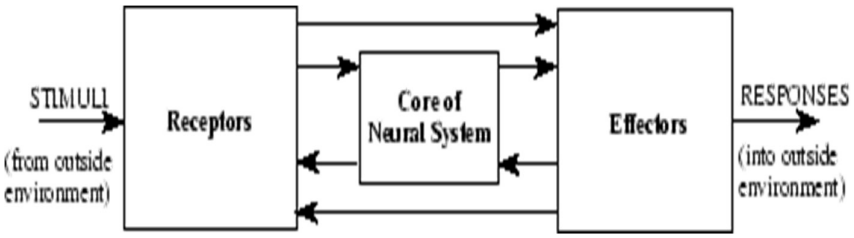
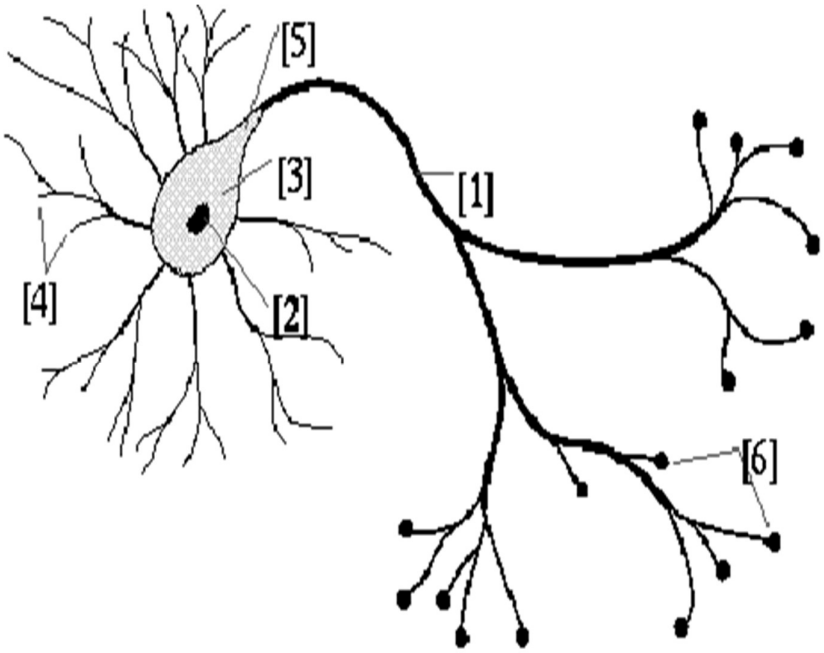


Fig 1.5 Block diagram of BNN

The fundamental element of the neural network is called a neuron. As shown in figure 2.2, a neuron mainly consists of three parts: dendrites, soma, and axon. Dendrites are the tree-like structure that receives the signal from surrounding neurons, where each line is connected to one neuron. Axon is a thin cylinder that transmits the signal from one neuron to others. At the end of axon, the contact to the dendrites is made through a synapse. The inter-neuronal signal at the synapse is usually chemical diffusion but sometimes electrical impulses. A neuron fires an electrical impulse only if certain condition is met.

Understanding neuronal networks of the brain has opened the horizon to the artificial neuronal network software and adaptive systems. These systems vary from single- or double-layered units to multiple-layered units with multiple feedback

connectivity and directions. In order to develop the programming that determine functions, “weights” have been introduced to change the output by changing the parameters and the connectivity of the neurons. Thus, through the variable parameters of connectivity, ANNs can be independent and self-regulated yet teachable systems through external input or indigenous input in the form of dictated standards.



1.Axon 2. Nucleus 3.Soma (Body) 4. Dendrite 5. Axon Hillock 6. Terminals (Synapses)

Fig 1.6 Biological Neural Network

The incoming impulse signal from each synapse to the neuron is either excitatory or inhibitory, which means helping or hindering firing. The condition of causing firing is that the excitatory signal should exceed the inhibitory signal by a certain amount in a short period of time, called the period of latent summation. As we assign a weight to each incoming impulse signal, the excitatory signal has positive weight and the inhibitory signal has negative weight. This way, we can say, A neuron fires only if the total weight of the synapses that receive

impulses in the period of latent summation exceeds the threshold.

1.3.2 Artificial neural network

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values.

1.3.2.1 Single Layered perceptron

Single layer perceptron is the first proposed neural model created. The content of the local memory of the neuron consists of a vector of weights. The computation of a single layer perceptron is performed over the calculation of sum of the input vector each with the value multiplied by corresponding element of vector of the weights. The value which is displayed in the output will be the input of an activation function.

Neural networks are the core of deep learning, a field which has practical applications in many different areas. Today neural networks are used for image classification, speech recognition, object detection etc. Now, Let's try to understand the basic unit behind all this state of art technique.

A single neuron transforms given input into some output. Depending on the given input and weights assigned to each input, decide whether the neuron fired or not. Let's assume the neuron has 3 input connections and one output.

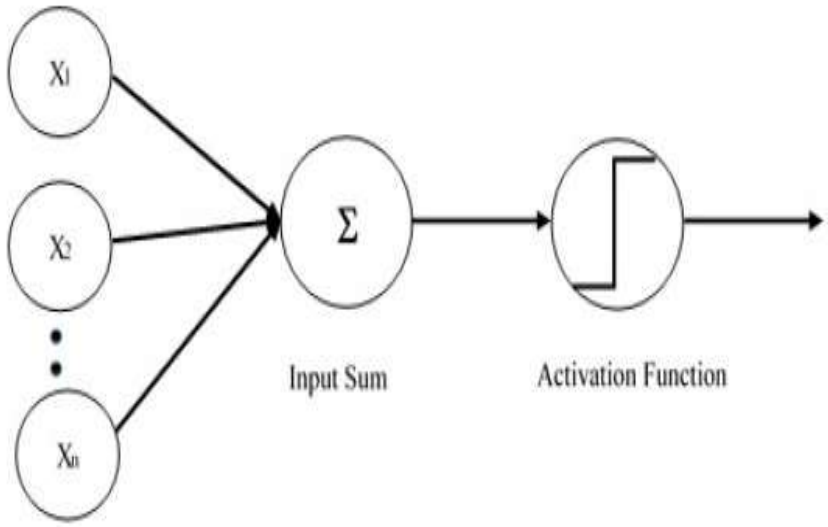


Fig 1.7 Single Layered Perceptron

1.3.2.2 Multi layered perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to refer to *any* feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron (with threshold activation); see &Terminology. Multilayer perceptron are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that decides or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer can approximate any continuous function.

Multilayer perceptron's are often applied to supervised learning problems[3]: they train on a set of input-output pairs and learn to model the correlation (or dependencies) between those inputs and outputs. Training involves adjusting the parameters, or the weights and biases, of the model in order to minimize error. Backpropagation is used to make those weigh and bias adjustments relative to the error, and the error itself can be measured in a variety of ways, including by root mean squared error (RMSE).

Feedforward networks such as MLPs are like tennis, or ping pong. They are mainly involved in two motions, a constant back and forth. You can think of this ping pong of guesses and answers as a kind of accelerated science, since each guess is a test of what we think we know, and each response is feedback letting us know how wrong we are.

In the forward pass, the signal flow moves from the input layer through the hidden layers to the output layer, and the decision of the output layer is measured against the ground truth labels.

In the backward pass, using backpropagation and the chain rule of calculus, partial derivatives of the error function w.r.t. the various weights and biases are backpropagated through the MLP. That act of differentiation gives us a gradient, or a landscape of error, along which the parameters may be adjusted as they move the MLP one step closer to the error minimum. This can be done with any

gradient-based optimisation algorithm such as stochastic gradient descent. The network keeps playing that game of tennis until the error can go no lower. This state is known as convergence.

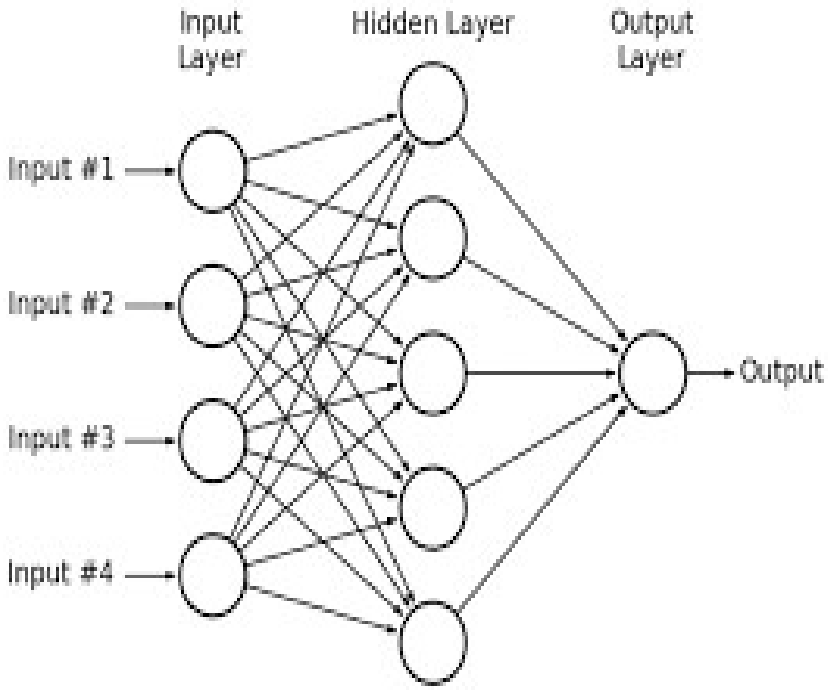


Fig 1.8 Architecture of multi layered perceptron

1.3.2.3 Recurrent neural network

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.[1] This makes them applicable to tasks such as unsegmented, connected handwriting_recognition or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks

with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behaviour. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

Recurrent Neural Network remembers the past and it's decisions are influenced by what it has learnt from the past. Note: Basic feed forward networks "remember" things too, but they remember things they learnt during training. For example, an image classifier learns what a "1" looks like during training and then uses that knowledge to classify things in production.

It's part of the network. RNNs can take one or more input vectors and produce one or more output vectors and the output(s) are influenced not just by weights applied on inputs like a regular NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series.

In summary, in a vanilla neural network, a fixed size input vector is transformed into a fixed size output vector. Such a network becomes "recurrent" when you repeatedly apply the transformations to a series of given input and produce a series of output vectors. There is no pre-set limitation to the size of the vector. And, in addition to generating the output which is a function of the input and hidden state, we update the hidden sate itself based on the input and use it in processing the next input.

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. So, if you are trying to process a paragraph of text to

do predictions, RNN's may leave out important information from the beginning. During back propagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update a neural networks weight. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning. So, in recurrent neural networks, layers that get a small gradient update stops learning. Those are usually the earlier layers. So, because these layers don't learn, RNN's can forget what it seen in longer sequences, thus having a short-term memory.

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behaviour required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field. There are few that are better at clearly and precisely articulating both the promise of LSTMs and how they work than the experts that developed them. So, in recurrent neural networks, layers that get a small gradient update stops learning. Those are usually the earlier layers. So because these layers don't learn, RNN's can forget what it seen in longer sequences, thus having a short-term memory.

These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state-of-the-art results based on recurrent neural networks are achieved with these two networks. LSTM's and GRU's can be found in speech

recognition, speech synthesis, and text generation. You can even use them to generate captions for videos.

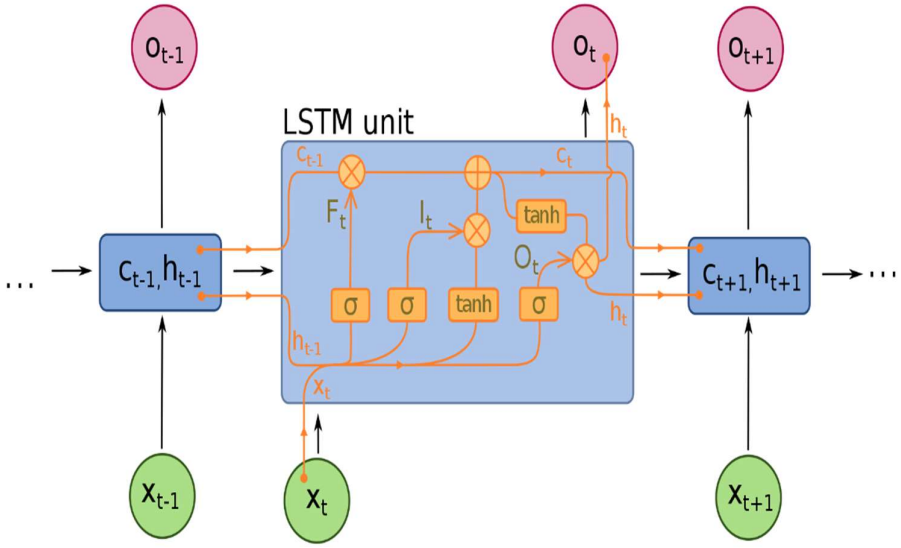


Fig 1.9 Architecture of LSTM

1.3.2.4 Convolutional neural network

CNNs are regularized versions of multilayer perceptron. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes[6][7][8][9] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical

neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and for other auto correlated data. A convolution is essentially sliding a filter over the input.

The most common use for CNNs is image classification, for example identifying satellite images that contain roads or classifying hand-written letters and digits. There are other quite mainstream tasks such as image segmentation and signal processing, for which CNNs perform well at. A CNN can also be implemented as a U-Net architecture, which are essentially two almost mirrored CNNs resulting in a CNN whose architecture can be presented in a U shape. U-nets are used where the output needs to be of similar size to the input such as segmentation and image improvement.

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are

linear transformations; each convolution is a type of affine function.

In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two-dimensional matrix) and a 3x3 convolutional kernel (a two-dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time, this is detailed further below.

In other non-vision applications, a one-dimensional convolution may slide vertically over an input matrix.

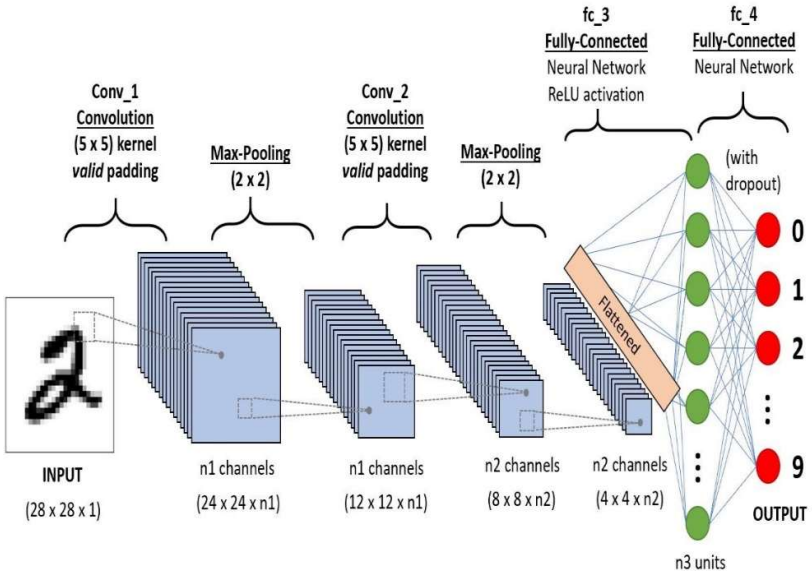


Fig 1.10 Architecture of Convolutional neural network

1.3.3 Activation Function

Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding

bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

1.3.3.1 Sigmoid

The Sigmoid [12] Function curve looks like a S-shape. The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we must predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points. function is monotonic but function's derivative is not. The logistic sigmoid function can cause a neural network to get stuck at the training time. The SoftMax function is a more generalized logistic activation function which is used for multiclass classification.

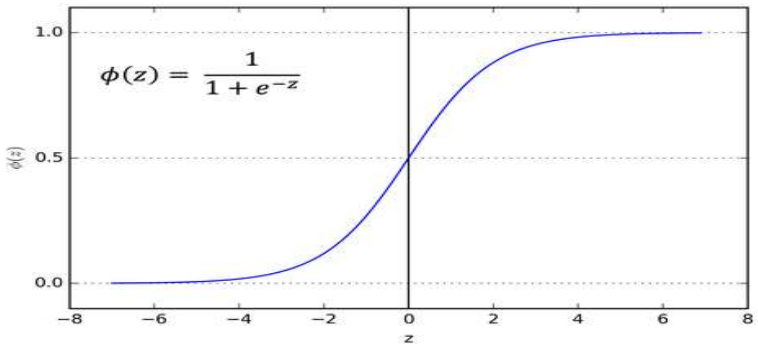


Fig 1.11 Sigmoid activation function

1.3.3.2 Tanh

Tanh [12] is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. The function is differentiable.

The function is monotonic while its derivative is not monotonic. The tanh function is mainly used classification between two classes.

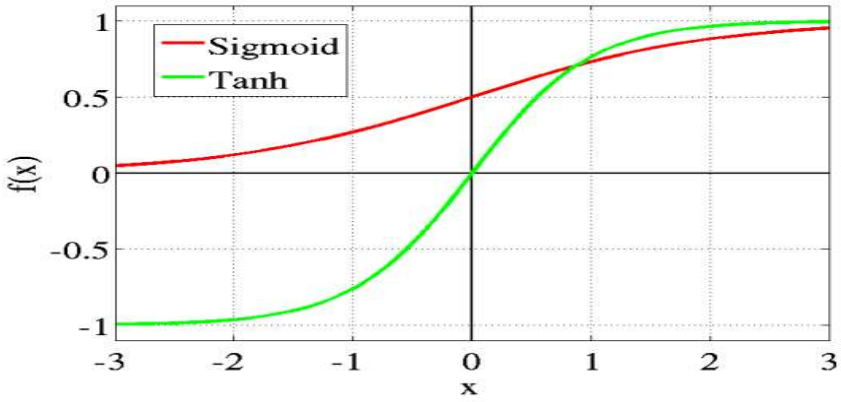


Fig 1.12 Comparison of tanh and sigmoid activation function

1.3.3.3 ReLU

The ReLU [12] is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning. As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: [0 to infinity)

The function and its derivative both are monotonic. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

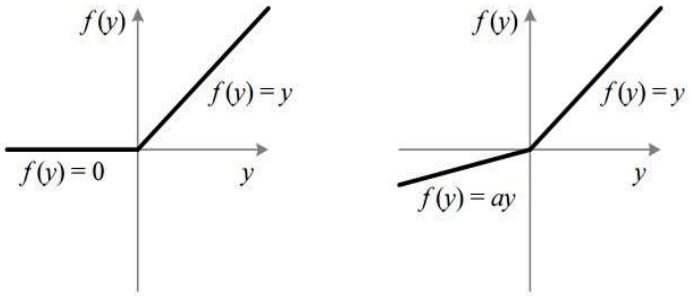


Fig 1.13 ReLU and Leaky ReLU activation functions

1.3.3.4 Leaky ReLU

The leak helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so. When a is not 0.01 then it is called Randomized ReLU. Therefore, the range of the Leaky ReLU is (-infinity to infinity). Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

1.4 MOTIVATION OF THE WORK

Text summarization has become one of the hottest topics in Text mining as there is more amount of text that is produced on the internet. So, has become so difficult to read all the content due to the busy schedules and as the time has become the more important constraint in everyone's life. To make the text content simpler to read, there came an idea of summarizing

the text by giving only the important information from the available text articles. The sports are the most admiring topics in news, and everyone wants to know about the sports news and due to the time constraint, they couldn't read the whole news which contains only a few important information. So, we have chosen to summarize the sports articles.

1.5 PROBLEM STATEMENT

Generating a summary from a text document. So that it helps in understanding a huge text data. With the availability of internet at every corner of the world, the amount of information is growing rapidly at certain rate. Considering the hectic schedule of the people and the need for information abstraction, here comes the summarization of the News Articles. Every domain in the news has its own type of information important. So, as it is difficult to summarize all types of news data. We are limiting our problem to extractive summarize the sports genre in news articles.

1.6 ORGANIZATION OF THESIS:

In this document, Chapter 2 consists of Literature survey, Chapter 3 consists of Proposed method and its architecture, Chapter 4 consists of Experimental analysis and results, Chapter 5 consists of Conclusion and future work.

2. LITERATURE SURVEY

2.1 Summarization method using Fuzzy rules:

Fabio Bif Goularte et.al (2018) [1] proposed a method using fuzzy rules to retrieve the most important sentences in a document. The text summarization task has gained much importance because of the large amount of online data, and its potential to extract useful information and knowledge in a way that could be easily handled by humans and used for a myriad of purposes, including expert systems for text assessment. This paper presents an automatic process for text assessment that relies on fuzzy rules on a variety of extracted features to find the most important information in the assessed texts. The automatically produced summaries of these texts are compared with reference summaries created by domain experts. Differently from other proposals in the literature, our method summarizes text by investigating correlated features to reduce dimensionality, and consequently the number of fuzzy rules used for text summarization. Thus, the proposed approach for text summarization with a relatively small number of fuzzy rules can benefit development and use of future expert systems able to automatically assess writing. The pro- posed summarization method has been trained and tested in experiments using a dataset of Brazilian Portuguese texts provided by students in response to tasks assigned to them in a Virtual Learning Environment (VLE). The proposed approach was compared with other methods including a naive baseline, Score, Model and Sentence, using ROUGE measures. The results show that the proposal provides better f-measure (with 95% CI) than methods. This paper presented a method that uses fuzzy metrics to deter- mine the most informative sentences in the text, and then compare and classify text written in response to learning tasks with reference responses for these tasks. Fuzzy logic was used because it is a widely recommended approach for applications that handle un- certain information (e.g., ambiguity in linguistic terms) (Das, 2013; Ross, 2010). We show that fuzzy logic associated with text summarization can represent the main

concepts in natural language on student's assessment. The observed percentile is 73 percentiles.

2.2Method using KNN for feature similarity:

Taeho Jo et.al (2017) [2] proposed a method that uses the feature vector of certain features and obtains the correlation between the vectors. we propose a version of KNN (K Nearest Neighbour) where the similarity between feature vectors is computed considering the similarity among attributes or features as well as one among values. The task of text summarization is viewed into the binary classification task where each paragraph or sentence is classified into the essence or non- essence, and in previous works, improved results are obtained by the proposed version in the text classification and clustering. In this research, we define the similarity which considers both attributes and attribute values, modify the KNN into the version based on the similarity, and use the modified version as the approach to the text summarization task. As the benefits from this research, we may expect the more compact representation of data items and the better performance. Therefore, the goal of this research is to implement the text summarization algorithm which represents data items more compactly and provides the more reliability. The proposed approach should be applied and validated in the specialized domains: engineering, medicine, science, and law, and it should be customized to the suitable version. We may consider similarities among only some essential features rather than among all features, to cut down the computation time. We develop and combine various schemes of computing the similarities among features. By adopting the proposed approach, we will develop the text summarization system as a real version. The total observed percentile is 65 percentage.

2.3Deep learning approaches for summarization of legal documents:

Deepa Anand et.al (2019) [3] proposed a method for summarizing the Indian legal judgment documents for this the author has used a neural network approach that is semi-supervised. The availability of legal judgment documents in digital form offers numerous opportunities for information extraction and application. Automatic summarization of these legal texts is a crucial and a challenging task due to the unusual structure and high complexity of these documents. Previous approaches in this direction have relied on huge labelled datasets, using hand engineered features, leveraging on domain knowledge and focussed their attention on a narrow sub-domain for increased effectiveness. In this paper, we propose simple generic techniques using neural network for the summarization task for Indian legal judgment documents. We explore two neural network architectures for this task utilizing the word and sentence embeddings for capturing the semantics. The main advantage of the proposed approaches is that they do not rely on hand crafted features, or domain specific knowledge, nor is their application restricted to a particular sub-domain thus making them suitable to be extended to other domains as well. We tackle the problem of unavailability of labelled data for the task by assigning classes/scores to sentences in the training set, based on their match with reference summary produced by humans. The experimental evaluations establish the effectiveness of our proposed approaches as compared with other baselines. A full Python Implementation of the ROUGE Metric (not a wrapper). Retrieved from in this paper we presented a data driven semi-supervised approach to extractive legal document summarization using. The observed percentile is 63 percentage.

2.4Method using local scoring and ranking:

Krishnaveni et.al (2017) [4] proposed a heading-based text summarization since the context can be known from the heading. Existence of large amount of textual information available on the internet emerged serious research in the area of machine generated summarization. Manual

summarization of these online text documents is a very difficult task for human beings. So, we need an automatic text summarizer. Automatic Text Summarization (ATS) is “condensing the source text into a shorter version, while preserving its information content and overall meaning”. Even though the work of automatic text summarization started in 1950’s, still it is lacking to achieve more coherent and meaningful summaries. The proposed approach provides automatic feature based extractive heading wise text summarizer to improve the coherence thereby improving the understandability of the summary text. It summarizes the given input document using local scoring and local ranking that is it provides heading wise summary. Headings of a document give contextual information and permit visual scanning of the document to find the search contents. The proposed approach applies the same features to all document sentences. But it ranks the sentences heading wise and selects top n sentences from each heading where n depends upon compression ratio. The final heading wise summary produced by this approach is a collection of summaries of individual headings. Since the heading wise summary contains the equal proportion of sentences from each heading, it reduces the coherent gap of the summary text. Also, it improves the overall meaning and understanding of the summary text. The outcomes of the experiment clearly show that heading wise summarizer provides better precision, recall and f-measure over the main summarizer. An automatic extractive text summarizer should produce the summary quickly with no redundancy or minimum redundancy. Two kinds of techniques can be used for summary evaluation. They are intrinsic evaluation and extrinsic evaluation. Here we used intrinsic evaluation. In the proposed approach, we implement an automatic extractive text and percentile is noticed as 70 percentage.

2.5Summarization of multi-documents using RNN:

Jingqiang Chen et.al (2018) [5] proposed a multi-model neural network-based extractive summarizer. Rapid growth of multi-modal documents containing images on the Internet makes multi-modal summarization

necessary. Recent advances in neural-based text summarization show the strength of deep learning technique in summarization. This paper proposes a neural-based extractive multi-modal summarization method based on multi-modal RNN. This method first encodes documents and images with a multi-modal RNN, and then calculates the summary probability of sentences through a logistic classifier using text coverage, text redundancy, and image set coverage as features. We extend the Daily Mail corpora by collecting images from the Web. Experiments show our method outperforms the state- of-the-art neural summarization methods We proposed a neural multi-modal extractive summarization method that outperforms state-of-the-art neural summarization methods. The sentences and documents are encoded by a hierarchical bi-directional RNN model. The vector representation of images is extracted by VGGNet. The image set is encoded by a bi-directional RNN model. We treated document summarization as a sentence classification problem and adopted the logistic classifier Training text coverage, and text redundancy, and image set coverage as features computed in RNN and CNN. Percentage of accuracy for the above explained is 60 percentiles.

2.6Summarization using Multi-layered network:

Jorge V. Tohalino et.al (2018) [6] has proposed a multi-layered neural network model to generate the extractive summary based on the sentences having high importance in the context for multi-documents. A graph which consists of nodes and edges is built with sentences and words respectively. The datasets used are CST News for Portuguese multi-document summarization and DUC-2002, DUC-2004 for English multi-document summarization. The word embedding feature is lacking in this work for generating a better summary. Huge volumes of textual information have been produced every single day. In order to organize and understand such large datasets, in recent years, summarization techniques have become popular. These techniques aim at finding relevant, concise and non-redundant content from such a big data. While network methods

have been adopted to model texts in some scenarios, a systematic evaluation of multilayer network models in the multi-document summarization task has been limited to a few studies. Here, we evaluate the performance of a multilayer-based method to select the most relevant sentences in the context of an extractive multi document summarization (MDS) task. In the adopted model, nodes represent sentences and edges are created based on the number of shared words between sentences. Differently from previous studies in multi- document summarization, we make a distinction between edges linking sentences from different documents (inter-layer) and those connecting sentences from the same document (intra-layer). As a proof of principle, our results reveal that such a discrimination between intra- and inter-layer in a multi-layered representation is able to improve the quality of the generated summaries. This piece of information could be used to improve current statistical methods and related textual models. The main contribution of this paper consisted in using a multilayer representation to address the multi-document summarization problem. While networked representations of texts have been used to model documents, to our knowledge, no previous study has used the multilayer approach to address this type of problem. The main advantage of the proposed technique is that it does not depend on any source of external information, such as lexical datasets or other deeper resources. As the main result, we showed – as proof of principle – that different weighting schemes for inter-layer edges can better represent a set of documents. Such an improvement in the model turned out to increase the informativeness of the generated summaries.

2.7Method using sentence ranking:

J.N.Madhuri et.al (2019) [7]has proposed a method to generate an extractive summary using sentence ranking technique using the term frequency after the removal of stop words. It works for any kind of text but cannot semantically distinguish sentences. The evaluation is done using the

MSWord summarizer and human summarized summary. The summaries are converted to mp3 format so that it is easy to evaluate or know the summary. Automatic Text summarization is the technique to identify the most useful and necessary information in a text. It has two approaches 1) Abstractive text summarization and 2) Extractive text summarization. An extractive text summarization means an important information or sentence are extracted from the given text file or original document. In this paper, a novel statistical method to perform an extractive text summarization on single document is demonstrated. The method extraction of sentences, which gives the idea of the input text in a short form, is presented. Sentences are ranked by assigning weights and they are ranked based on their weights. Highly ranked sentences are extracted from the input document, so it extracts important sentences which directs to a high-quality summary of the input document and store summary as audio. Automatic text summarization is a complex task which contains many sub-tasks in it. Every subtask has an ability to get good quality summaries. The important part in extractive text summarization is identifying necessary paragraphs from the given document. In this work we proposed extractive based text summarization by using statistical novel approach based on the sentences ranking the sentences are selected by the summarizer. The sentences which are extracted are produced as a summarized text and it is converted into audio form. The proposed model improves the accuracy when compared traditional approach. The observed percentile is 60.

2.8 Method using Deep learning:

Nikhil S. Shirwandkar et.al (2018) [8] has proposed an approach using both Restricted Boltzmann machine and Fuzzy logic to identify the important sentences. An approach for generating short and precise summaries for long text documents is proposed. Lately, the size of information on the internet is increasing. It has become tough for the users to dig into the loads of information to analyse it and draw conclusions. Text summarization solves this problem by generating a summary, selecting sentences which are most important from the document without losing the

information. In this work, an approach for Extractive text summarization is designed and implemented for single document summarization. It uses a combination of Restricted Boltzmann Machine and Fuzzy Logic to select important sentences from the text keeping the summary meaningful and lossless. The text documents used for summarization are in English language. Various sentence and word level features are used to provide meaningful sentences. Two summaries for each document are generated using Restricted Boltzmann Machine and Fuzzy logic. Both summaries are then combined and processed using a set of operations to get the final summary of the document. The results show that the designed approach overcomes the problem of text overloading by generating an effective summary. In this work, RBM is used as an unsupervised learning algorithm along with fuzzy logic for improving the accuracy of the summary. It is observed that the proposed approach generates short and precise summaries without any irrelevant text. Using features like Sentence-Centroid similarity and thematic words has improved the connectivity of the sentences. Using the proposed method, on an average 88% precision, 80% recall and 84% F measure is obtained. The results produced using the proposed method give better evaluation parameters in comparison with prevailing RBM method. The proposed method can be extended for multi document summarization. Documents in different languages can be summarized. Various other features can be used, and the method can be combined with other methods for improving the nature of the summary. Also, it can be used in Abstractive text summarization.

2.9 Method using Unsupervised Deep learning techniques:

Mahmood Yousefi-Azar et.al (2016) [9] has proposed a text summarizer for a query-oriented system using an unsupervised deep learning network which is a deep auto-encoder to calculate feature space from term frequency. This is a stochastic machine. The experiments are done on SKE and BC3 email datasets. A semi-supervised learning approach works better because it can be used for unlabelled data. We present methods

of extractive query-oriented single-document summarization using a deep auto- encoder (AE) to compute a feature space from the term-frequency (tf) input. Our experiments explore both local and global vocabularies. We investigate the effect of adding small random noise to local tf as the input representation of AE and propose an ensemble of such noisy AEs which we call the Ensemble Noisy Auto-Encoder (ENAE). ENAE is a stochastic version of an AE that adds noise to the input text and selects the top sentences from an ensemble of noisy runs. In each individual experiment of the ensemble, a different randomly generated noise is added to the input representation. This architecture changes the application of the AE from a deterministic feed-forward network to a stochastic runtime model. Experiments show that the AE using local vocabularies clearly provide a more discriminative feature space and improves the recall on average 11.2%. The ENAE can make further improvements, particularly in selecting informative sentences. To cover a wide range of topics and structures, we perform experiments on two different publicly available email corpora that are specifically designed for text summarization. This technique may be useful because of the limited availability of manually annotated data, or noisy labelling could be used with the un- labelled data. Other ensemble approaches should also be investigated to improve performance including accuracy.

2.10 Method using word-embedding:

Aditya Jain et.al (2017) [10] has proposed a neural network-based text summarization, the input of the nodes is the values of different features and the dataset used is DUC-2002. The summary is compared with four online summarizers. The accuracy of summarization can be exalted using a large training dataset for a neural network. These days, text summarization is an active research field to identify the relevant information from large documents produced in various domains such as finance, news media, academics, politics, etc. Text summarization is the process of shortening the documents by preserving the important contents of the text. This can be achieved through extractive and abstractive summarization. In this paper,

we have proposed an approach to extract a good set of features followed by neural network for supervised extractive summarization. Our experimental results on Document Understanding Conferences 2002 dataset show the effectiveness of the proposed method against various online extractive text summarizers. In this paper we presented an extensive text summarization approach using neural network. The neural network has been trained by extracting ten features including word vector embedding from the training dataset. Testing has been performed on DUC 2002 dataset, where upto 284 documents were used in various test experiments. ROUGE scores (1, 2, and L) computed for our proposed model and four of the online text summarizers show the effectiveness of the proposed model. Performance of the proposed model may further be improved by increasing the size and diversity of the training dataset and applying more effective approaches [10] to convert the abstractive summaries into extractive summaries. The percentile we observed is 68.

2.11 Existing methods for Text Summarization

2.11.1 Ranking of sentences using Cosine similarity

Cosine similarity is a metric used to determine how similar the documents are irrespective of size. Mathematically, it measures the cosine of the angle between the two vectors projected in a multi-dimensional space.

This is advantageous because even if two similar documents are far apart by Euclidian distance, they could have still had a similar smaller angle between them. Smaller angle means high similarity.

$$cosine\ similarity = \frac{A.B}{|A||B|}$$

Example:

D1: the best Italian restaurant enjoy the best pasta

D2: American restaurant enjoy the best hamburger

D3: Korean restaurant enjoy the best bibimbap

D4: the best the best American restaurant

Using Cosine similarity: (comparing similarity with D4)

D1: 0.82

D2: 0.77

D3: 0.65

D4: 1

2.11.2 Multi document text summarization using TF-IDF

TF-IDF stands for term frequency-inverse document frequency. This is a statistical measure used to evaluate how important a word is to a document in collection of corpora. The importance increases proportionally to the number of times a word appears in the document in the document but the offset by the frequency of the word in the corpus.

- TF: Term Frequency**,** which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization. TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
- IDF: Inverse Document Frequency**,** which measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = loge(Total number of documents / Number of documents with term t in it).

D1: 0

D2: 0.5

D3: 0

D4: 1

2.12 Different stemming techniques:

i. Potters Stemmer:

It is one of the most popular stemming methods proposed in 1980. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. For Example: EED -> EE means "if the word has at least one vowel and consonant plus EED ending, change the ending to EE" as 'agreed' becomes 'agree'.

Advantage: Less error rate.

Limitation: Morphological variants are not always real words.

ii. Lovins Stemmer:

It is proposed by Lovins in 1968, that removes the longest suffix from a word then word is recoded to convert this stem into valid words.

Example: sitting -> sitt -> sit

Advantage: Fast and handles irregular plurals.

Limitation: Time consuming.

iii. Dawson Stemmer:

It is extension of Lovins stemmer in which suffixes are stored in the reversed order indexed by their length and last letter.

Advantage: Fast and covers more suffixes.

Limitation: Complex to implement.

iv. Krovertz Stemmer:

It was proposed in 1993 by Robert Krovetz

1) Convert the plural form of a word to its singular form.

2) Convert the past tense of a word to its present tense and remove suffixing.

Example: ‘children’ -> ‘child’

Advantage: Used as pre-stemmer for other stemmers.

Limitation: Inefficient for large documents.

v. Xerox Stemmer:

Advantage: Works well in case of large documents.

Limitation: Language dependent

vi. N-gram Stemmer:

An n-gram is a set of n consecutive characters extracted from a word in which similar words will have a high proportion of N-grams in common.

Example: ‘INTRODUCTIONS’ for n=2 becomes : *I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S*

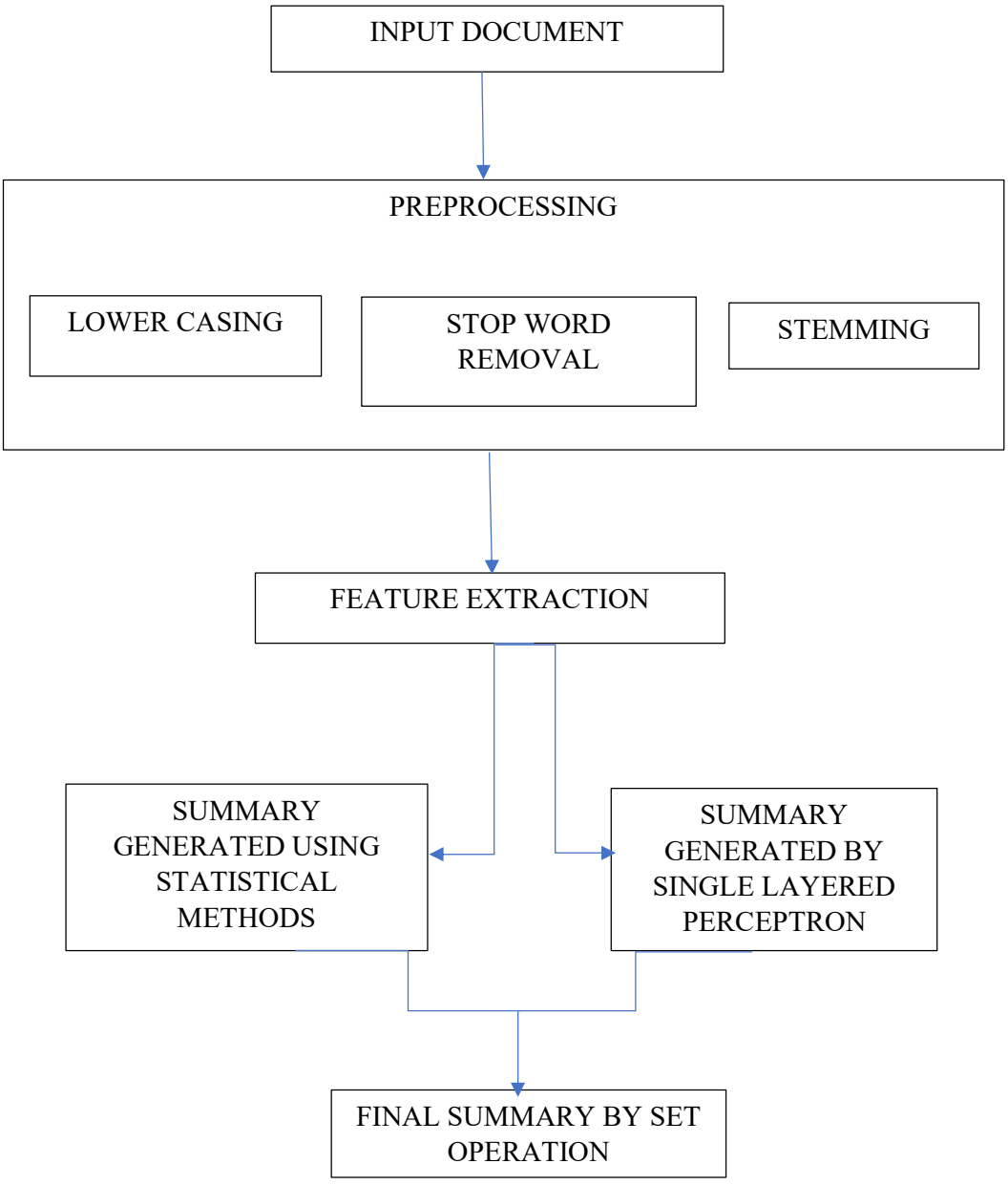
Advantage: Based on string comparison.

Limitation: Requires extra space to create and index the N-grams and is not time efficient.

3. METHODOLOGY

3.1PROPOSED SYSTEM

3.1.1 ARCHITECTURE



3.1.2 Detailed Understanding of architecture:

1) Input Document:

   To generate a summary some input should be given in the file format ".txt". The text documents of sports articles are considered for summarization and are valid since the author has proposed a method for summarizing sports articles. The input document should only be in English language. The size of the input document can be large as well.

2) Pre-processing:

   The input text is divided into sentences based on the sentence terminator. These sentences are individually pre-processed using the below three techniques:

   a. Lower casing:

      In the lower casing, the whole document is converted into lowercase alphabetical letters. This is done to remove the ambiguity of the words with different casing. This is done using the lower() of python language which is available inbuilt. There is another method also, considering each character and its ascii values. Add the required value to convert to lower case.

   b. Stop word removal:

      The stop words in the document are removed from the document because these are the most frequent words like articles, prepositions, conjunctions which don't add any value in defining the importance of a sentence. The stop words are removed using nltk (natural language tool kit) library. All the stop words are stored in a list and each word is compared with the words present in the list. If the word is present in the list it is removed from the document, else it is remained unaltered.

c. Stemming:

Stemming is referred as getting the words reduced to their root form. It removes the different forms of the same word present in different sentences. In this porters stemmer is used.

3) Feature Extraction:

a. TF-ISF:

Term frequency and inverse document frequency (TF-IDF) are used for information retrieval systems. In this paper, the summarization is done for a single document of sports article and Term frequency inverse sentence frequency (TF-ISF) is calculated as follows.

$$TF - ISF(i,j) = TF(j) * \log\left(\frac{N}{ISF(j)}\right) \quad (1)$$

$$TF - ISF(i) = \sum_{j=0}^{n} \frac{(\text{TF}-\text{ISF(i,j)})}{n} \quad (2)$$

Where TF_j indicates the term frequency of a word in a sentence, ISF_j is the inverse sentence frequency of a j^{th} word in a sentence, TF-ISF_i(2) is the Term frequency and inverse sentence frequency of i^{th} sentence in a document, n cites to the number of words in a sentence, N cites to the number of sentences in a document.

b. Sentence length:

This feature provides the less weightage to short sentences as short sentences are relatively less important when compared to long sentences. It is measured by calculating the ratio of no of words in the sentence to the no of words in the longest sentence of the document. This feature's value is calculated using eq. (3).

$$sentence\ length = \frac{no\ of\ words\ in\ sentence}{no\ of\ words\ in\ longest\ sentence} \qquad (3)$$

c. Sentence position:

In a document the numerical value of the sentence's position is gauged (4). The position is evaluated as the normalized percentile score in the range of 0 to 1.

$$sentence\ position = \begin{cases} 0, & if\ first\ sentence \\ 1, & if\ last\ sentence \\ \frac{p}{n}, & if\ 0 < p < n \end{cases} \qquad (4)$$

p is the position of the current sentence and n is the total number of sentences.

d. Cosine similarity:

Analogy between sentences is calculated using this property. This briefs how much similar this sentence with other sentences in a document is. A vector for each sentence is assessed and (5) is practiced obtaining the score of the feature.

$$cosine\ similarity = \frac{A.B}{|A||B|} \qquad (5)$$

A is the first vector and B is the second vector with which it is compared.

e. Existence of proper noun:

Proper names broaching to places and people might be useful to decide the relevance of a sentence. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.

For Example: New York, Monday, Johnathan, etc.

f. Existence of pronoun:

Pronouns cites to the persons which are described in the previous sentences. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.

For Example: He, she, it, they.

g. Sentence containing scores:

Since in sports the score of the team or the individual participant is important the score is taken into consideration. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.

4) Summary generated using Statistical methods:

The mean, median and mode are generally some statistical tools used. The mean is used in many cases.

Firstly, the input document is divided in to sentences and then the sentences are divided into words. Secondly, the pre-processing which consists of lower casing, stop word removal and stemming is done and stored in a list.

Then TF-ISF for each word is calculated using equation (1) and (2). But we need the TF-ISF for each sentence. So, for a sentence there may be 'n' words then we get 'n' different values, we calculate the mean of the 'n' values to obtain total TF-ISF for the sentence.

Sentence length is calculated using (3). We get a value for each sentence.

Sentence position is calculated using (4). We get a value for each sentence.

Cosine similarity is calculated by converting the words to vectors and each word has a unique vector. The similarity between the sentence is calculated using equation (5). Like TF-ISF each word has its own similarity value. To obtain a value for each sentence we calculate the mean of all the words of a sentence.

Based on all these seven-feature values for each sentence. We take the mean of TF-ISF and cosine similarity of each sentence and considering the sentence position, sentence length, existence of proper noun, existence of pronoun, and a sentence with score. We give each sentence a predictive score which is also called as the rank of the sentence.

Compression ratio is defined as the percentage of the original input should be present in the summary.

All the predictive scores of sentences are sorted in descending order. The sentence with highest predictive score is taken and added to the summary in the order of the original document to not loose the context of what author is trying to convey.

The final summary for the compression ratio is stored and the sentence number is also stored along with it for easy identification of the sentence.

5) Summary generated using Single layered perceptron:

The single layered perceptron is used as a neural network. It consists of one input layer, one hidden layer and an output layer. Input layer has 7 nodes and output layer has one node. The values of the features become the input to the neural network which is multiplied by weights and it is passed through activation function, sigmoid is chosen and we get a value for each sentence and it becomes the predictive score of a sentence.

Weight matrix= [0.8, 0.6, 0.4, 0.2, 0.3, 0.5, 0.7]

For each sentence:

Input: [x1, x2, x3, x4, x5, x6, x7]

Weight: [w1, w2, w3, w4, w5, w6, w7]

Sum:

(x1*w1+x2*w2+x3*w3+x4*w4+x5*w5+x6*w6+x7*w7)

$$a = \frac{1}{1 + pow(e, -sum)}$$

The value of 'a' is sorted and the sentences with highest value for given compression ratio is given as summary.

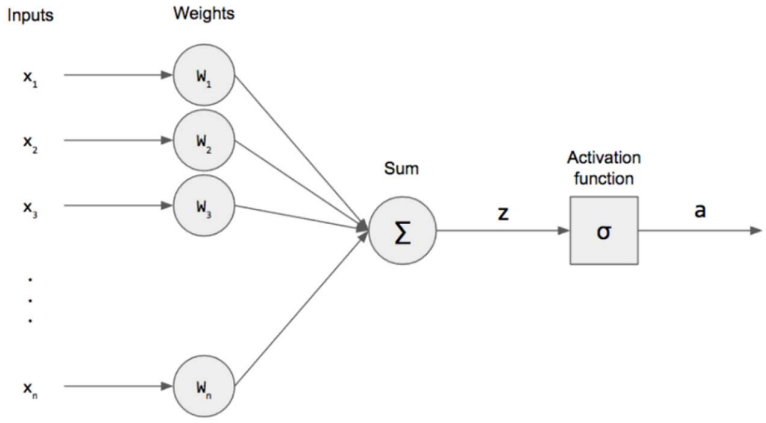


Fig 3.1: Neural network architecture (Single layered perceptron)

The summary is stored along with the number of the sentence stored.

6) Final summary by set operation:

The common sentences are taken from both the techniques and the summary is given based on the compression ratio considering it to be 50%.

4. EXPERIMENTAL ANALYSIS AND RESULTS

4.1 System Configuration

4.1.1 Software Requirements

a. Python:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.
Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable.

Characteristics of Python:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to bytecode for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.

Reasons why python is used:

- Readable and Maintainable code
- Multiple programming paradigms
- Compatible with major platforms and systems
- Robust standard library
- Many open source frameworks and tools

- Simplify complex software development
- Adopt test driven development
- Highly productive compared to other programming languages like C++ and Java.

b. Pycharm Framework:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License.

Features:

- Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
- Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others
- Support for web frameworks: Django, web2py and Flask [professional edition only] [9]
- Integrated Python debugger
- Integrated unit testing, with line-by-line code coverage
- Database tools

- Python web frameworks
- Scientific Stack support
- Interactive python console
- Plugins
- Cross platform IDE
- Customisable UI
- VCS, Development and Remote Development

c. Natural language toolkit:

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cookbook and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac

OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analysing linguistic structure, and more. The online version of the book has been updated for Python 3 and NLTK 3.

4.1.2 Hardware Requirements

a. Processor:

The first trade-off we came across was which chip to buy? i7 processors seemed to have lesser cores compared to Xeon processors, but higher frequency. i7 processors are also very easy to overclock, which is not the case with Xeon. Xeon on the other hand, supported much higher RAM – i7 maxed out at 128 GB RAM (that too was available on a single model). Also, the power consumption of Xeon is lower compared to an i7. We started shortlisting possible processor chips, which would fit our requirement. We came down to a shortlist of a few i7 and Xeon processors. We decided to go for higher cores, power efficiency and possibility to expand more RAM in future over high frequency i7 chips. After searching for prices of various chips and their benchmarks, we finally zeroed in on Intel Xeon E2630 v4 .

b. RAM:

We bought 128 GB RAM for now. We an obviously add more, if required. This should be good for now. Needless to say DDR4 is better than DDR3 and 2400 MHz is better than 2133 MHz. We ended up with 2133 MHz to save a few bucks.

c. Motherboard:

Once the processor is finalized, we looked for compatible motherboards. Since we wanted to use this machine for deep learning, we had to make sure that the motherboard had enough PCIe slots. We wanted to have the option to add up to 2 GPUs in future. Additional things to look out for included the number of RAM slots, which would directly impact the amount of RAM you can have. You can also look at whether the motherboard support one socket or 2 sockets. Having 2 sockets is more desirable as you can add one more processor to the box later. We zeroed in on ASRock EPC612D8A – while it had only one socket, it gave us option to expand RAM further and add more GPUs, if needed in future.

d. GPU:

Given the evolution in deep learning, we knew that we had to invest in the best in class GPU. The options include NVidia GTX 1080, NVidia Tesla K40. Again, we went to the basics on why we need GPUs. For deep learning, we do not need high precision computations, so the expensive Tesla K series went out of consideration. Among the two remaining Titan X Pascal clearly looked far superior for deep learning. There was only one problem – this card was

not available in India yet. I had to ask one of my friends in the US to get it for us.

e. Hard disk:

The first thing we did was to buy a SSD and a hard disk. The idea was to do the frequent read write operations on the SSD, while keep the large datasets on the normal hard disk. We also decided to not go with RAID based hard disks as they were adding to the costs and most of the time, we have backups of the data anyways. So, we took a 512 GB SSD + 2 TB HDD (7200 RPM)

f. Power:

We added a 1000W Corsair SMPS to make sure that the GPU and the motherboard get ample power.

g. Cooling:

Cooling is very important for a workstation like this. Ideally, liquid based cooling systems are the most effective. However, we could not find a compatible model. So, we went with Intel Heatsink which was the next best option. I was a bit sceptical whether this would be sufficient or not. However, we have not faced any heating problems till now. The Fan on GPU, heat sink and 2 of them on CPU have been effective to keep the temperature under control.

4.2Sample Code

- Dividing the sentence from file

```
for line in f:
    l=line.split(".")
    m.append(l)
print(m)
sentence=[]
for i in range(len(m)):
    for j in range(len(m[i])):
        sentence.append(m[i][j])
```

- Lowercasing

```
for i in range(len(sentence)):
        sentence[i]=sentence[i].lower()
print(sentence)
```

- Stop word removal

```
words=[]
for i in range(len(sentence)):
    words.append(sentence[i].split(" "))
print(words)
words_after_stopword_removal=[[] for x in range(len(words))]
print(words_after_stopword_removal)
for i in range(len(words)):
    for j in range(len(words[i])):
        if(words[i][j] not in stopwords):
            words_after_stopword_removal[i].append(words[i][j])
print(words_after_stopword_removal)
```

- Stemming

```
words_after_stemming=[[] for x in range(len(words))]
for i in range(len(words_after_stopword_removal)):
    for j in range(len(words_after_stopword_removal[i])):

words_after_stemming[i].append(ps.stem(words_after_stopwo
rd_removal[i][j]))
print(words_after_stemming)
```

- TF-ISF

```
tfisf=[[] for x in range(len(words))]
N=len(words_after_stemming)
for i in range(len(words_after_stemming)):
    for j in range(len(words_after_stemming[i])):

tf=words_after_stemming[i].count(words_after_stemming[i][j]
)
        s=words_after_stemming[i][j]
        c=0
        for x in range(len(words_after_stemming)):
            if(s in words_after_stemming[x]):
                c+=1
        isf=math.log((N/c),10)
        tfisf[i].append(tf*isf)
print(tfisf)
tfisf_for_each_sentence=[[] for x in range(len(words))]
for i in range(len(tfisf)):
    c=0
    for j in range(len(tfisf[i])):
        c+=tfisf[i][j]
    tfisf_for_each_sentence[i].append(c)
print(tfisf_for_each_sentence)
```

- Sentence Length

```
count_of_words_in_sentence=[[] for x in range(len(words))]
sentence_length=[[] for x in range(len(words))]
maximum_length=0
for i in range(len(words_after_stemming)):
    c=len(words_after_stemming[i])
    count_of_words_in_sentence[i].append(c)
    if(maximum_length<c):
        maximum_length=c
print(count_of_words_in_sentence)
print(maximum_length)
for i in range(len(count_of_words_in_sentence)):

sentence_length[i].append(count_of_words_in_sentence[i][0]/
maximum_length)
print(sentence_length)
```

- Sentence position

```
pos=[[0] for x in range(len(words))]
num=len(words)
print(num)
for i in range(1,num+1):
    pos[i-1][0]=(i/num)
print(pos)
```

- Proper noun

```
if_propernoun_available=[[0] for x in range(len(words))]
print(if_propernoun_available)
for i in range(len(words_after_stemming)):
    flag=0
    for j in range(len(words_after_stemming[i])):
        l=pos_tag(words_after_stemming[i][j].split())
```

```
        print(l)
        if(l[0][1]=='NN'):
            flag=1
            break
        elif(l==[]):
            continue
    if(flag==1):
        if_propernoun_available[i][0]=1
        flag=0
print(if_propernoun_available)
```

- Proper noun

```
if_pronoun_available=[[0] for x in range(len(words))]
print(if_pronoun_available)
for i in range(len(words_after_stemming)):
    flag=0
    for j in range(len(words_after_stemming[i])):
        l=pos_tag(words_after_stemming[i][j].split())
        if(l[0][1]=='PRP'):
            flag=1
            break
    if(flag==1):
        if_pronoun_available[i][0]=1
        flag=0
print(if_pronoun_available)
```

- Sentence containing score

```
score=[[0] for x in range(len(words))]
flag=0
flag1=0
for i in range(len(words_after_stemming)):
    flag=0
```

```
    for j in range(len(words_after_stemming[i])):
        flag1=0
        for k in range(len(words_after_stemming[i][j])):
            if(('0' in words_after_stemming[i][j][k]) or ('1' in
words_after_stemming[i][j][k])or ('2' in
words_after_stemming[i][j][k]) or ('3' in
words_after_stemming[i][j][k]) or ('4' in
words_after_stemming[i][j][k]) or ('5' in
words_after_stemming[i][j][k]) or  ('6' in
words_after_stemming[i][j][k]) or  ('7' in
words_after_stemming[i][j][k]) or  ('8' in
words_after_stemming[i][j][k]) or  ('9' in
words_after_stemming[i][j][k])):
                flag=1
                flag1=1
                break
        if(flag1==1):
            flag1=0
            break
    if(flag==1):
        score[i][0]=1
        flag=0
    else:
        score[i][0]=0
print("score:\n",score)
```

- Cosine similarity

```
cosine_similarity=[[] for x in range(len(words))]
print(cosine_similarity)
for i in range(len(vector)):
    vec1=vector[i]
    print(vec1)
    for j in range(len(vector)):
```

```
        vec2=vector[j]
        dot=0
        for k in range(len(vec1)):
            dot+=(vec1[k]*vec2[k])
        c1=0
        for k in range(len(vec1)):
            c1+=(vec1[k]**2)
        c1=(c1**0.5)
        c2=0
        for k in range(len(vec2)):
            c2+=(vec2[k]**2)
        c2=(c2**0.5)
        try:
            cosine_similarity[i].append((dot)/(c1*c2))
        except ZeroDivisionError:
            cosine_similarity[i].append(0.0)
    print(cosine_similarity)

    final_cosine=[[0] for x in range(len(words))]
    for i in range(len(cosine_similarity)):
        p=0
        for j in range(len(cosine_similarity[i])):
            p+=cosine_similarity[i][j]
        final_cosine[i][0]=(p/len(cosine_similarity[i]))
    print(final_cosine)
```

- Statistical summary

```
    mean_of_tfisf_and_cs = [[0] for x in range(len(words))]

    for i in range(len(cosine_similarity)):

    mean_of_tfisf_and_cs[i][0]=(cosine_similarity[i][0]+tfisf_for_
    each_sentence[i][0])/2
```

```
print(mean_of_tfisf_and_cs)

dict = {}
for i in range(len(mean_of_tfisf_and_cs)):
    dict[i] = mean_of_tfisf_and_cs[i][0]

sorted_dict = sorted(dict.items(), key=lambda kv:
kv[1],reverse=True)
compression_ratio=int(len(mean_of_tfisf_and_cs)*0.5)
print(compression_ratio)
output_lines=[]
for i in range(compression_ratio):
    output_lines.append(sorted_dict[i][0])
print(output_lines)

output_lines.sort()
print(output_lines)

output=""
for i in output_lines:
    output=output+m[0][i]+". "
print(output)
```

- Summary generated from single layered perceptron

```
input_multiplied_by_weight = [[0] for x in range(len(words))]

for i in range(len(words)):

input_multiplied_by_weight[i][0]=(w1*tfisf_for_each_sentenc
e[i][0])+(w2*sentence_length[i][0])+(w3*if_propernoun_avail
able[i][0])+(w4*if_pronoun_available[i][0])+(w5*pos[i][0])+(
w6*score[i][0])+(w7*final_cosine[i][0])

print(input_multiplied_by_weight)
```

```
sigmoid_activation_func = [[0] for x in range(len(words))]
for i in range(len(words)):

sigmoid_activation_func[i][0]=1/(1+math.exp(input_multiplie
d_by_weight[i][0]*-1))

print(sigmoid_activation_func)

dict1 = {}
for i in range(len(mean_of_tfisf_and_cs)):
   dict1[i] = sigmoid_activation_func[i][0]

print(dict1)

sorted_dict1 = sorted(dict1.items(), key=lambda kv: kv[1],
reverse=True)
output_lines1 = []
for i in range(compression_ratio):
   output_lines1.append(sorted_dict1[i][0])
print(output_lines1)

output_lines1.sort()
print(output_lines1)

output1 = ""
for i in output_lines1:
   output1 = output1 + m[0][i] + ". "
print(output1)
```

- Summary after set operation

```
total_lines=output_lines+output_lines1
lines_after_set=list(set(total_lines))
print(lines_after_set)

output_afer_set = ""
```

```
for i in lines_after_set:
    output_afer_set = output_afer_set + m[0][i] + ". "
print(output_afer_set)
```

4.3 Screenshots

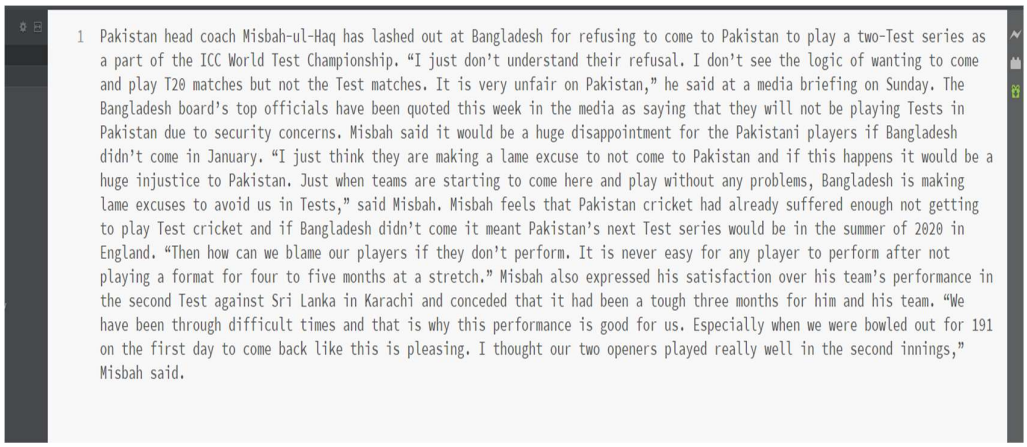


Fig 4.1 Input text document

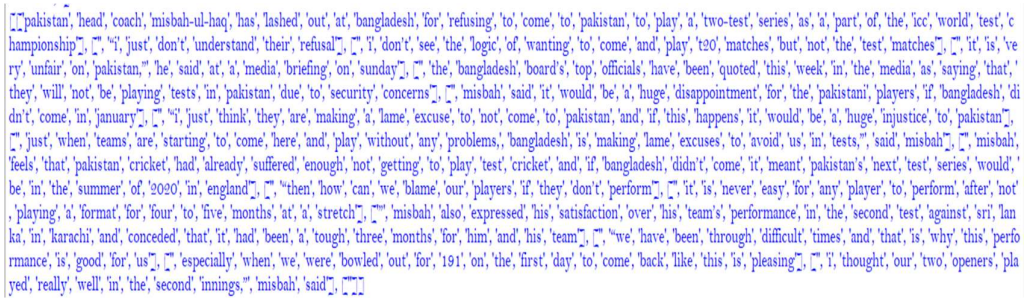


Fig 4.2 Lower casing

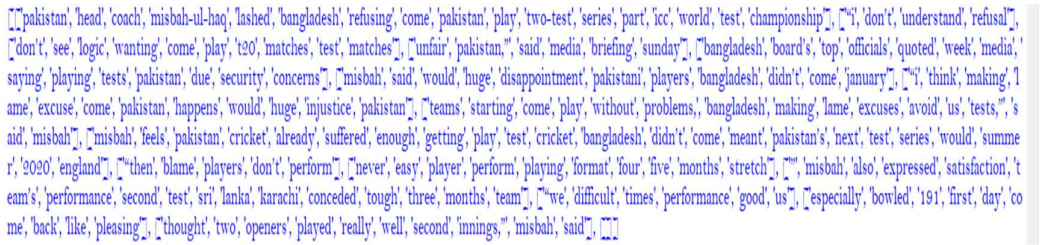


Fig 4.3 Stop word removal

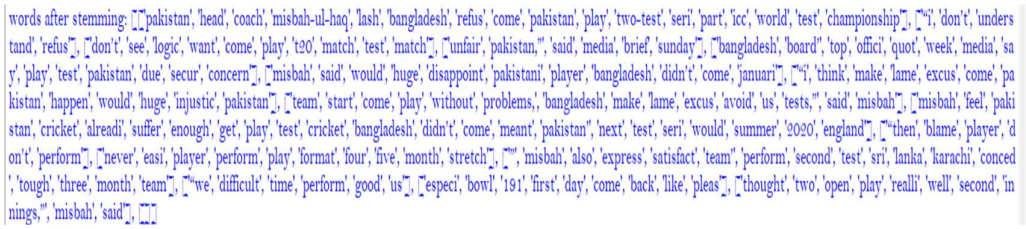


Fig 4.4 Stemming

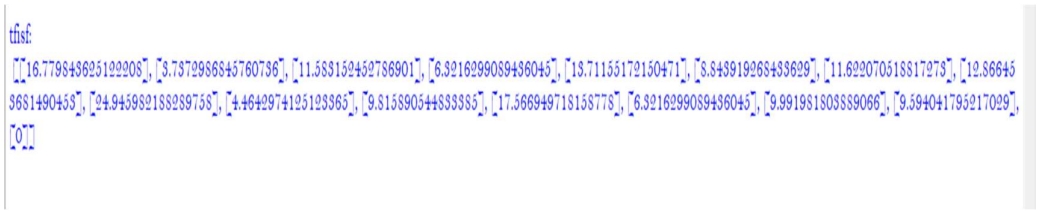


Fig 4.5 TF-ISF values of sentences

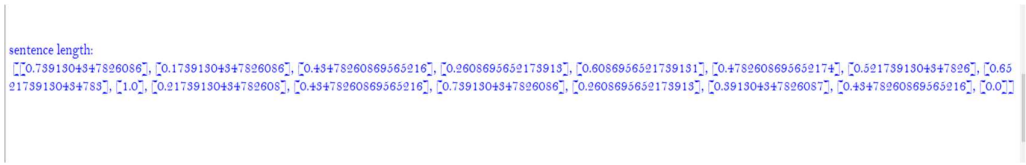


Fig 4.6 Sentence Length values of sentences

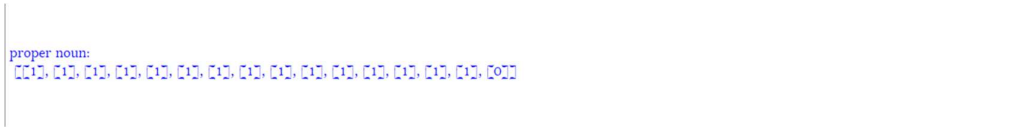


Fig 4.7 Proper noun present of not vector

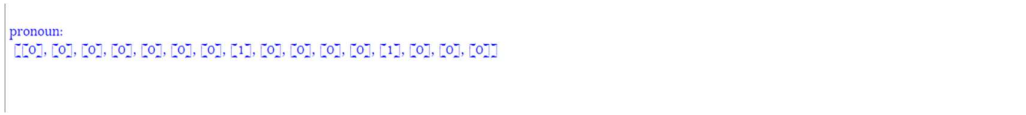


Fig 4.8 Pronoun vector

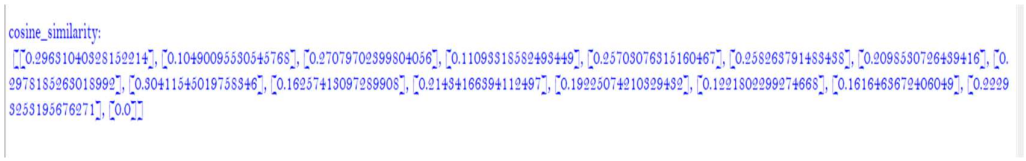


Fig 4.9 Cosine similarity matrix

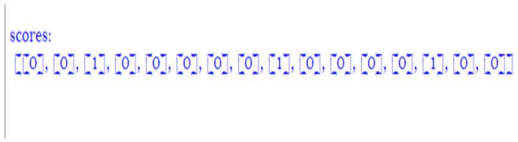


Fig 4.10 Sentence containing scores vector

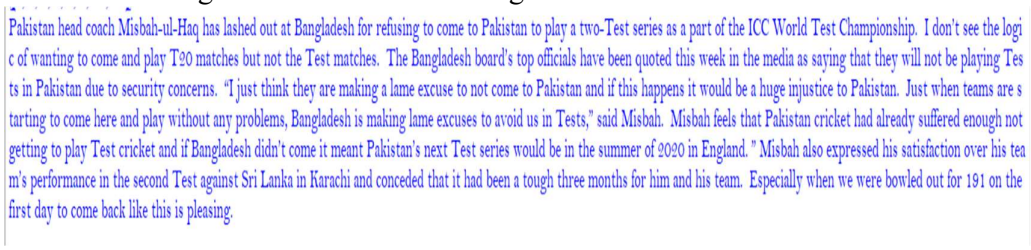


Fig 4.11 Summary generated by Statistical method

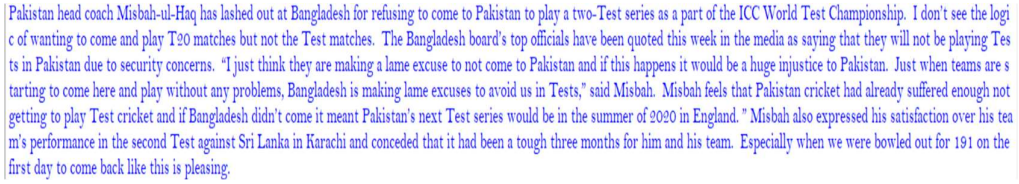


Fig 4.12 Summary generated by single layered perceptron

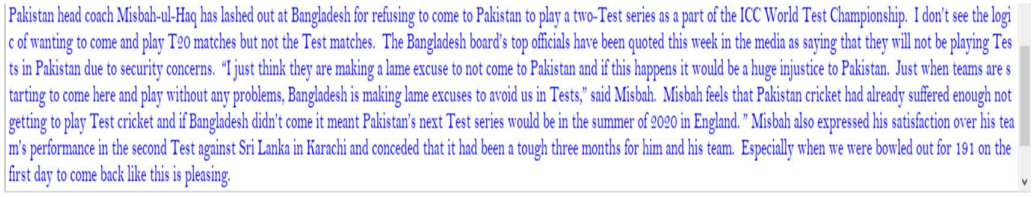


Fig 4.13 Summary after set operation

4.4 Experimental Analysis

The evaluation measures used are for context evaluation which comes under co-selection. In co-selection there are three types of measures they are precision, recall, f-measure.

Precision (P) can be defined as the fraction of the number of sentences common in manual and automated summary to the number of sentences in the automated summary. Precision (P) can be procured using equation (6).

$$P = \frac{S(manual) \cap S(auto)}{S(auto)} \quad (6)$$

Recall (R) can be defined as the ratio of the number of sentences common in manual and automated summary to the manual summary. The recall can be calculated using equation (7).

$$R = \frac{S(manual) \cap S(auto)}{S(manual)} \quad (7)$$

F-measure (F) can be defined as the harmonic mean of precision and recall. F-measure (F) can be procured using equation (8).

$$F = \frac{2 * P * R}{P + R} \quad (8)$$

Dataset:

The dataset used is the BBC sports corpus since it is the worldwide famous for its news. It consists of 737 articles for five different types of sports namely athletics, cricket, football, rugby, tennis and 101, 124, 265, 147, 100 articles respectively. The model is tested for these articles and performed with a good amount of accuracy by generating the relevant context leaving the unnecessary sentences.

The System generated summary (SGS) is tested for accuracy with a manual summary and an online summarizer tool. The manual summary is given by one of the literates who have secured a degree

in Literature. The online tool used for generating the summary is

Title	Manual w.r.t SGS			Manual w.r.t online summarizer tool		
	P	R	F	P	R	F
Doc-1	0.8	0.8	0.8	0.667	0.8	0.7274
Doc-2	0.7142	0.625	0.6662	0.6667	0.5	0.5714
Doc-3	0.7142	0.8333	0.7689	0.5	0.667	0.5714
Doc-4	0.75	0.75	0.75	0.667	0.5	0.5714
Doc-5	0.6842	0.7647	0.7219	0.7334	0.6470	0.6874
Doc-6	0.7142	0.7142	0.7142	0.5714	0.5714	0.5714
Doc-7	0.7	0.7368	0.7245	0.834	0.834	0.834
Doc-8	0.7142	0.7692	0.7401	0.5834	0.5384	0.56
Doc-9	0.8	0.889	0.8336	0.667	0.667	0.667
Doc-10	0.75	0.8	0.7741	0.5625	0.6	0.58

Table 1: Comparison of performance between manual w.r.t SGS and manual w.r.t online summarizer tool

Text Compactor. The author has considered the compression ratio half of the original document.

Firstly, a manual summary is compared with SGS and a manual summary is compared with online summarizer tool for precision, recall, and f-measure. The documents tested are the sports articles from various newspapers including the New York Times, CNN. For analysing the accuracy, the author also checked it with DUC-2002.

The Table-1 shows the number of sentences in each document taken from the news articles. The data of 10 documents is shown.

Title	Number of sentences in document	Number of sentences in SGS (α=50%)
Doc-1	11	5

Doc-2	16	7
Doc-3	15	7
Doc-4	8	4
Doc-5	36	19
Doc-6	15	7
Doc-7	39	20
Doc-8	27	14
Doc-9	19	10
Doc-10	32	16

Table-2: Shows number of sentences in document as well as SGS.

The evaluation measures of all the documents are given below.

	Average		
	P	R	F
Manual w.r.t SGS	0.7341	0.7681	0.7493
Manual w.r.t online summarizer tool	0.6452	0.6328	0.6341

Table-3: The average of the Precision(P), recall(R), f-measure(F) for all the 10 documents

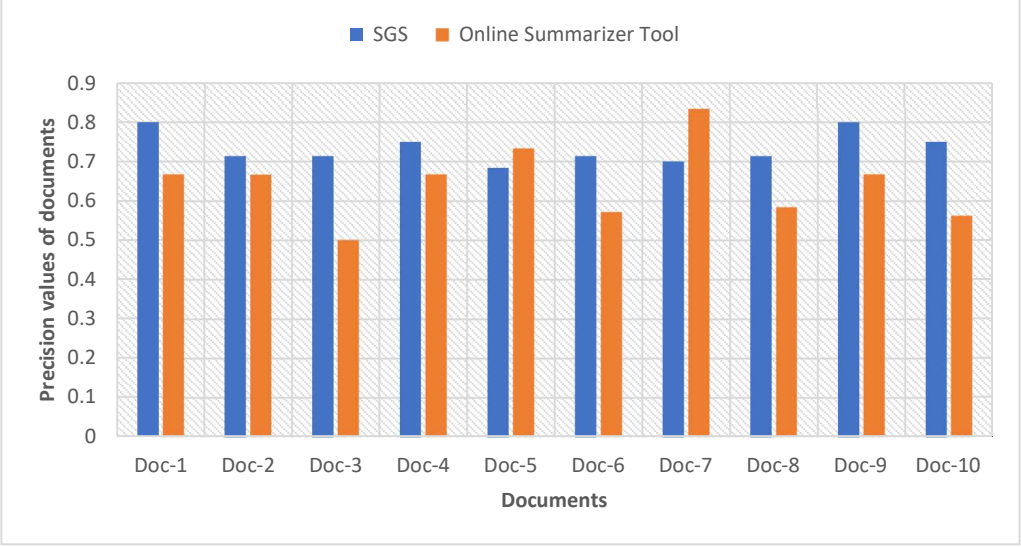


Fig.4.14 Comparison between SGS and Online summarizer tool

5. CONCLUSION AND FUTURE WORK

5.1Conclusion:

The extractive text summarization has many tasks involved in generating the précis. The author proposed a statistical approach for generating an extractive précis of a sports articles that uses sentence ranking based on the selected features for the sentences. The most critical part is identifying the important sentences without loss of meaning from the given document. Using the proposed method on an average, 73% precision, 76% recall, and 74% f-measure is obtained which when compared with a manual generated summary and an online summarizer tool.

5.2 Future work:

In nearly future, the proposed model can also be developed to a neural network model that takes the values of different features as input for generating an accurate precis by considering the contextual meaning. To increase the accuracy of the precis, various features can also be added.

REFERENCES:

[1] Anand, Deepa & Wagh, Rupali. (2019). Effective Deep Learning Approaches for Summarization of Legal Texts. Journal of King Saud University - Computer and Information Sciences. 10.1016/j.jksuci.2019.11.015.

[2] Azar, Mahmood & Hamey, Len. (2016). Text Summarization Using Unsupervised Deep Learning. Expert Systems with Applications. 68. 10.1016/j.eswa.2016.10.017.

[3] Goularte, Fábio & Nassar, Silvia & Fileto, Renato & Saggion, Horacio. (2018). A Text Summarization Method based on Fuzzy Rules and applicable to Automated Assessment. Expert Systems with Applications. 115. 10.1016/j.eswa.2018.07.047.

[4]https://heartbeat.fritz.ai/the-7-nlp-techniques-that-will-change-how-you-communicate-in-the-future-part-i-f0114b2f0497

[5] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[6] https://www.upgrad.com/blog/what-is-text-mining-techniques-and-applications/

[7] Jain, Aditya & Bhatia, Divij & Thakur, Manish. (2017). Extractive Text Summarization Using Word Vector Embedding. 51-55. 10.1109/MLDS.2017.12.

[8] J. Chen and H. Zhuge, "Extractive Text-Image Summarization Using Multi-Modal RNN," *2018 14th International Conference on Semantics, Knowledge and Grids (SKG)*, Guangzhou, China, 2018, pp. 245-248. doi: 10.1109/SKG.2018.00033

[9] J. N. Madhuri and R. Ganesh Kumar, "Extractive Text Summarization Using Sentence Ranking," *2019 International Conference on Data Science and Communication (IconDSC)*, Bangalore, India, 2019, pp. 1-3. doi: 10.1109/IconDSC.2019.8817040

[10] Jo, Duke Taeho. (2017). K nearest neighbor for text summarization using feature similarity. 1-5. 10.1109/ICCCCEE.2017.7866705.

[11] N. S. Shirwandkar and S. Kulkarni, "Extractive Text Summarization Using Deep Learning," *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 2018, pp. 1-5. doi: 10.1109/ICCUBEA.2018.8697465

[12] P. Krishnaveni and S. R. Balasundaram, "Automatic text summarization by local scoring and ranking for improving coherence," *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, 2017, pp. 59-64. doi: 10.1109/ICCMC.2017.8282539

[13] Valverde Tohalino, Jorge & Amancio, Diego. (2017). Extractive Multi-document Summarization Using Multilayer Networks. Physica A: Statistical Mechanics and its Applications. 503. 10.1016/j.physa.2018.03.013.

Extractive Text Summarization using Deep Learning

Nikhil S. Shirwandkar
Student, Mtech Electronics Engineering
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India
nikhilshirwandkar@gmail.com

Dr. Samidha Kulkarni
Associate Professor, Dept. of Electronics Engineering
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India
samidhakulkarni@somaiya.edu

***Abstract*— An approach for generating short and precise summaries for long text documents is proposed. Lately, the size of information on the internet is increasing. It has become tough for the users to dig into the loads of information to analyze it and draw conclusions. Text summarization solves this problem by generating a summary, selecting sentences which are most important from the document without losing the information. In this work, an approach for Extractive text summarization is designed and implemented for single document summarization. It uses a combination of Restricted Boltzmann Machine and Fuzzy Logic to select important sentences from the text still keeping the summary meaningful and lossless. The text documents used for summarization are in English language. Various sentence and word level features are used to provide meaningful sentences. Two summaries for each document are generated using Restricted Boltzmann Machine and Fuzzy logic. Both summaries are then combined and processed using a set of operations to get the final summary of the document. The results show that the designed approach overcomes the problem of text overloading by generating an effective summary.**

Keywords*— *Deep Learning, Extractive, Fuzzy logic, RBM, Sentence Features, Single document, Summarization, Unsupervised.

I. INTRODUCTION

Earlier, humans used to summarize the text by their own, but today due to increasing data, it is difficult for the human beings to cope up with the huge data. To overcome this issue, text summarization is needed. Text summarization generates succinct form of large documents keeping most of the original info. It is carried out on single or multiple documents of similar kind. Based on the nature of summary, it is classified as Extractive, wherein the generated summary consists of the most important sentences of the document put together and Abstractive, wherein new sentences are formed from the scratch to produce the summary. Text summarization has increased the attention of the researchers since 1950. H. P. Luhn was the first to make an automatic text summarization system. It was grounded on the frequency of terms [1]. In 1958, he stated significance of words based of their frequency measures. According to him, words which have a medium frequency i.e. which neither occur much frequently nor less frequently are important. He also removed the stop-words from the text. In 1958, Baxendale proposed salient features based on the sentence position [2]. He stated that the beginning 7% of the document and the last sentences have most of the information. In 1969, Edmundson suggested new method of automatic text summarization which included two fresh features in addition to the position and term which were pragmatic words (cue words), title and heading words [3]. In 1995, Kupiec, Pedersen, and Chen developed Naïve Bayes classifier which classified the sentences of the document to add in the summary [4]. They used sentence scoring features like Sentence length cutoff, fixed-phrase, paragraph feature, thematic words, and uppercase words to decide the probability of the sentence. In 1995, SUMMONS (Summarizing online news articles) a natural language based summarizer was developed by McKeown and Radev [5]. This was the first multi-document summarization system. In 1997, Inderjeet Mani and Eric Bloedorn developed a graph based method in which the text was represented in the form of graphs [6]. Spreading activation technique was used to find semantic nodes which are related. It was used to identify the similarities and differences between the documents. In 2001, Conroy and O'leary proposed a Hidden Markov Model (HMM) for text summarization [7]. Features like position, sentence terms and their respective frequencies were incorporated for HMM development. In 2004, Radev, Jing, Stys, and Tam developed a centroid based multi document text summarizer named MEAD [8]. Term Frequency- Inverse Document Frequency (TF-IDF) feature was used for centroid based summarization. MEAD was used to summarize clusters of news articles. In 2005, Evans and Klavans proposed a new method for summarizing the clusters of documents on the same events based on text similarity [9]. English and Arabic language documents were used. In 2015, S. A. Babar and P. D. Patil proposed an approach to improve the performance of text summarization using Singular value Decomposition and Fuzzy inference system [10]. In 2016, S.P Singh, A. Kumar, A. Mangal, S. Singhal proposed a method for bilingual text summarization using Restricted Boltzmann Machine (RBM) [11]. Eleven sentence scoring features were used and given to RBM for feature enhancement. In 2017, an approach for auto text summarization was developed by H. A. Chopade and M. Narvekar using Deep network and Fuzzy logic which provided significant increase in the accuracy of the summary [12].

This work is focused on extractive text summarization. Combining some of the significant works an approach is designed which uses RBM as an unsupervised deep learning algorithm and Fuzzy logic, along with some sentence features to improve the connectivity and the relevance in the sentences.

A. Organisation of the paper

In this paper, section I gives an introduction to the topic and its literature survey, section II describes the proposed method for extractive text summarization, section III produces experiments

978-1-5386-5257-2/18/$31.00 ©2018 IEEE

and results, and section IV states conclusions and the future scope.

II. Proposed Method

A. Problem Statement

Due to increasing text data on the web, the time required for the users to summarize and analyze the huge data is increased. The solution to reduce reading time of the user is producing a succinct document summary.

B. Method

Fig.1 shows the proposed method for Extractive text summarization using Deep Learning:

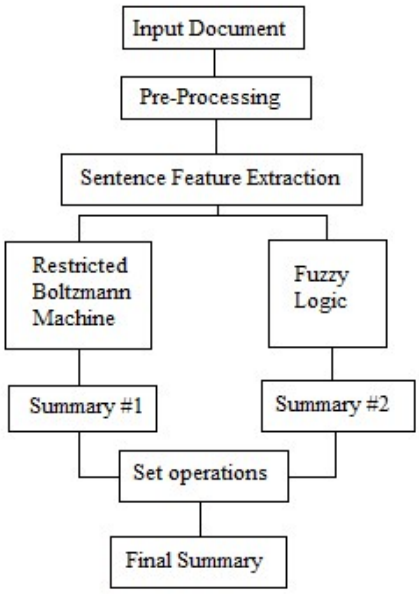


Fig. 1 Proposed Method

1) Input Document: The first step of the process for generating the summary is to input a text document with format as .txt. Text Documents used in this work are in English language. The text files are imported using tkinter python library.

2) Pre-Processing: The imported text is processed as shown in Fig. 2.

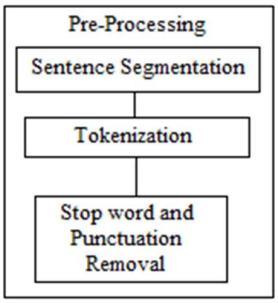


Fig. 2 Pre-Processing

In Sentence Segmentation, the whole text is broken down into sentences and is stored in an array with their respective sentence positions. In Tokenization, the sentences obtained are further broken down into words for some feature calculations. In Stop word and punctuation removal, the commonly occurring words such as the, an, a, but, and, or etc. are removed along with all the punctuation.

All the pre-processing steps are carried out using a library in python called as Natural Language Toolkit (NLTK) for Natural Language Processing.

3) Sentence Feature Extraction: After Pre-processing the text, sentence features are calculated to find the sentence score. Following sentence features contribute in deciding the sentence score:

- Sentence Position: On the basis of sentence location, its relevance is known. In [2] author states that the first and the last sentence of the document is always important and has maximum information. Position feature is calculated using (1).

$$\text{Sent. Pos.} = \begin{cases} 1, & \text{if first or last sentence} \\ \frac{N-P}{N}, & \text{if others} \end{cases} \quad (1)$$

where, N is the total sentences, P is the location of the sentence.

- Sentence Length: According to the author, in [2] very short sentences do not contain much information. To find the important sentence based on its length, the feature score is calculated using (2).

$$\text{Sent. Len}_{\,i} = \frac{\text{words in sentence}_i}{\text{words in largest sentence}} \quad (2)$$

- Numerical Token: Numerical token is the total numerical values in a sentence. It is calculated using (3).

$$\text{Numerical token Score}_i = \frac{\text{num\_numeric}_i}{\text{len}} \quad (3)$$

where, num_numeric_i is the numerical tokens in i^{th} sentence and len is the total words in i^{th} sentence.

- TF-ISF: Term frequency - Inverse document frequency (TF-IDF) is necessary for systems retrieving information. In this work, text summarization is carried out for a single document. So, Term frequency - Inverse Sentence Frequency (TF-ISF) is calculated using (4) [11].

$$\text{TFISF score} = \frac{(\log(\text{isf}) * (\text{tf}))}{\text{len}} \quad (4)$$

Where, isf is the total occurrences of the each term of i^{th} sentence in all other sentences, tf is term frequency of each term in i^{th} sentence and len is total words in i^{th} sentence.

- Cosine similarity between Sentence and Centroid: Centroid is the sentence with largest TF-ISF [11]. Cosine similarity with the centroid is calculated for each sentence as given in (5).

$$\text{Cos}_{\text{sim}_i} = \cos(\text{sentence}_i, \text{centroid}) = \frac{\text{sentence}_i \cdot \text{centroid}}{||\text{sentence}_i|| \, ||\text{centroid}||} \quad (5)$$

- Bi-Gram: Bi-grams are pair of two adjacent words formed for each sentence in the document. NLTK library is used for finding the total no. of bi-grams in a sentence. The feature score is normalized to restrict the feature value between 0 and 1.

- Tri-Gram: Tri-grams are a triple of three adjacent words formed each sentence in the document. NLTK library is used for finding the total no. of tri-grams in a sentence. The feature score is normalized to restrict the value between 0 and 1.

- Proper Noun: Proper noun is a noun which refers to a unique identity like name, place, etc. In this feature, the total no. of proper nouns in each sentence is calculated using (6) [10]. To find the proper nouns, the sentences are POS tagged using NLTK.

$$\text{Proper noun score}_i = \frac{\text{No.of proper nouns}_i}{\text{Sentence length}_i} \quad (6)$$

  where, No. of proper nouns$_i$ is the no. of proper nouns in ith sentence and sentence length$_i$ is the no. of total words in ith sentence.

- Thematic Words: Thematic words are keywords in each sentence which are domain specific. The no. of thematic words in a sentence is calculated using (7) [10].

$$\text{Thematic word}_i = \frac{\text{No.of thematic words}_i}{\text{Total no.of thematic words}} \quad (7)$$

  where, No. of thematic words$_i$ is the no. of thematic words in ith sentence.

After calculating all the sentence features, a sentence feature matrix is formed. In this work, each sentence has nine feature values. The number of feature values is variable based on the number of features used.

4) Restricted Boltzmann Machine (RBM): RBM is a neural network with random probability distributions. The network contains a visible layer of visible neurons (input nodes) and hidden layers of hidden neurons (hidden nodes). Every input node has a bi-directional connection with every hidden node. The bias node has a connection with every hidden node. The input nodes are not interconnected in the visible layer. Also the hidden nodes are not interconnected in the hidden layers. Because of these restricted connections, the name of the network is Restricted Boltzmann machine. Fig.3 shows the RBM Network architecture.

   After the formation of the sentence matrix, it is normalized by dividing each of its elements with the highest element and is given as an input to the RBM to enhance the values for obtaining the sentence score. In this work, RBM consists of three layers: One Visible layer, and two Hidden layers.

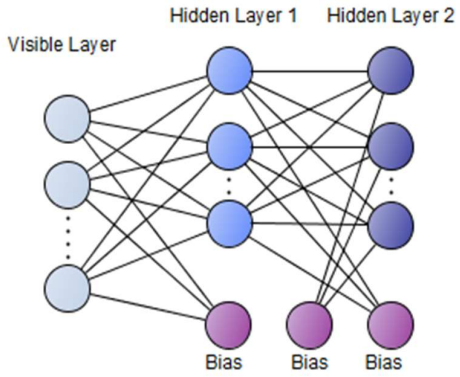


Fig. 3 RBM architecture

Visible layer contains nine nodes corresponding to nine features.

As shown in Fig.4, in the Forward pass, features of every sentence are multiplied by random weights when going through Hidden layer 1.

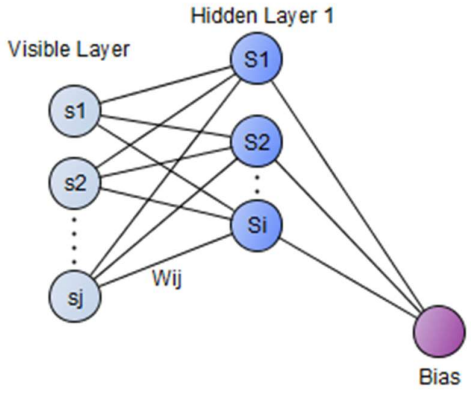


Fig. 4 RBM: Forward Pass- Visible layer to Hidden layer 1

A randomly generated bias is then added to the product. The probability of activation of hidden unit S_i given visible unit s_j is given in (8).

$$p(S_i|s_j) = \sigma(\sum_{j=1}^{n} s_j \times w_{ij} + b_i) \quad (8)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (9)$$

where, $\sigma(x)$ is the sigmoid function given in (9), w_{ij} are weights which are randomly generated and b_i is the bias.

The process is same for hidden layer 2 where in hidden layer 1 output is input for hidden layer 2 as shown in Fig. 5.

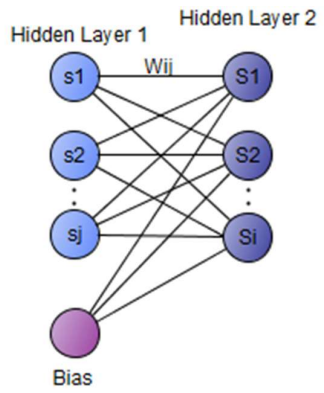


Fig. 5 RBM: Forward Pass- Hidden layer 1 to Hidden layer 2

In the backward pass, the output value is multiplied by the same weights and added to a bias as done in the forward pass and output is obtained at the visible layer as shown in Fig. 6.

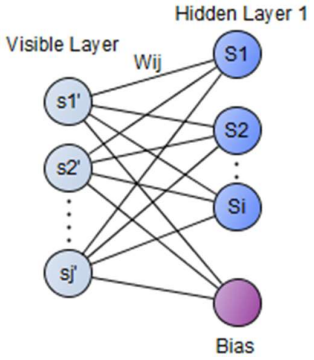


Fig. 6 RBM: Backward Pass- Reconstruction

The probability of activation of visible unit s_j given hidden unit S_i is given in (10).

$$p(s_j|S_i) = \sigma(\sum_{i=1}^{m} S_i \times w_{ij} + b_j) \quad (10)$$

where, $\sigma(x)$ is the sigmoid function given in (9), w_{ij} are weights and b_i is the bias.

Thus, while training, the values of hidden units are predicted, and after knowing the values of hidden units, the new values of input are predicted. This process is called Gibbs sampling. Difference between the input values s_j and the new input values s_j' is calculated which states the training loss. This process is carried out for certain epochs and the weights are learned by Contrastive Divergence given by (11).

$$w_{ij\,new} = w_{ij\,old} + (LR \times \Delta w) \quad (11)$$

$$\Delta w = (s_j \otimes p(S_i|s_j) - s_j' \otimes p(S_i'|s_j')) \quad (12)$$

where, LR is the learning rate, Δw is change in weights given in (12) as difference of outer products of probabilities with original input values s_j and new input values s_j'. An enhanced feature matrix is obtained after the epochs which has enhanced feature values for each sentence.

5) Summary #1: Fig. 7 shows the process of generating the first summary. The sum of all enhanced feature values for each sentence in the document is calculated and stored in a list.

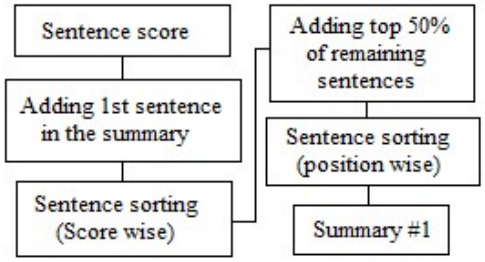


Fig. 7 Generating Summary #1

Thus, for each sentence one value is generated which is the score of that sentence. On the basis of scores, sentences are arranged in descending manner. Summary always includes the first sentence as it is the most important sentence and then top 50% of the remaining sentence based on their descending scores are added in the first summary and sorted according to their original position in the document.

6) Fuzzy Logic: The feature scores calculated earlier are converted into percentage. Triangular membership functions are used to fuzzify each score into three levels HIGH, MEDIUM and LOW as shown in Fig. 8. IF-THEN fuzzy rules are then applied for de-fuzzification to determine whether the sentence is Important, Average or Unimportant e.g. IF (Feature1 is HIGH, Feature2 is HIGH, Feature3 is MEDIUM, Feature4 is MEDIUM), THEN (Sentence is Important). Python library used for fuzzy logic system is skfuzz.

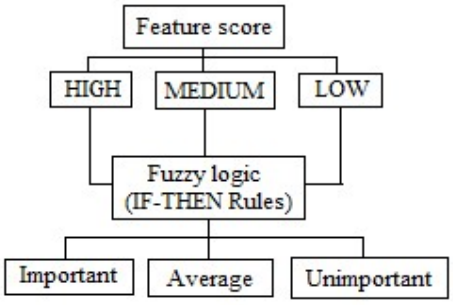


Fig. 8 Fuzzy logic system

7) Summary #2: After getting the computation results from the fuzzy logic system, the sentences which fit in "Important" category are added in the second summary according to their original position in the document.

8) Set operations for Final Summary: Procedure for generating the final summary is shown in Fig. 9. From the first and second summary, a common and uncommon set of sentences are found. Inclusion of common set is made in the Final summary.

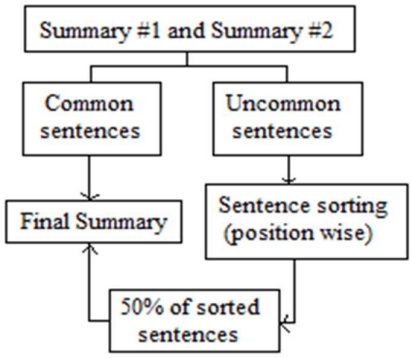


Fig. 9 Generating final summary

The uncommon sentences are sorted position wise and half part of uncommon set of sentences are added in final summary. After adding, the sentences are arranged according to their original position in the text document.

III. Experimentations and Results

Experimentations are performed using Intel Core i3 2.0GHz processor, 4GB RAM. The programming language used is Python 3.6 with Jupyter Notebook as a platform.

In the proposed method, nine feature values are calculated for a sentence to get better relevance. RBM is trained for fifteen epochs for generating the first summary of the document. Fuzzy logic IF-THEN rules are used for generating the second summary of the same document and then using set operations a final summary is obtained.

The proposed method for Extractive text summarization is implemented for single document along with the original method which uses RBM only. The generated summaries from both the methods are compared. Recall Oriented Understudy for Gisting Evaluation (ROUGE) is used for evaluating generated summaries. The parameters for performance evaluation are Precision, Recall and F measure. The values are found out for ten English documents from the News articles dataset from Kaggle using both the methods. Average values with both the methods are shown in TABLE 1.

TABLE 1. COMPARISON RESULTS OF TEXT SUMMARIZATION

	RBM method			Proposed method		
Doc	***Precision***	***Recall***	***F measure***	***Precision***	***Recall***	***F measure***
1	0.78	0.7	0.74	0.8	0.88	0.84
2	0.85	0.76	0.80	0.86	0.75	0.80
3	0.75	0.7	0.72	0.87	0.77	0.82
4	0.82	0.78	0.80	0.91	0.79	0.85
5	0.84	0.8	0.82	0.89	0.80	0.84
6	0.77	0.76	0.76	0.92	0.86	0.89
7	0.87	0.83	0.85	0.88	0.78	0.83
8	0.79	0.76	0.77	0.89	0.72	0.80
9	0.85	0.82	0.83	0.87	0.84	0.85
10	0.83	0.79	0.80	0.86	0.81	0.83
Avg	**0.82**	**0.77**	**0.79**	**0.88**	**0.80**	**0.84**

IV. Conclusions and Future Scope

In this work, RBM is used as an unsupervised learning algorithm along with fuzzy logic for improving the accuracy of the summary. It is observed that the proposed approach generates short and precise summaries without any irrelevant text. Using features like Sentence-Centroid similarity and thematic words has improved the connectivity of the sentences. Using the proposed method, on an average 88% precision, 80% recall and 84% F measure is obtained. The results produced using the proposed method give better evaluation parameters in comparison with prevailing RBM method.

The proposed method can be extended for multi document summarization. Documents in different languages can be summarized. Various other features can be used and the method can be combined with other methods for improving the nature of the summary. Also it can be used in Abstractive text summarization.

References

[1] H. P. Luhn, "The automatic creation of literature abstracts," IBM Journal of Research Development, vol. 2, no. 2, pp. 159-165, 1958.

[2] P. Baxendale, "Machine-made index for technical literature - an experiment." IBM Journal of Research Development, vol. 2, no. 4, pp. 354-361, 1958.

[3] H. P. Edmundson, "New methods in automatic extracting," Journal of the ACM, vol. 16, no. 2, pp. 264-285, 1969.

[4] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," In Proceedings SIGIR '95, pp. 68-73, New York, NY, USA, 1995

[5] K. R. McKeown, and D. R. Radev, "Generating summaries of multiple news articles," In Proceedings of SIGIR '95, pp. 74-82, Seattle, Washington, 1995.

[6] Inderjeet Mani and Eric Bloedorn, "Multi-document summarization by graph search and matching," AAAI/IAAI, vol. cmplg/ 9712004, pp. 622-628, 1997.

[7] J. M. Conroy, and D. P. O'leary, "Text summarization via hidden markov models," In Proceedings of SIGIR '01, pp. 406-407, New York, NY, USA, 2001.

[8] D. R. Radev, H. Jing, M. Stys, and D. Tam, "Centroid-based summarization of multiple documents," Information Processing and Management, vol. 40 , pp. 919-938, 2004.

[9] D. K. Evans, "Similarity-based multilingual multidocument summarization," Technical Report CUCS-014- 05, Columbia University, 2005.

[10] S. A. Babar, and P. D. Patil, "Improving performance of text summarization," Procedia Computer Science, vol. 46, pp. 354-363, 2015.

[11] S. P. Singh, A. Kumar, A. Mangal, and S Singhal, "Bilingual Automatic Text Summarization Using Unsupervised Deep Learning." International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)-2016, pp. 1195-1200, 2016.

[12] H. A. Chopade and M. Narvekar, "Hybrid auto text summarization using deep neural network and fuzzy logic system," 2017 International Conference on Inventive Computing and Informatics (ICICI), COIMBATORE, India, pp. 52-56, 2017.

Extractive Text Summarization for Sports Articles using Statistical Method

Sai Teja Polisetty, K Selvani Deepthi, Shaik Ameen, Ravivarma G, M Mounisha

Abstract: ***The past decade has endorsed a great rise in Artificial Intelligence. Text summarization which comes under AI has been an important research area that identifies the relevant sentences from a piece of text. By Text Summarization, we can get short and precise information by preserving the contents of the text. This paper presents an approach for generating a short and precise extractive summary for the given document of text. A statistical method for extractive text summarization of sports articles using extraction of various features is discussed in this paper. The features taken are TF-ISF, Sentence Length, Sentence Position, Sentence to Sentence cohesion, Proper noun, Pronoun. Each sentence is given a score known as the predictive score is calculated and the summary for the given document of text is given based on the predictive score or also known as the rank of the sentence. The accuracy is checked using the BBC Sports Article dataset and sports articles of various newspapers like the New York Times, CNN. The precision of 73% is acquired when compared with System Generated Summary (SGS) and manual summary, on an average.***

Keywords: Artificial Intelligence, Cosine similarity, Natural Language Processing, System Generated Summary (SGS), Term frequency inverse sentence frequency.

I. INTRODUCTION

Text mining can also be referred as Text analysis, which uses different Artificial Intelligence technologies. Text mining can be referred to as deriving high-quality information by drawing patterns and identifying the important keywords from the unstructured data. Text mining tasks include text classification which is the classification of the text for example genre, the sentimental analysis which tells about how the author wrote the sentences that is in which tone, document clustering.

Text mining tasks include text classification [2] which is the classification of the text for example genre, the sentimental analysis which tells about how the author wrote the sentences that is in which tone, document clustering refers to as clustering the documents using unsupervised learning, text summarization for obtaining a short and precise text. Each task is different from each other and has its own probe and methodologies. Natural Language Processing (NLP)is all about computer and human languages interactions. A finest procedure that helps the system in reading and understanding the given text. With the rapid increase in data the text summarization manually consumes a lot of time. That resulted in development of various summarization methods. Using them, the system should deliver a précis with the main essence of the document in reduced size. Text summarization is of two types: i) extractive ii) abstractive. Firstly, the extractive text summarization where important sentences and words from the given text document are identified and those are combined into the summary in a meaningful way. Finally, in abstractive text summarization, new sentences are fabricated, and the summary is given without the loss of information which is understandable. This can be done using different methods that include statistical measures, deep learning techniques with supervised or unsupervised learning and word vector embedding [10]. The literature survey is as follows:

Fabio Bif Goularte et.al (2018) [1] proposed a method using fuzzy rules to retrieve the most important sentences in a document. Fuzzy logic is applied because it can be used for sentences having ambiguity. The summaries are trained for Brazilian Portuguese texts and the summary generated is tested by the summary given by the domain experts. The author has cross-checked with other methods as well as using ROUGE measures. The f-measure obtained is 0.95 which is huge than obtained in any other method. This method works accurately with only unigrams and it doesn't work for semantic verification of the information.

Taeho Jo et.al (2017) [2] proposed a method that uses the feature vector of certain features and obtains the correlation between the vectors. In previous works only the feature value is considered but, in this work, the author considers both features and its value for better performance for this KNN (K Nearest Neighbour) is used to check the correlation between the vectors. The author has used a binary classification, KNN is used which tells whether a sentence is important or not. This is way different from text categorization which uses the previous knowledge for classifying.

Deepa Anand et.al (2019) [3] proposed a method for summarizing the Indian legal judgment documents for this the author has used a neural network approach that is semi-supervised. The author used two different LSTM architectures for both word embedding and sentence embedding. The author used Glove vectors and co-selection measures are used for analysis. The recall of 0.7 to 0.8 approximately is obtained which is moderate. This approach cannot be used for long sentences so sentence simplification methods should be used.

Revised Manuscript Received on February 01, 2020.

*Correspondence Author

***Sai Teja Polisetty,** pursuing B. Tech final year, Department of CSE, Anil Neerukonda Institute of Technology and Sciences, India,

Dr. K. Selvani Deepthi, Associate Professor, Department of CSE, Anil Neerukonda Institute of Technology and Sciences, India,

Shaik Ameen, pursuing B. Tech final year, Department of CSE, Anil Neerukonda Institute of Technology and Sciences, India,

Ravivarma G, pursuing B. Tech final year, Department of CSE, Anil Neerukonda Institute of Technology and Sciences, India,

M Mounisha, pursuing B. Tech final year, Department of CSE, Anil Neerukonda Institute of Technology and Sciences, India,

Krishnaveni et.al (2017) [4] proposed a heading-based text summarization since the context can be known from the heading. Each sentence is given a rank on considering how much relevant the sentences are with the heading and the top sentences are retrieved based on the compression ratio. Since the summary is generated based on the heading there is not irrelevancy in context. The summaries are compared with that of the summaries generated from the main summariser, Ms-word summarizer.

Jingqiang Chen et.al (2018) [5] proposed a multi-model neural network-based extractive summarizer. A multi-model architecture is used on is bidirectional RNN and the other is CNN. This method encodes the text and images using this architecture and the probability of the sentence is calculated using a logistic classifier that has text coverage and text redundancy as features. The dataset used the Daily mail corpus by gathering the images from the internet. Encoding of the images is done using bidirectional RNN.

Jorge V. Tohalino et.al (2018) [6] has proposed a multi-layered neural network model to generate the extractive summary based on the sentences having high importance in the context for multi-documents. A graph which consists of nodes and edges is built with sentences and words respectively. The datasets used are CST News for Portuguese multi- document summarization and DUC-2002, DUC-2004 for English multi-document summarization. The word embedding feature is lacking in this work for generating a better summary.

J.N.Madhuri et.al (2019) [7] has proposed a method to generate an extractive summary using sentence ranking technique using the term frequency after the removal of stop words. It works for any kind of text but cannot semantically distinguish sentences. The evaluation is done using the MSWord summarizer and human summarized summary. The summaries are converted to mp3 format so that it is easy to evaluate or know the summary.

Nikhil S. Shirwandkar et.al (2018) [8] has proposed an approach using both Restricted Boltzmann machine and Fuzzy logic to identify the important sentences. Two summaries for the same input document are obtained and then the final summary is obtained by considering both the summarization get meaningful and no loss of information. The author did summarization for only a single document. The f-measure obtained is 84%.

Mahmood Yousefi-Azar et.al (2016) [9] has proposed a text summarizer for a query-oriented system using an unsupervised deep learning network which is a deep auto-encoder to calculate feature space from term frequency. This is a stochastic machine. The experiments are done on SKE and BC3 email datasets. A semi-supervised learning approach works better because it can be used for unlabelled data.

Aditya Jain et.al (2017) [10] has proposed a neural network-based text summarization, the input of the nodes is the values of different features [11] and the dataset used is DUC-2002. The summary is compared with four online summarizers. The accuracy of summarization can be exalted using a large training dataset for a neural network.

II. PROPOSED METHOD

The author used statistical methods for generating precis. There are seven features which are chosen to precis the information. All the seven features that are calculated for each sentence and the sentences are given a ranking from which the threshold known as the compression ratio [6] is chosen which tells how many sentences should be present in the summary. Suppose ‘k’ sentences should be included in the summary then ‘k’ sentences with the highest rank are added to the summary. The sentences with a certain compression ratio should be in ascending order so that the context of the article will not change.

A. Architecture

1) Input Document: To generate a summary some input should be given in the file format “.txt”. The text documents of sports articles are considered for summarization and are valid since the author has proposed a method for summarizing sports articles.
2) Pre-processing: Stop word removal and stemming are done to the text considered as input after lower casing. Fig2 shows various techniques of pre-processing the author used. In the lower casing, the whole document is converted into lowercase alphabetical letters. This is done to remove the ambiguity of the words with different casing. The stop words in the document are removed from the document because these are the most frequent words like articles, prepositions, conjunctions which don’t add any value in defining the importance of a sentence. The stop words are removed using nltk (natural language tool kit) library. Stemming [4] is referred as getting the words reduced to their root form. It removes the different forms of the same word present in different sentences. For example, eating, ate and eat are identified as eat.

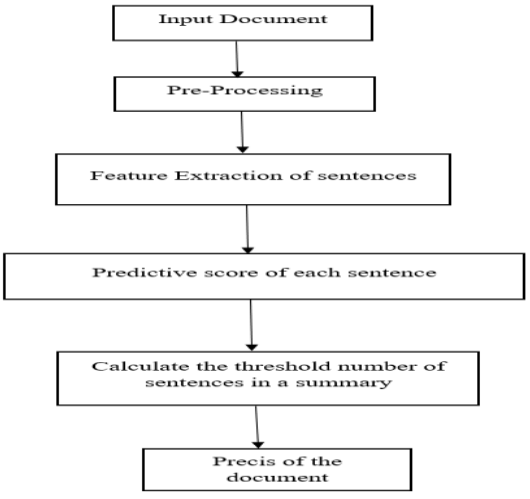


Fig.1 Architecture of the proposed method

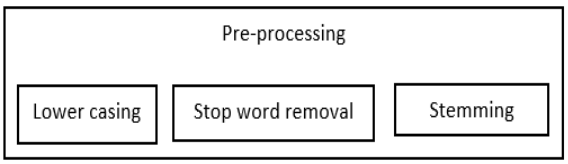


Fig.2 Steps included in Pre-processing

3) Feature Extraction – After pre-processing the text, sentence features are extracted to calculate the predictive scores.

- TF-ISF: Term frequency and inverse document frequency (TF-IDF) [4] are used for information retrieval systems. In this paper, the summarization is done for a single document of sports article and Term frequency inverse sentence frequency (TF-ISF) is calculated as follows.

$$TF - ISF(i,j) = TF(j) * \log\left(\frac{N}{ISF(j)}\right) \quad (1)$$

$$TF - ISF(i) = \sum_{j=0}^{n} \frac{(\text{TF-ISF(i,j)})}{n} \quad (2)$$

Where TF_j indicates the term frequency of a word in a sentence, ISF_j is the inverse sentence frequency of a j^{th} word in a sentence, TF-ISF_i(2) is the Term frequency and inverse sentence frequency of i^{th} sentence in a document, n cites to the number of words in a sentence, N cites to the number of sentences in a document.

- Sentence length: This feature provides the less weightage to short sentences as short sentences are relatively less important when compared to long sentences [2]. It is measured by calculating the ratio of no of words in the sentence to the no of words in the longest sentence of the document. This feature's value is calculated using eq. (3).

$$sent.length = \frac{no\ of\ words\ in\ sentence}{no\ of\ words\ in\ longest\ sentence} \quad (3)$$

- Sentence position: In a document the numerical value of the sentence's position is gauged(4). The position is evaluated as the normalized percentile score in the range of 0 to 1.

$$sent.position = \begin{cases} 0, & if\ first\ sentence \\ 1, & if\ last\ sentence \\ \frac{p}{n}, & if\ 0 < p < n \end{cases} \quad (4)$$

p is the position of the current sentence and n is the total number of sentences.

- Cosine similarity: Analogy between sentences is calculated using this property. This briefs how much similar this sentence with other sentences in a document is [7]. A vector for each sentence is assessed and (5) is practiced obtaining the score of the feature.

$$cosine\ similarity = \frac{A.B}{|A||B|} \quad (5)$$

A is the first vector and B is the second vector with which it is compared

- Existence of proper noun: Proper names broaching to places and people might be useful to decide the relevance of a sentence. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.
- Existence of pronoun: Pronouns cites to the persons which are described in the previous sentences. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.
- Sentence containing scores: Since in sports the score of the team or the individual participant is important the score is taken into consideration. This binary feature, with value '1' if a sentence contains these proper names and 0 otherwise.

4) Predictive score of each sentence: The predictive score is calculated by considering the mean of cosine similarity feature and TF-ISF feature and taking the sentence length and sentence position and addition of the remaining features and the score for each sentence is given known as the predictive score.

5) Calculate the threshold number of sentences in a summary: The threshold value known as the compression ratio tells us how much percent of the total document is presented as a summary.

6) Precis of the document: This is the output of the text document which is the summary. Suppose 'k' be the threshold and the 'k' sentences with the highest predictive score should be in ascending order so that the context of the article will not change.

B. Algorithm

Step1: Take text file as an input
Step 2: Split the entire document into sentences.
Step 3: Convert all the sentences into lowercase.
Step 4: Split the sentences into words and remove the stop words from the obtained words.
Step 5: Stemming for all the words after stop word removal is done using nltk toolkit.
Step 6: For all the sentences
Step 7: Calculate the TF-ISF for each word after stemming and take the mean of all the values of a sentence and one value is obtained which is the TF-ISF for one sentence.
Step 8: Repeat step 7.
Step 9: Calculate the sentence length for each sentence.
Step 10: Check for proper noun in a sentence if present update to 1 else 0.
Step 11: Check for pronoun in a sentence if present update to 1 else 0.
Step 12: Calculate the sentence position using (4)
Step 13: Check if a value is present in sentence if present update to 1 else 0.
Step 14: Convert the sentences into vectors.
Step 15: For all the sentence vector
Step 16: For all the next sentence vector
Step 17: Calculate the cosine similarity. Repeat 16
Step 18: Repeat 15
Step 19: Calculate the mean of all the cosine similarity with respect to other sentences and obtain a value for each sentence.
Step 20: Calculate the mean of TF-ISF and cosine similarity values, obtain predictive score.
Step 21: Sort the predictive scores and list the top sentences based on the compression ratio.
Step 22: Add the sentences to the final summary.
Step 23: Print the final summary.

C. Pseudo Code

- Pre-processing

```
f=open("input.txt",'r')
m= [] ([] : Empty list)
for line in f:
```

```
        m.append(line.split('.')
    sentence= []
    for i in range(len(m)):
        sentence.append(m[i].lower())
    words_after_stemming=[]
    for i in range(len(sentence)):
        if(sentence[i] not a stop word):
            words_after_stemming.append(PorterSt
        emmer().stem(sentence[i]))
```

- TF-ISF

```
tfisf=[]
for i in range(len(words_after_stemming):
for j in range(len(words_after_stemming[i]):
tf=words_after_stemming[i].count(words_after_ste
mming[i][j])
        isf=math.log(N/c); tfisf[i]=tf*isf
```

- Sentence length

```
maximum_length=max(words_after_stemming.cou
nt())
sentence_length=[]
for i in range(len(words_after_stemming):
sentence_length.append(words_after_stemming.co
unt()/maximum_length)
```

- Proper Noun

```
propernoun=[]
for i in range(len(words_after_stemming)):
        if(pos_tag(words_after_stemming[i].split())[
        0][1]=='NN')
            propernoun.append(1)
        else
            propernoun.append(0)
```

- Pronoun

```
pronoun=[]
for i in range(len(words_after_stemming)):
        if(pos_tag(words_after_stemming[i].
            split())[0][1]=='PRP')
                    pronoun.append(1)
        else
                    pronoun.append(0)
```

- Sentence position

```
sentenceposition=[]
for i in range(1,len(words_after_stemming)+1):
        sentenceposition[i-
        1]=(i/len(words_after_stemming))
```

- Sentence containing scores

```
score=[]
for i in range(len(words_after_stemming)):
        if(sords_after_stemming[i] in
        '0123456789'):
                score.append(1)
        else
                score.append(0)
```

- Cosine similarity

```
cosine_similarity=[]
generate vectors for words
apply (5)
```

- Mean of TF-ISF and cosine similarity

```
score=[]
for i in range(len(cosine_similarity)):
        score=mean(cosine_similarity[i],tfisf[i])
```

- Sort predictive score

```
sort.score()
summary=[]
summary.append(score(i,compression_ratio))
```

- Print summary

```
print(summary)
```

III. RESULTS

The author considered BBC sports article data set since it is the worldwide famous for its news. It consists of 737 articles for five different types of sports namely athletics, cricket, football, rugby, tennis and 101, 124, 265, 147, 100 articles respectively. The model is tested for these articles and performed with a good amount of accuracy by generating the relevant context leaving the unnecessary sentences. The author also considered different sports articles from New York Times, CNN during the testing of model for accuracy.

A. Sample Input

```
England coach Andy Robinson faces the first major test of his tenure as
he tries to get back to winning ways after the Six Nations defeat by
Wales. Robinson is likely to make changes in the back row and centre
after the 11-9 loss as he contemplates Sunday's set-to with France at
Twickenham. Lewis Moody and Martin Corry could both return after missing
the game with hamstring and shoulder problems. And the midfield pairing
of Mathew Tait and Jamie Noon is also under threat. Olly Barkley
immediately allowed England to generate better field position with his
kicking game after replacing debutant Tait just before the hour. The
Bath fly-half-cum-centre is likely to start against France, with either
Tait or Noon dropping out. Tait, given little opportunity to shine in
attack, received praise from Robinson afterwards, even if the coach
admitted Cardiff was an "unforgiving place" for the teenage prodigy.
Robinson now has a tricky decision over whether to withdraw from the
firing line, after just one outing, a player he regards as central to
England's future. Tait himself, at least outwardly, appeared unaffected
by the punishing treatment dished out to him by Gavin Henson in
particular. "I want more of that definitely," he said. "Hopefully I can
train hard this week and get selected for next week but we'll have to
look at the video and wait and see. "We were playing on our own 22 for a
lot of the first half so it was quite difficult. I thought we defended
reasonably well but we've just got to pick it up for France." His
Newcastle team-mate Noon hardly covered himself in glory in his first
major Test. He missed a tackle on Michael Owen in the build-up to Wales'
try, conceded a penalty at the breakdown, was turned over in another
tackle and fumbled Gavin Henson's cross-kick into touch, all inside the
first quarter. His contribution improved in the second half, but England
clearly need more of a playmaker in the inside centre role. Up front,
the line-out remains fallible, despite a superb performance from Chris
Jones, whose athleticism came to the fore after stepping into the side
for Moody. It is more likely the Leicester flanker will return on the
open side for the more physical challenge posed by the French forwards,
with Andy Hazell likely to make way. Lock Ben Kay also justified his
recall with an impressive all-round display on his return to the side,
but elsewhere England positives were thin on the ground.
```

Fig.3 Sample input from dataset

B. Sample Output

```
[0, 1, 4, 6, 7, 10, 14, 16, 17, 18]
England coach Andy Robinson faces the first major test of his tenure as he tries to get back
to winning ways after the Six Nations defeat by Wales. Robinson is likely to make changes
in the back row and centre after the 11-9 loss as he contemplates Sunday's set-to with Fra
nce at Twickenham. Olly Barkley immediately allowed England to generate better field po
sition with his kicking game after replacing debutant Tait just before the hour. Tait, given
little opportunity to shine in attack, received praise from Robinson afterwards, even if the c
oach admitted Cardiff was an "unforgiving place" for the teenage prodigy. Robinson now h
as a tricky decision over whether to withdraw from the firing line, after just one outing, a p
layer he regards as central to England's future. "Hopefully I can train hard this week and g
et selected for next week but we'll have to look at the video and wait and see. He missed a
tackle on Michael Owen in the build-up to Wales' try, conceded a penalty at the breakdown
, was turned over in another tackle and fumbled Gavin Henson's cross-kick into touch, all i
nside the first quarter. Up front, the line-out remains fallible, despite a superb performanc
e from Chris Jones, whose athleticism came to the fore after stepping into the side for Moo
dy. It is more likely the Leicester flanker will return on the open side for the more physica
l challenge posed by the French forwards, with Andy Hazell likely to make way. Lock Ben
Kay also justified his recall with an impressive all-round display on his return to the side, b
ut elsewhere England positives were thin on the ground.
```

Fig.4 Sample Output for compression ratio (α=50%)

IV. DISCUSSION

The evaluation measures used are for context evaluation which comes under co-selection. In co-selection there are three types of measures they are precision, recall, f-measure.

Precision (P) can be defined as the fraction of the number of sentences common in manual and automated summary to the number of sentences in the automated summary. Precision (P) can be procured using equation (6).

$$P = \frac{S(manual) \cap S(auto)}{S(auto)} \quad (6)$$

Recall (R) can be defined as the ratio of the number of sentences common in manual and automated summary to the manual summary. The recall can be calculated using equation (7).

$$R = \frac{S(manual) \cap S(auto)}{S(manual)} \quad (7)$$

F-measure (F) can be defined as the harmonic mean of precision and recall. F-measure (F) can be procured using equation (8).

$$F = \frac{2*P*R}{P+R} \quad (8)$$

The System generated summary (SGS) is tested for accuracy with a manual summary and an online summarizer tool. The manual summary is given by one of the literates who have secured a degree in Literature. The online tool used for generating the summary is Text Compactor. The author has considered the compression ratio half of the original document.

Firstly, a manual summary is compared with SGS and a manual summary is compared with online summarizer tool for precision, recall, and f-measure. The documents tested are the sports articles from various newspapers including the New York Times, CNN. For analysing the accuracy, the author also checked it with BBC Sports Articles Dataset.

The Table-1 shows the number of sentences in each document taken from the news articles. The data of 10 documents is shown.

Title	Number of sentences in document	Number of sentences in SGS (α=50%)
Doc-1	11	5
Doc-2	16	7
Doc-3	15	7
Doc-4	8	4
Doc-5	36	19
Doc-6	15	7
Doc-7	39	20
Doc-8	27	14
Doc-9	19	10
Doc-10	32	16

The evaluation measures for some of the documents are given below.

Table.1 Shows number of sentences in document as well as SGS.

Title	Manual w.r.t SGS		
	P	R	F
Doc-1	0.8	0.8	0.8
Doc-2	0.7142	0.625	0.6662
Doc-3	0.7142	0.8333	0.7689
Doc-4	0.75	0.75	0.75
Doc-5	0.6842	0.7647	0.7219
Doc-6	0.7142	0.7142	0.7142
Doc-7	0.7	0.7368	0.7245
Doc-8	0.7142	0.7692	0.7401
Doc-9	0.8	0.889	0.8336
Doc-10	0.75	0.8	0.7741

Table.2 Comparison of performance between manual w.r.t SGS

Title	Manual w.r.t online summarizer tool		
	P	R	F
Doc-1	0.667	0.8	0.7274
Doc-2	0.6667	0.5	0.5714
Doc-3	0.5	0.667	0.5714
Doc-4	0.667	0.5	0.5714
Doc-5	0.7334	0.6470	0.6874
Doc-6	0.5714	0.5714	0.5714
Doc-7	0.834	0.834	0.834
Doc-8	0.5834	0.5384	0.56
Doc-9	0.667	0.667	0.667
Doc-10	0.5625	0.6	0.58

Table.3 Comparison of performance between manual w.r.t online summarizer tool.

	Average		
	P	R	F
Manual w.r.t SGS	0.7341	0.7681	0.7493
Manual w.r.t online summarizer tool	0.6452	0.6328	0.6341

Table. 4 The average of the Precision(P), recall(R), f-measure(F) for all the 10 documents

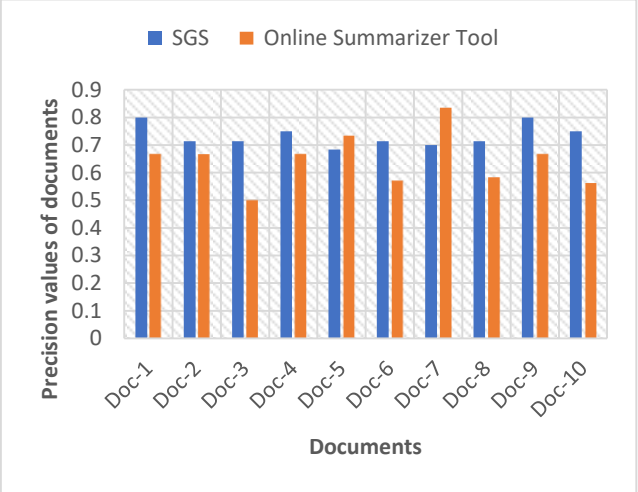


Fig.5 Comparison between SGS and Online summarizer tool

V. CONCLUSION AND FUTURE SCOPE

The extractive text summarization has many tasks involved in generating the précis. The author proposed a statistical approach for generating an extractive précis of a sports articles that uses sentence ranking based on the selected features for the sentences. The most critical part is identifying the important sentences without loss of meaning from the given document. The manual summary is generated for all the documents and a summary from online summarizer tool and SGS is compared with them.

Summing up using the proposed method on an average, 73% precision,

76% recall, and 74% f-measure is obtained which when compared with a manual generated summary and an online summarizer tool.

In nearly future, the proposed model can also be developed to a neural network model that takes the values of different features as input for generating an accurate precis by considering the contextual meaning. To increase the accuracy of the precis, various features can also be added.

REFERENCES

1. Goularte, Fábio & Nassar, Silvia & Fileto, Renato & Saggion, Horacio. (2018). A Text Summarization Method based on Fuzzy Rules and applicable to Automated Assessment. Expert Systems with Applications. 115. 10.1016/j.eswa.2018.07.047.
2. Jo, Duke Taeho. (2017). K nearest neighbor for text summarization using feature similarity. 1-5. 10.1109/ICCCCEE.2017.7866705.
3. Anand, Deepa & Wagh, Rupali. (2019). Effective Deep Learning Approaches for Summarization of Legal Texts. Journal of King Saud University - Computer and Information Sciences. 10.1016/j.jksuci.2019.11.015.
4. P. Krishnaveni and S. R. Balasundaram, "Automatic text summarization by local scoring and ranking for improving coherence," *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, 2017, pp. 59-64. doi: 10.1109/ICCMC.2017.8282539
5. J. Chen and H. Zhuge, "Extractive Text-Image Summarization Using Multi-Modal RNN," *2018 14th International Conference on Semantics, Knowledge and Grids (SKG)*, Guangzhou, China, 2018, pp. 245-248. doi: 10.1109/SKG.2018.00033
6. Valverde Tohalino, Jorge & Amancio, Diego. (2017). Extractive Multi-document Summarization Using Multilayer Networks. Physica A: Statistical Mechanics and its Applications. 503. 10.1016/j.physa.2018.03.013.
7. J. N. Madhuri and R. Ganesh Kumar, "Extractive Text Summarization Using Sentence Ranking," *2019 International Conference on Data Science and Communication (IconDSC)*, Bangalore, India, 2019, pp.1-3.doi: 10.1109/IconDSC.2019.8817040
8. N. S. Shirwandkar and S. Kulkarni, "Extractive Text Summarization Using Deep Learning," *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 2018, pp. 1-5. doi: 10.1109/ICCUBEA.2018.8697465
9. Azar, Mahmood & Hamey, Len. (2016). Text Summarization Using Unsupervised Deep Learning. Expert Systems with Applications. 68. 10.1016/j.eswa.2016.10.017.
10. Jain, Aditya & Bhatia, Divij & Thakur, Manish. (2017). Extractive Text Summarization Using Word Vector Embedding. 51-55. 10.1109/MLDS.2017.12.
11. K.Selvani Deepthi, Dr.Radhika Y(2015),"Extractive Text Summarization using Modified Weighing and Sentence Symmetric Feature Methods", in an International Journal of Modern Education and Computer Science, ISSN: 2075-0161 Volume 7 No-10 pp: 33-39, October 2015.

AUTHORS PROFILE

Sai Teja Polisetty is currently pursuing final year in Computer Science and Engineering, Anil Neerukonda Institute of Technology and Sciences. His area of Interest is Machine Learning, Natural Language Processing, Big Data Analytics.

K. Selvani Deepthi is doctorate in computer science and engineering. She is working as Associate Professor, Department of CSE, Anil Neerukonda Institute of Technology. Her area of Interest is Machine Learning, Natural Language Processing, Text mining.

Shaik Ameen is currently pursuing final year in Computer Science and Engineering, Anil Neerukonda Institute of Technology and Sciences. His area of Interest is Machine Learning, Business Intelligence.

Ravivarma G is currently pursuing final year in Computer Science and Engineering, Anil Neerukonda Institute of Technology and Sciences. His area of Interest is Deep Learning, Natural Language Processing.

Mounisha M is currently pursuing final year in Computer Science and Engineering, Anil Neerukonda Institute of Technology and Sciences. Her area of Interest is Natural Language Processing.