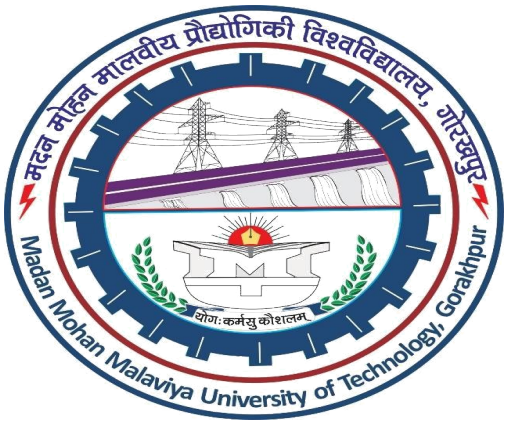
# TEXT SUMMARIZER USING MACHINE LEARNING

In Partial Fulfillment of the Requirements
For the Degree of Bachelor of Technology


**PROJECT TEAM MEMBERS:**
AMIT KUSHWAHA (2015021012)
DEEPAK KUMAR (2015021029)
NITIN SINGH (2015021067)

**UNDER GUIDANCE OF:**
SMT. MEENU (ASSISTANT PROFESSOR)



Department of Computer Science and Engineering

**MADAN MOHAN MALAVIYA UNIVERSITY OF TECHNOLOGY, GORAKHPUR (U.P.)-INDIA**



SESSION: 2018-2019

# TABLE OF CONTENTS:



# INTRODUCTION

# 1.1 OVERVIEW OF PROBLEM AREA:

Today we know that machines have become smarter than us and can help us with every aspect of life, the technologies have reached to an extent where they can do all the tasks of human beings like household tasks, controlling home devices, making appointments etc. The field which makes these things happen is Machine Learning. Machine Learning trains the machines with some data which makes it capable of acting when tested by the similar type of data. The machines have become capable of understanding human languages using Natural Language Processing. Today researches are being done in the field of text analytics.

As the project title suggests, Text Summarizer is an application which helps in summarizing the text. We can upload our data and this application gives us the summary of that data in as many numbers of lines as we want. The product is mainly a text summarizing using Deep Learning concepts. The main purpose is to provide reliable summaries of web pages or uploads files depend on the user’s choice. The unnecessary sentences will be discarded to obtain the most important sentences.

Text summarization methods are greatly needed to address the ever-growing amount of text data available online to both better help discover relevant information and to consume relevant information faster. There is a great need to reduce much of this text data to shorter, focused summaries that capture the salient details, both so we can navigate it more effectively as well as check whether the larger documents contain the information that we are looking for.

## Why Text Summarization?

In the modern Internet age, textual data is ever increasing; we need some way to condense this data while preserving the information and meaning. Text summarization is a fundamental problem that we need to solve. I would help in easy and fast retrieval of information, and use the retrieve information for our required purpose.

## Type of Summarization:

Extractive summarization:

Copying parts/sentences of the source text and then combines those part/sentences together to render a summary. Importance of sentence is based on linguistic and statistical features

Abstractive summarization:

These methods try to first understand the text and then rephrase it in a shorter manner, using possibly different words. For perfect abstractive summary, the models has to first truly understand the document and then try to express that understanding in short possibly using new words and phrases, and are much harder than extractive. It have complex capabilities like generalization, paraphrasing and incorporating real-world knowledge.

Majority of the work has traditionally focused on Extractive approaches due to the easy of defining hard-coded rules to select important sentences than generate new ones. I But they often don’t summarize long and complex texts well as they are very restrictive. The traditional rule-based AI does poorly on Abstractive Text Summarization. It is inspired by the performance of Neural Attention Model in the closely related task of Machine Translation Rush et al. 2015 and Chopra et al. 2016 applied this Neural Attention Model to Abstractive Text Summarization and found that it already performed very well and beat the previous non-Deep Learning-based approaches.

## 1.2 PROBLEM SPECIFICATION

**“Creating short, accurate, and fluent summaries from larger text documents using Machine Learning”.**

The trainable Text Summarizer is expected to learn the patterns which lead to the summaries, by identifying relevant feature values which are most correlated with the classes “correct” or “incorrect”. When a new document is given to the system, the “learned” patterns are used to classify each sentence of that document into either a “correct” or “incorrect” sentence, producing an extractive summary. A crucial issue in this framework is how to obtain the relevant set of features; the next section treats this point in more detail.

Researchers and students constantly face the scenario where it’s impossible to read most if not all of the newly published papers to be informed of latest progress and when the work on a research project, the time spent on reading literature review seems endless. The goal of this project is to design a Summarizer that generates a summary that helps the text to read in short time and focus on key features. The most important task is to generate an efficient scoring algorithm that would produce the best results for a wide range of text types. The only means to arrive at it was to manually summarize and then evaluate sentences for common traits, which would take a lot of time, but using Machine learning we can develop an algorithm that can process text and generate summary from it so that we can use it according to our need.

Textual information in the form of digital documents quickly accumulates to huge amounts of data. Most of this large volume of documents is unstructured: it is unrestricted and has not been organized into traditional databases. Processing documents is

therefore a perfunctory task, mostly due to the lack of standards. We cannot possibly create summaries of all of the text manually; there is a great need for automatic methods and Text Summarizer helps us to achieve that summary as result of its output to a given input as text to be processed.

# 2. TECHNOLOGY USED

## 2.1 Machine Learning

Machine Learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine Learning focuses on the development of computer programs that can access data and use it learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

### Some machine learning methods

Machine learning algorithms are often categorized as supervised or unsupervised.

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- **Unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer

a function to describe a hidden structure from unlabelled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabelled data

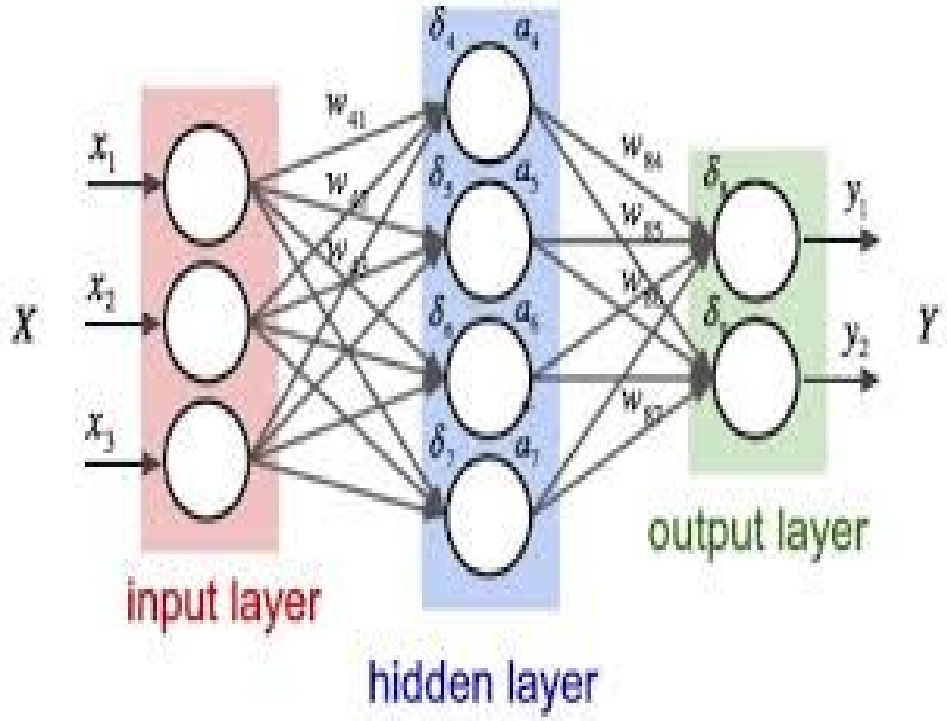
- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with Artificial intelligence and cognitive technologies can make it even more effective in processing large volumes of information.

## 2.2 Recurrent Neural Network (RNN)

### 2.2.1 Neural Network:

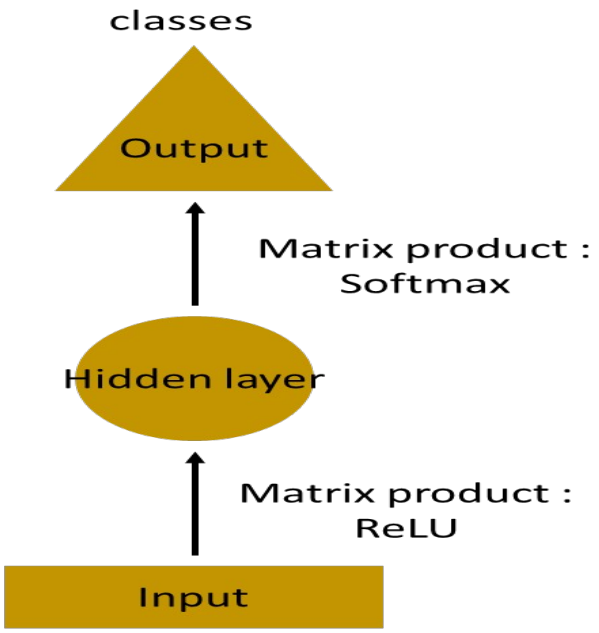
A Neural Network is an information processing paradigm that is inspired by the way biological nervous system, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. The neural network itself isn't and algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs.



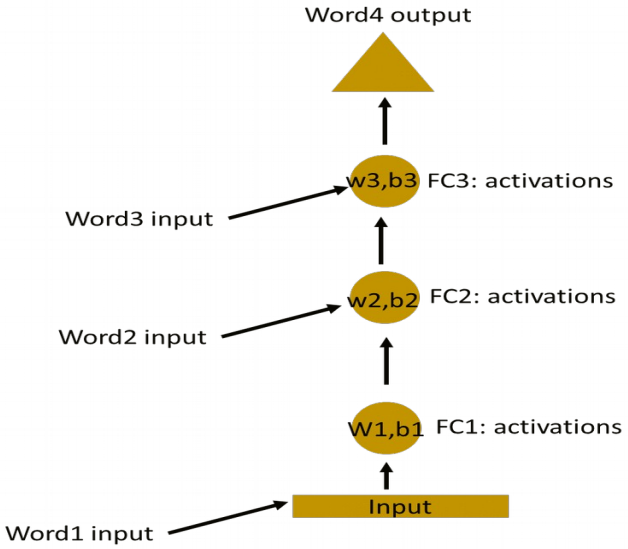

## 2.2.2 Recurrent Neural Network

A Recurrent Neural Network is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behaviour for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

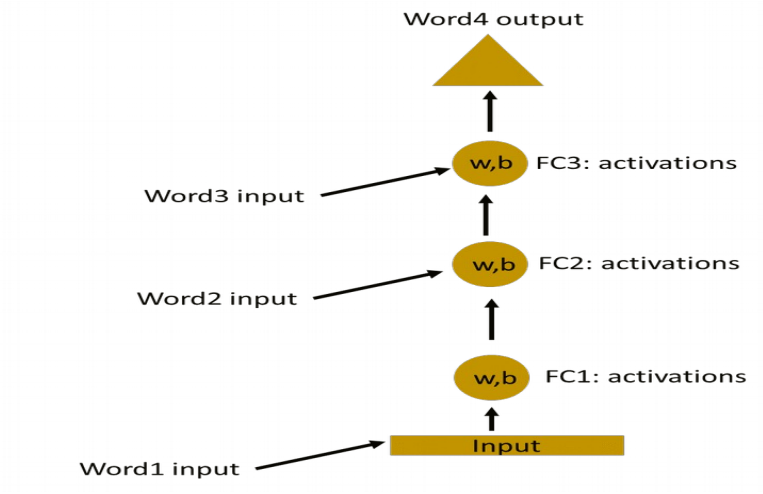
For example: Let’s say task is to predict the next word in a sentence. Using Neural Network in simplest form, we have an input layer, a hidden layer and an output layer. The input layer receives the input, the hidden layer activations are applied and then we finally receive the output.



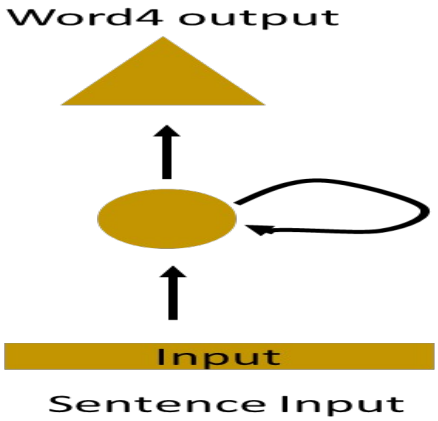

Let’s have a deeper network, where multiple hidden layers are present. So here, the input layer receives the input, the first hidden layer activations are applied and then these activations are sent to next hidden layer, and successive activations through the layers to produce the output. Each hidden layer is characterized by its own weights and biases. Since each hidden layer has its own weights and activations, they behave independently. Now the objective is to identify the relationship between successive inputs.




Here, the weights and bias of these hidden layers are different. And hence each of these layers behaves independently and cannot be combined together. To combine these hidden layers together, we shall have the same weights and bias for these hidden layers.




We can now combines these layers together, that the weights and bias of all hidden layers is the same. All these hidden layers can be rolled in together in a single recurrent layer.




So it's like supplying the input to the hidden layer. At all the time steps weights of the recurrent neuron would be the same since it's a single neuron now. So a recurrent neuron stores the state of a previous input and combines with the current input thereby preserving some relationship of the current input with the previous input.

## 2.3 Sequence to Sequence model using RNN:

In Sequence to Sequence model with attention two recurrent neural networks work together to transform one sequence to another. An encoder network condenses an input sequence into a vector, and a decoder network unfolds that vector into a new sequence. Unlike sequence prediction with a single RNN, where every input corresponds to an output, the seq2seq model frees us

from sequence length and order, which makes it ideal for translation between two languages.

With a seq2seq model the encoder creates a single vector which, in the ideal case, encodes the “meaning” of the input sequence into a single vector — a single point in some N dimensional space of sentences.

The Encoder

The Encoder of a seq2seq network is a RNN that outputs some value for every word from the input sentence. For every input word the encoder outputs a vector and a hidden state, and uses the hidden state for the next input word.

The Decoder

The decoder is another RNN that takes the encoder output vector(s) and outputs a sequence of words to create the translation.

Simple Decoder: In the simplest seq2seq decoder we use only last output of the encoder. This last output is sometimes called the *context vector* as it encodes context from the entire sequence. This context vector is used as the initial hidden state of the decoder. At every step of decoding; the decoder is given an input token and hidden state. The initial input token is the start-of-string `<SOS>` token, and the first hidden state is the context vector (the encoder’s last hidden state).

Attention Decoder
If only the context vector is passed between the encoder and decoder, that single vector carries the burden of encoding the entire sentence. Attention allows the decoder network to “focus” on a different part of the encoder’s outputs for every step of the decoder’s own outputs. First we calculate a set of attention weights. These will be multiplied by the encoder output vectors to create a weighted combination. The result should contain information about that specific part of the input sequence, and thus help the decoder choose the right output words.

Calculating the attention weights is done with another feed-forward layer , using the decoder’s input and hidden state as inputs. Because there are sentences of all sizes in the training data, to actually create and train this layer we have to choose a maximum sentence length (input length, for encoder outputs) that it can apply to. Sentences of the maximum length will use all the attention weights, while shorter sentences will only use the first few.

The encoder-decoder model is composed of encoder and decoder like its name. The encoder converts an input document to a latent representation (vector), and the decoder generates a summary by using it

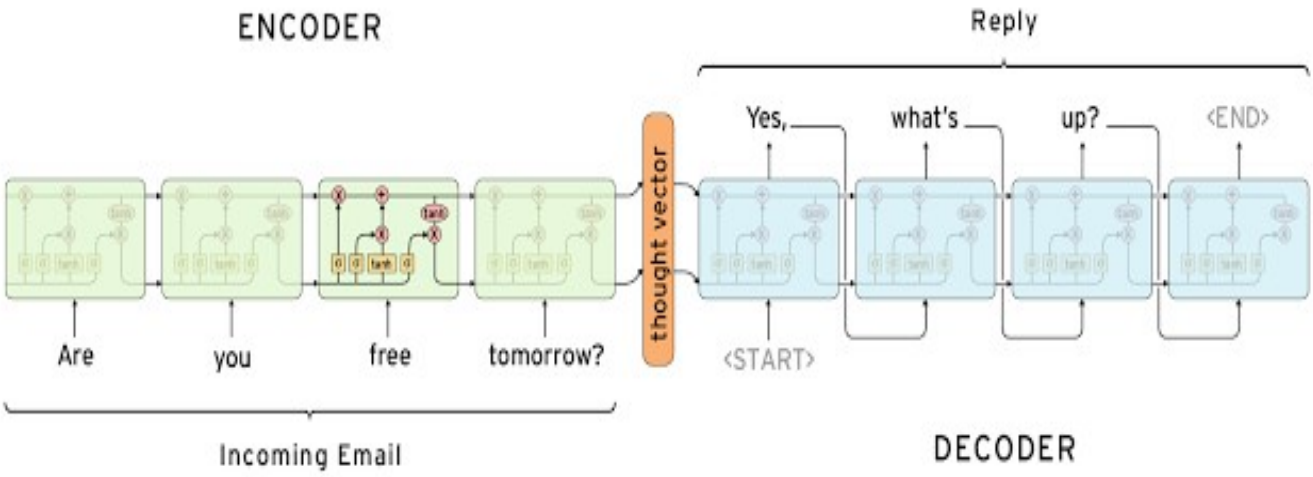



Diagram by Chris Olah

# 3.TOOLS

## 3.1 DATA MANIPULATION

### 3.1.1 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of

becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Pandas is well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet, Ordered and unordered (not necessarily fixed-frequency) time series data. Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels. Any other form of observational / statistical data sets. The data actually need not be labelled at all to be placed into a pandas data structure
The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating-point data
- Size mutability: columns can be inserted and deleted from Data Frame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, Data Frame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into Data Frame objects
- Intelligent label-based slicing, fancy indexing, and sub setting of large data sets
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Hierarchical labelling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format

## 3.2 MODELLING

### 3.2.1 Scikit Learn (SKLearn)

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, and is designed to interoperate with the Python numerical and scientific library NumPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

### 3.2.3 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

**Guiding principles:**

- **User friendliness:** Keras is an API designed for human beings, not machines. It puts user experience front and centre. Keras follows best practices for reducing cognitive load it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

- **Modularity:** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.

- **Easy extensibility:** New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

- **Work with Python:** No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

# 5. WORKING

## 5.1 Preparing the Data

- Convert to lowercase.
- Replace contractions with their longer forms.
- Remove any unwanted characters (this step needs to be done after replacing the contractions because apostrophes will be removed. Notice the backward slash before the hyphen. Without this slash, all characters that are 'between' the characters before and after the hyphen would be removed. This can create some unwanted effects. To give you an example, typing "a-d" would remove a, b, c, d.).
- Stop words will only be removed from the descriptions. They are not very relevant in training the model, so by removing them we are able to train the model faster because there is less data. They will remain in the summaries because they are rather short and I would prefer for them to sound more like natural phrases.

## 5.2 Modelling

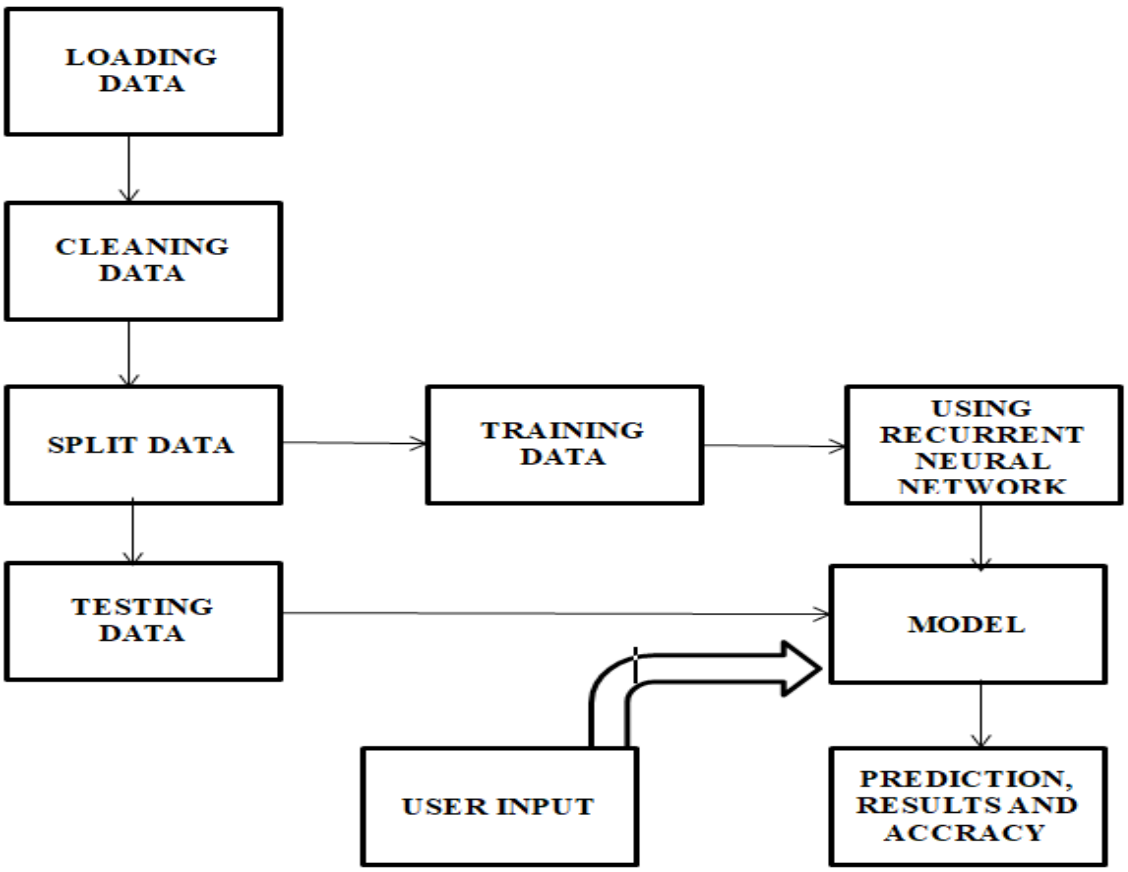
To generate model we use Sequence to Sequence model with attention using Recurrent Neural Network. With training data RNN is provided to generate the model that will be used to predict. The model generated is saved as file and utilized to predict for input summary. And with that we can use testing data to test and predict the accuracy and loss using the model available. To train we run the input sentence through the encoder, and keep track of every output and the latest hidden state. Then the decoder is given the <SOS> token as its first input, and the last hidden state of the encoder as its first hidden state. "Teacher forcing" is the concept of using the real target outputs as each next input, instead of using the decoder's guess as the next input. Using teacher forcing causes it to converge faster but when the trained network is exploited, it may exhibit instability. We can observe outputs of teacher-forced networks that read with coherent grammar but wander far from the correct translation - intuitively it has learned to represent the output grammar and can "pick up" the meaning once the teacher tells it the first few words, but it has not properly learned how to create the sentence from the translation in the first place.

### 5.3. Testing

Model generated using Recurrent Neural Network is saved as file. This model is then used for testing for accuracy and loss, data which we used to train is split in train and test part and Test part is used to test for the accuracy given by the model generated after training it on train data.

### 5.4. Prediction

Finally user can get result from input text provided by them. Model generated is used to predict, as text provided input is converted to summarize text that act as final output which is displayed through interface through which user and machine are communicating.




## WORKING MODEL

## 6.Conclusion

People need to learn much from texts, but they tend to want to spend less time while doing this. It aims to solve this problem by supplying them the summaries of the text from which they want to

gain information. Goals of this project are that these summaries will be as important as possible in the aspect of the texts intention. The user will be eligible to select the summary length. Supplying the user, a smooth and clear interface to use the facility in order to latest technology and applying it for saving time and increasing efficiency and productivity of their work.

Machine Learning provides us important technique to handle situation which are related to human behaviour and used on daily basis that allow us to save time and money and focus on important goal. Text summarizer provides users to help use of machine learning algorithm and to use for their benefit as reading is an important part of life. And to read less and gain more information from the document and text we have to handle in our life we need to be faster and focus on important piece of information rather than wasting our whole time on reading unnecessary text and to avoid this text summarizer provides us feature that we can use to apply in real life.

## 7.Refrences


- https://www.wikipedia.org/
- https://towardsdatascience.com/text-summarization-with-amazon-reviews-41801c2210b
- Deep Learning with Keras by Antonio Gulli and Sujit Pal