Don Bosco Institute and Technology, Kurla(W), Mumbai-400 070

DEPARTMENT OF COMPUTER ENGINEERING

(Session 2020-2021 EVEN)

Subject: Computational Lab II (CSL 804)

Final Year Computer Engineering

"AUTOMATIC SUMMARIZATION OF SCIENTIFIC PAPERS AND WEB ARTICLES"

A Project Report

Submitted by:

Sr. No.	*Roll No.*	*Name of Student*
1.	*10*	*Reuben Crasto*
2.	*16*	*Lance Dias*
3.	*22*	*Salil Fernandes*

DECLARATION

We hereby declare that the project entitled "Automatic Summarization of Scientific Papers and Web Articles" submitted for the Computational Lab II (CSL 804) of final year (Computer Engineering) of Mumbai University syllabus course work is our original work/hypothesis/algorithm design/mathematical modelling and result analysis for the specific Machine Learning algorithm.

[illegible] [illegible] [illegible]

Reuben Crasto **Lance Dias** **Salil Fernandes**

Place: Mumbai

Date: 10/05/2021

Don Bosco Institute and Technology, Kurla(W), Mumbai-400 070

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the project titled "Automatic Summarization of Scientific Papers and Web Articles" is the bonafide work carried out by **Reuben Crasto**, **Lance Dias & Salil Fernandes,** the students of Final Year, Department of Computer Engineering, Don Bosco Institute of Technology, Kurla (W), Mumbai-400 070 affiliated to Mumbai University, Mumbai, Maharashtra (India) during the academic year 2019-2020, in coursework completion of subject Computational LabII (CSL 804) of 8th semester.

Dr. Shaikh Phiroj	
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

Place : Mumbai

Date : 10/05/2021

ABSTRACT

The scientific research process generally starts with the examination of the state of the art, which may involve a vast number of publications. Automatically summarizing scientific articles would help researchers in their investigation by speeding up the research process. The automatic summarization of scientific articles differs from the summarization of generic texts due to their specific structure and inclusion of citation sentences. Most of the valuable information in scientific articles is presented in tables, figures, and algorithm pseudocode. These elements, however, do not usually appear in a generic text. Therefore, several approaches that consider the particularity of a scientific article structure were proposed to enhance the quality of the generated summary. This project tries to make life easier for those people who regularly read research papers and always look to summarize their learnings. It creates a supervised learning-based system that can do a summarization of the scientific papers. With regards to web scraping and information extraction, similar to scientific papers, the same problem and concept is applicable. Users often find it tedious to browse through an entire webpage before they find the information which is of their interest. Reading an entire webpage is required just to find out the pieces of information that were required by a user. This process is laborious and time consuming on an average. Consider a scenario of a web article that is reporting outcome of the final of a football World Cup. The article may contain several paragraphs regarding the build up to the final and only mid-way will inform the user which team won the World Cup. Thus, the user would have to process so much unwanted information before they read information that they wanted to know. In such scenarios a text summarizer would be able to save time and effort by giving the user important information in short sothat they can get a gist of the article without having to scarp through tonnes of text.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Text summarization refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document. Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning.

Automatic text summarization is a common problem in machine learning and natural language processing (NLP). Machine learning models are usually trained to understand documents and distil the useful information before outputting the required summarized texts.

There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is *generic summarization*, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is *query relevant summarization*, sometimes called *query-based summarization*, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.

An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). This problem is called multi-document summarization. A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.

Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set of images from a larger set of images. A summary in this context is useful to show the most representative images of results in an image collection exploration system. Video summarization is a related domain, where the system automatically creates a trailer of a long video. This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions. Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured.

At a very high level, summarization algorithms try to find subsets of objects (like set of sentences, or a set of images), which cover information of the entire set. This is also called the *core-set*. These algorithms model notions like diversity, coverage, information and representativeness of the summary. Query based summarization techniques, additionally model for relevance of the summary with the query. Some techniques and algorithms which naturally model summarization problems are TextRank and PageRank, Submodular set function, Determinantal point process, maximal marginal relevance (MMR) etc.

CHAPTER 2: LITERATURE SURVEY

2.1 SURVEY OF EXISTING SYSTEM

Paper Title	**Year**	**Idea**
Extractive Text Summarization Using Sentence Ranking	2019	The paper presents an extractive summarization technique. The input text is tokenised into sentences which are broken into words and the frequencies of occurrence are calculated. The words with higher frequencies are joined to form meaningful sentences in the output summary
Automatic text summarization based on sentences clustering and extraction	2009	The paper presents the idea of forming sentence clusters based on different measures of similarity like Word Form Similarity, Word Order Similarity, Word Semantic Similarity and Sentence Similarity. The k-clustering approach is used and after sentences are divided into k-clusters a priority extraction of sentences from each cluster is done to form the final summary.
Extractive Text Summarization using Deep Learning	2018	The idea of using a deep learning approach is explored in this paper. Extractive text summarization is performed. The proposed model performs pre-processing of text using nltk followed by sentence feature extraction to calculate sentence scores based on bigrams, trigrams and cosine similarity. A restricted Boltzmann machine neural network produces one summary and fuzzy logic produces a second summary. The final output is a combination of the two summaries.
An approach to Abstractive Text Summarization	2013	This paper is one of the few papers that focuses on abstractive type of summarization. The summary is produced from the main ideas in the input text rather than copying its most salient features. A word graph is constructed using nodes to represent word and their POS tags while edges represent adjacency relations between words. Keywords are extracted by computing tf-isf(term frequency - inverted sentence frequency) and getting top k per cent keywords with highest tf-isf.
A Survey on NLP based Text Summarization for Summarizing Product Reviews	2020	The paper analyses various techniques used to summarize reviews of online products. The methods explained include extractive text summarization, generic text summarization, domain specific text summarization, query-based text summarization and abstractive text summarization. The paper concludes that domain specific summarization produces the higher accuracy result as the other methods have no prior knowledge about the input text.
Automatic Text Summarization of News Articles	2017	The paper presents a method of extractive summarization of news articles. The technique involves first generating POS tags of individual words and then generating lexical chains which are groups of semantically related words. After generating scores the strong lexical chains are used to identify the more frequently occurring sentences in order to generate the final summary.

2.2 PROBLEMS OF EXISTING SYSTEMS / RESEARCH GAP

The main problems of the existing text summarizers is the lack of good quality training datasets. When neural networks are trained for summarization it is often found that all scenarios are never covered because it is difficult to find a dataset that has every possible combination of nouns, pronouns, adjectives, verb, prepositions, etc. Another gap that is noticed in the field of summarization in NLP is the relatively lower accuracy of the output summaries being produced whether an abstractive summarization is performed or an extractive summarization is performed. Processes like chunking, POS Tagging, Lexical and Morphological analysis and frequency counting all help in identifying words and sentences that would form a key part of the final summary but only to a small extent. Understanding the gist or idea being conveyed by the text by a computer program still remains an unsolved task in NLP.

2.3 PROBLEM STATEMENT AND OBJECTIVE

With such a big amount of data circulating in the digital space, there is need to develop machine learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages. Daily on an average millions of new web articles and scientific research papers are being generated. It becomes difficult to keep up with the latest trends and going through so much information. Our project aims to create an automatic text summarizer which would produce summaries of high accuracy and meaning using traditional techniques of tokenization, tagging and frequency counting to mention a few. The summarizer would be able to scrap through web article text as well as scientific papers and summarize the general idea being conveyed in just

2.4 SCOPE

Our project has a tremendous scope of exploration and improvement since summarization is a vast area of research in natural language processing. With vast amounts of text data being generated on a daily basis, more and more datasets with better word combinations covering a wider range of words or lexemes can be created to train summarization models. As computational power increases models can be trained to give a higher accuracy faster. More programming libraries containing in built functions for performing summarization tasks can be developed which would make it easier to build a summarization model.

CHAPTER 3: PROPOSED SYSTEM

3.1 Analysis

Broadly, there are two approaches to summarizing texts in NLP: extraction and abstraction.

Extraction-based summarization

In extraction-based summarization, a subset of words that represent the most important points is pulled from a piece of text and combined to make a summary. Think of it as a highlighter—which selects the main information from a source text.

In machine learning, extractive summarization usually involves weighing the essential sections of sentences and using the results to generate summaries. Different types of algorithms and methods can be used to gauge the weights of the sentences and then rank them according to their relevance and similarity with one another—and further joining them to generate a summary.

Abstraction-based summarization

In abstraction-based summarization, advanced deep learning techniques are applied to paraphrase and shorten the original document, just like humans do. Think of it as a pen—which produces novel sentences that may not be part of the source document.

Since abstractive machine learning algorithms can generate new phrases and sentences that represent the most important information from the source text, they can assist in overcoming the grammatical inaccuracies of the extraction techniques. Here is an example:

Although abstraction performs better at text summarization, developing its algorithms requires complicated deep learning techniques and sophisticated language modelling.
To generate plausible outputs, abstraction-based summarization approaches must address a wide variety of NLP problems, such as natural language generation, semantic representation, and inference permutation. As such, extractive text summarization approaches are still more popular.

In our project the first method we will be implementing for summarization is a generic extractive based approach.

The second method we will implement for text summarization is based on the Text Rank Algorithm.

TextRank is a general purpose, **graph based** ranking algorithm for NLP.

TextRank is an automatic summarization technique.

TextRank Model -

The basic idea implemented by a graph-based ranking model is that of *voting* or *recommendation.*

When one vertex links to another one, it is basically casting a vote for that vertex. The higher the number of votes cast for a vertex, the higher the importance of that vertex.

Text as a graph -

We have to build a graph that represents the text, interconnects words or other text entities with meaningful relations.

TextRank includes two NLP tasks-

1. Keyword extraction task
2. Sentence extraction task

Keyword Extraction -

The task of keyword extraction algorithm is to automatically identify in a text a set of terms that best describe the document.

The simplest possible approach is to use a frequency criterion.

HOWEVER, this leads to poor results.

The TextRank keyword extraction algorithm is fully unsupervised. No training is necessary.

Sentence Extraction -

TextRank is very well suited for applications involving entire sentences, since it allows for a ranking over text units that is recursively computed based on information drawn from the entire text.

To apply TextRank, we first build a graph associated with the text, where the graph vertices are representative for the units to be ranked. The goal is to rank entire sentences, therefore, a vertex is added to the graph for each sentence in the text.

The PageRank Algorithm is the foundation of TextRank and is used by Google search. It computes the rank of web pages. It assumes that important pages are linked by important pages. The PageRank valueof a page is the probability of a user visiting that page. In TextRank, the only difference is that we consider sentences instead of pages.

3.2 DETAILS OF HARDWARE AND SOFTWARE

Both the summarization methods will be implemented in the Python programming language. For the extractive summarization method the following Python libraries will be used:

NLTK

NLTK (Natural Language Toolkit) is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response. The tokenizer provided by this library will be used.

RE

A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing). Python's re package will be used to write regular expressions needed to remove special characters, necessary spaces, hyperlinks and other types of references from the input text.

Gensim

Gensim is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. Target audience is the natural language processing (NLP) and information retrieval (IR) community. It will be used in the pre-processing step along with regular expressions to clean the text and prepare it for summarization.

HeapQ

This is Python's library that is used to implement a priority queue data structure. Sentences would be arranged in descending order of their scores similar to a priority queue and this library will select the first N sentences with the best scores as the final summary.

For the text summarization using the TextRank Algorithm we will use the following Python libraries:

NLTK

NLTK (Natural Language Toolkit) is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response. The tokenizer provided by this library will be used.

Networkx

NetworkX is a Python library for studying graphs and networks. The page rank algorithm function from this library will be used on the similarity matrix for computing the most important sentences.

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. This library will be needed to prepare the similarity matrix as well as the vector representation of sentences.

Pandas

pandas is a software library written for the Python programming language for data manipulation and analysis. This library will be used for text cleaning which includes removal of punctuation, unnecessary spaces, numbers and special characters.

3.3 METHODOLOGY

Method 1:

In our first approach which is more of a generalised way of summarization we will make use of the following steps:

a. Converting Paragraphs to Sentences: The most common way of converting paragraphs to sentences is to split the paragraph whenever a period is encountered.

b. Text Pre-processing

After converting paragraph to sentences, we need to remove all the special characters, stop words and numbers from all the sentences.

c. Sentence Tokenisation

We need to tokenize all the sentences to get all the words that exist in the sentences. A list of words is obtained as a result of this step.

d. Find Weighted Frequency of Occurrence

Next, we need to find the weighted frequency of occurrences of all the words. We can find the weighted frequency of each word by dividing its frequency by the frequency of the most occurring word.

e. Replace Words by Weighted Frequencies in Original Sentences

The final step is to plug the weighted frequency in place of the corresponding words in original sentences and finding their sum. It is important to mention that weighted frequency for the words removed during pre-processing (stop words, punctuation, digits etc.) will be zero and therefore is not required to be added.

f. Sort Sentences in Descending order of Sum

The final step is to sort the sentences in inverse order of their sum. The sentences with highest frequencies summarize the text.

Method 2:

In our second approach we will use the Text Rank Algorithm which is very similar to the Page Rank algorithm used for browsing web links on the internet. In place of web pages, we use sentences. Similarity between any two sentences is used as an equivalent to the web page transition probability. The similarity scores are stored in a square matrix, similar to the matrix M used for PageRank.

TextRank is an extractive and unsupervised text summarization technique.

Steps:

- The first step would be to concatenate all the text contained in the articles
- Then split the text into individual sentences.
- In the next step, we will find vector representation (word embeddings) for each and every sentence.
- Similarities between sentence vectors are then calculated and stored in a matrix.
- The similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation.
- Finally, a certain number of top-ranked sentences form the final summary.

CHAPTER 4 : IMPLEMENTATION AND RESULT ANALYSIS

4.1 PROGRAMS

A. Extractive Text Summariser

```
mport bs4 as bs
import urllib.request
import re
import heapq
import nltk
#import pdftotext

#nltk.download('punkt')
#nltk.download('stopwords')

scraped_data = urllib.request.urlopen('https://en.wikipedia.org/wiki/Artificial_Intelligence')
article = scraped_data.read()

#article = open("paper.html", "r")

parsed_article = bs.BeautifulSoup(article,'lxml')

paragraphs = parsed_article.find_all('p')

article_text = ""

for p in paragraphs:
    article_text += p.text

# Removing Square Brackets and Extra Spaces
article_text = re.sub(r'\[[0-9]*\]', ' ', article_text)
article_text = re.sub(r'\s+', ' ', article_text)

# Removing special characters and digits
formatted_article_text = re.sub('[^a-zA-Z]', ' ', article_text )
formatted_article_text = re.sub(r'\s+', ' ', formatted_article_text)

sentence_list = nltk.sent_tokenize(article_text)

stopwords = nltk.corpus.stopwords.words('english')

word_frequencies = {}
for word in nltk.word_tokenize(formatted_article_text):
    if word not in stopwords:
        if word not in word_frequencies.keys():
            word_frequencies[word] = 1
        else:
            word_frequencies[word] += 1

maximum_frequncy = max(word_frequencies.values())

for word in word_frequencies.keys():
    word_frequencies[word] = (word_frequencies[word]/maximum_frequncy)
```

```
sentence_scores = {}
for sent in sentence_list:
    for word in nltk.word_tokenize(sent.lower()):
        if word in word_frequencies.keys():
            if len(sent.split(' ')) < 30:
                if sent not in sentence_scores.keys():
                    sentence_scores[sent] = word_frequencies[word]
                else:
                    sentence_scores[sent] += word_frequencies[word]


summary_sentences = heapq.nlargest(15, sentence_scores, key=sentence_scores.get)

summary = ' '.join(summary_sentences)
print(summary)
```

B. Summarization Using Text Rank Algorithm

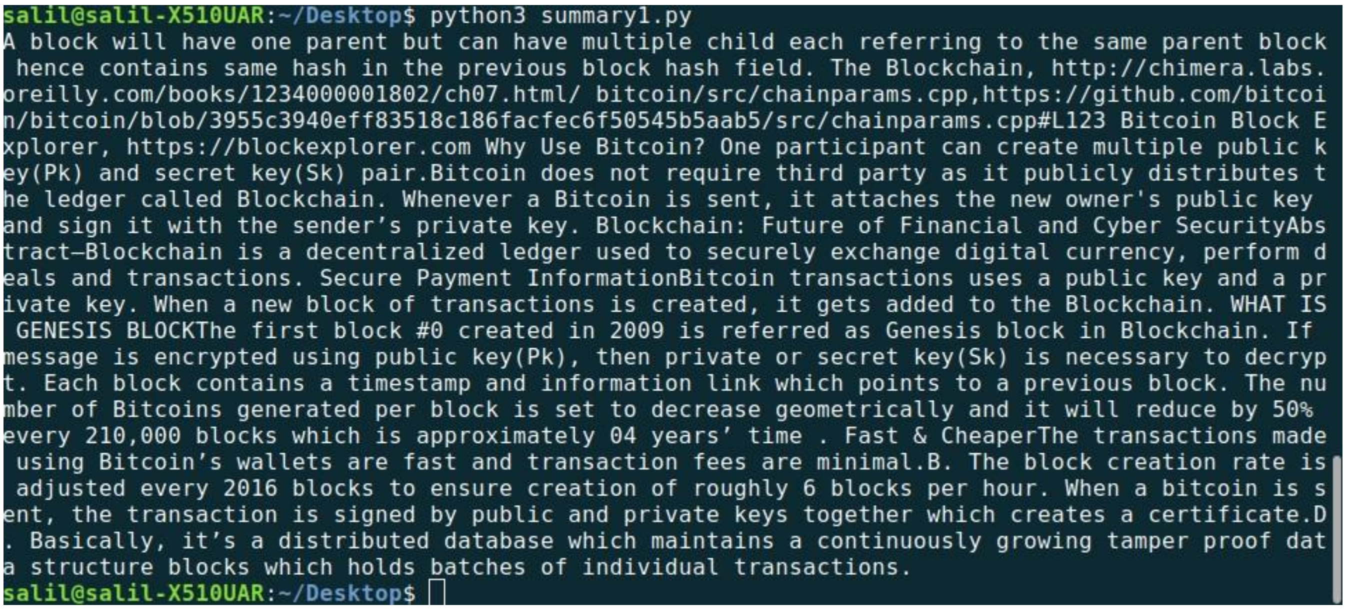
```
import numpy as np
import pandas as pd
import nltk
import  re
#import pdftotext
import bs4 as bs
import urllib.request
import networkx as nx

from nltk.tokenize import sent_tokenize
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import cosine_similarity

#nltk.download('punkt')
#nltk.download('stopwords')

scraped_data = urllib.request.urlopen('https://en.wikipedia.org/wiki/Artificial_Intelligence')
article = scraped_data.read()

#article = open("paper.html", "r")

parsed_article = bs.BeautifulSoup(article,'lxml')

paragraphs = parsed_article.find_all('p')

article_text = ""

for p in paragraphs:
    article_text += p.text

# Removing Square Brackets and Extra Spaces
article_text = re.sub(r'\[[0-9]*\]', ' ', article_text)
article_text = re.sub(r'\s+', ' ', article_text)

# Removing special characters and digits
```

```
formatted_article_text = re.sub('[^a-zA-Z]', ' ', article_text )
formatted_article_text = re.sub(r'\s+', ' ', formatted_article_text)

sentences = []
sentences.append(sent_tokenize(article_text))

sentences = [y for x in sentences for y in x] # flatten list
word_embeddings = {}
f = open('glove.6B.100d.txt', encoding='utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    word_embeddings[word] = coefs
f.close()
len(word_embeddings)

# remove punctuations, numbers and special characters
clean_sentences = pd.Series(sentences).str.replace("[^a-zA-Z]", " ")

# make alphabets lowercase
clean_sentences = [s.lower() for s in clean_sentences]

stop_words = stopwords.words('english')
# function to remove stopwords
def remove_stopwords(sen):
    sen_new = " ".join([i for i in sen if i not in stop_words])
    return sen_new
# remove stopwords from the sentences
clean_sentences = [remove_stopwords(r.split()) for r in clean_sentences]

# Extract word vectors
word_embeddings = {}
f = open('glove.6B.100d.txt', encoding='utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    word_embeddings[word] = coefs
f.close()
sentence_vectors = []
for i in clean_sentences:
  if len(i) != 0:
    v = sum([word_embeddings.get(w, np.zeros((100,))) for w in i.split()])/(len(i.split())+0.001)
  else:
    v = np.zeros((100,))
  sentence_vectors.append(v)

  # similarity matrix
sim_mat = np.zeros([len(sentences), len(sentences)])
for i in range(len(sentences)):
  for j in range(len(sentences)):
    if i != j:
      sim_mat[i][j] = cosine_similarity(sentence_vectors[i].reshape(1,100),
sentence_vectors[j].reshape(1,100))[0,0]

nx_graph = nx.from_numpy_array(sim_mat)
scores = nx.pagerank(nx_graph)
ranked_sentences = sorted(((scores[i],s) for i,s in enumerate(sentences)), reverse=True)
# Extract top 10 sentences as the summary
for i in range(10):
  print(ranked_sentences[i][1])
```

4.2 OUTPUT/RESULT

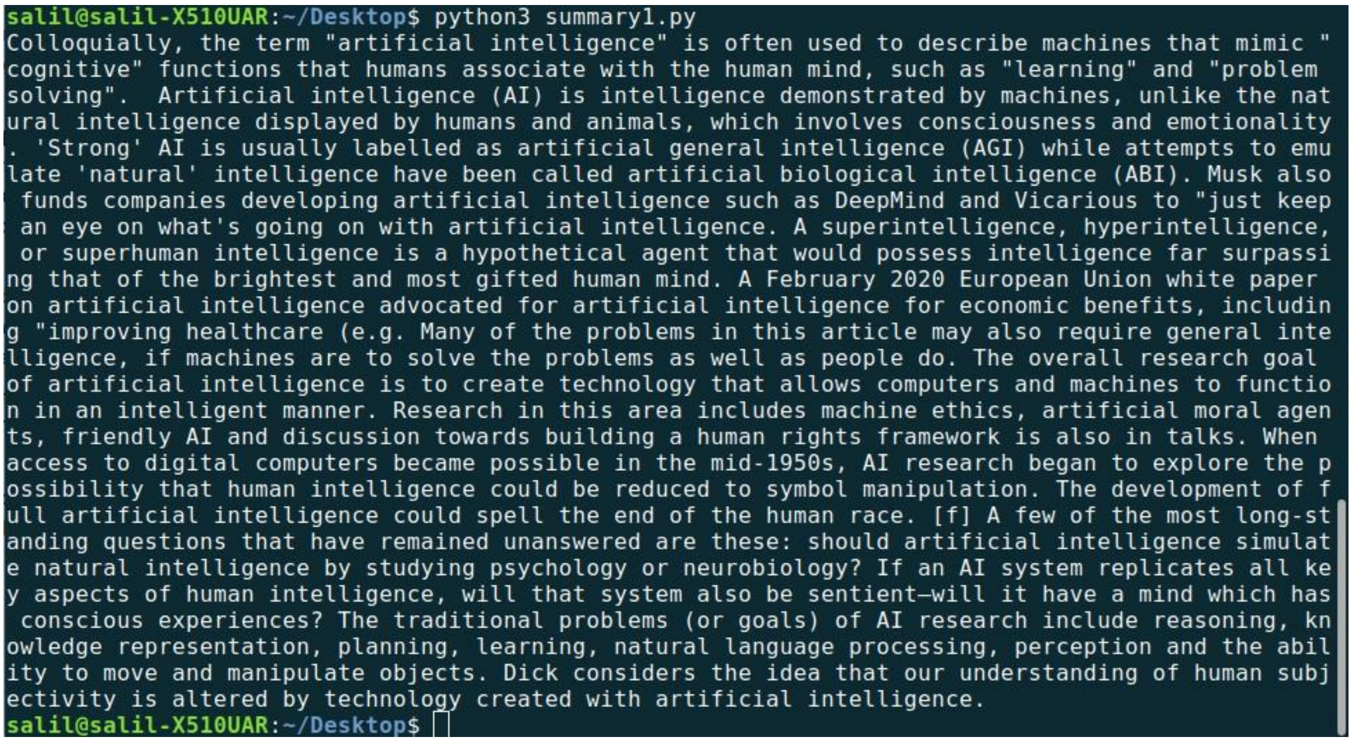
The scientific paper we have used as input to both the summarisers is an IEEE Paper entitled "Blockchain: Future of Financial and Cyber Security" published in the year 2016 by Sachchidanand Singh and Nirmala Singh.
The web article which we have used as input for the summarisers is the Wikipedia article on Artificial Intelligence.
A. Output OF THE Extractive SUMMARISER

For THE Paper:



```
salil@salil-X510UAR:~/Desktop$ python3 summary1.py
A block will have one parent but can have multiple child each referring to the same parent block
 hence contains same hash in the previous block hash field. The Blockchain, http://chimera.labs.
oreilly.com/books/1234000001802/ch07.html/ bitcoin/src/chainparams.cpp,https://github.com/bitcoi
n/bitcoin/blob/3955c3940eff83518c186facfec6f50545b5aab5/src/chainparams.cpp#L123 Bitcoin Block E
xplorer, https://blockexplorer.com Why Use Bitcoin? One participant can create multiple public k
ey(Pk) and secret key(Sk) pair.Bitcoin does not require third party as it publicly distributes t
he ledger called Blockchain. Whenever a Bitcoin is sent, it attaches the new owner's public key
and sign it with the sender's private key. Blockchain: Future of Financial and Cyber SecurityAbs
tract—Blockchain is a decentralized ledger used to securely exchange digital currency, perform d
eals and transactions. Secure Payment InformationBitcoin transactions uses a public key and a pr
ivate key. When a new block of transactions is created, it gets added to the Blockchain. WHAT IS
 GENESIS BLOCKThe first block #0 created in 2009 is referred as Genesis block in Blockchain. If
message is encrypted using public key(Pk), then private or secret key(Sk) is necessary to decryp
t. Each block contains a timestamp and information link which points to a previous block. The nu
mber of Bitcoins generated per block is set to decrease geometrically and it will reduce by 50%
every 210,000 blocks which is approximately 04 years' time . Fast & CheaperThe transactions made
 using Bitcoin's wallets are fast and transaction fees are minimal.B. The block creation rate is
 adjusted every 2016 blocks to ensure creation of roughly 6 blocks per hour. When a bitcoin is s
ent, the transaction is signed by public and private keys together which creates a certificate.D
. Basically, it's a distributed database which maintains a continuously growing tamper proof dat
a structure blocks which holds batches of individual transactions.
salil@salil-X510UAR:~/Desktop$
```

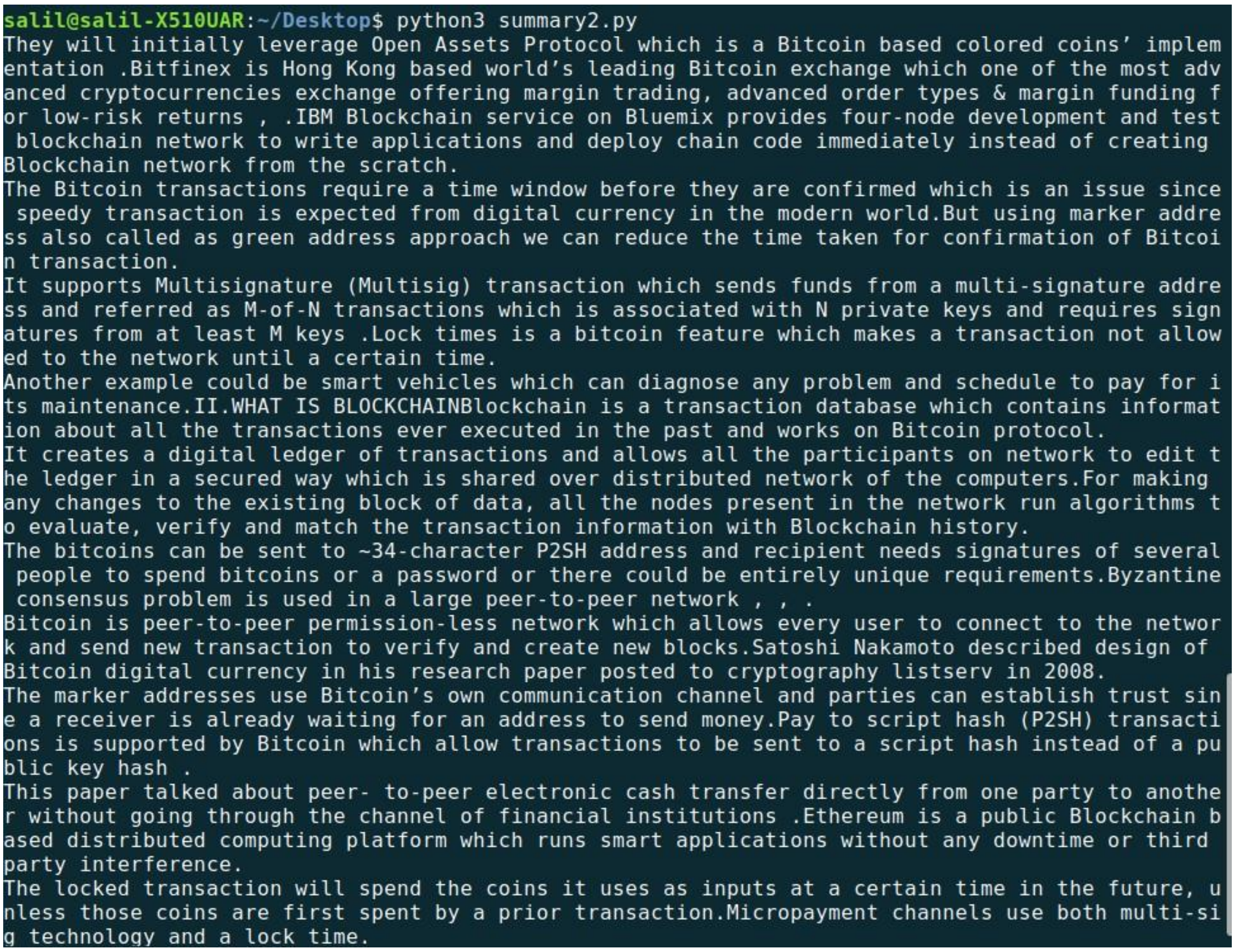
For the web article:



```
salil@salil-X510UAR:~/Desktop$ python3 summary1.py
Colloquially, the term "artificial intelligence" is often used to describe machines that mimic "
cognitive" functions that humans associate with the human mind, such as "learning" and "problem
solving".  Artificial intelligence (AI) is intelligence demonstrated by machines, unlike the nat
ural intelligence displayed by humans and animals, which involves consciousness and emotionality
. 'Strong' AI is usually labelled as artificial general intelligence (AGI) while attempts to emu
late 'natural' intelligence have been called artificial biological intelligence (ABI). Musk also
 funds companies developing artificial intelligence such as DeepMind and Vicarious to "just keep
 an eye on what's going on with artificial intelligence. A superintelligence, hyperintelligence,
 or superhuman intelligence is a hypothetical agent that would possess intelligence far surpassi
ng that of the brightest and most gifted human mind. A February 2020 European Union white paper
on artificial intelligence advocated for artificial intelligence for economic benefits, includin
g "improving healthcare (e.g. Many of the problems in this article may also require general inte
lligence, if machines are to solve the problems as well as people do. The overall research goal
of artificial intelligence is to create technology that allows computers and machines to functio
n in an intelligent manner. Research in this area includes machine ethics, artificial moral agen
ts, friendly AI and discussion towards building a human rights framework is also in talks. When
access to digital computers became possible in the mid-1950s, AI research began to explore the p
ossibility that human intelligence could be reduced to symbol manipulation. The development of f
ull artificial intelligence could spell the end of the human race. [f] A few of the most long-st
anding questions that have remained unanswered are these: should artificial intelligence simulat
e natural intelligence by studying psychology or neurobiology? If an AI system replicates all ke
y aspects of human intelligence, will that system also be sentient—will it have a mind which has
 conscious experiences? The traditional problems (or goals) of AI research include reasoning, kn
owledge representation, planning, learning, natural language processing, perception and the abil
ity to move and manipulate objects. Dick considers the idea that our understanding of human subj
ectivity is altered by technology created with artificial intelligence.
salil@salil-X510UAR:~/Desktop$ 
```

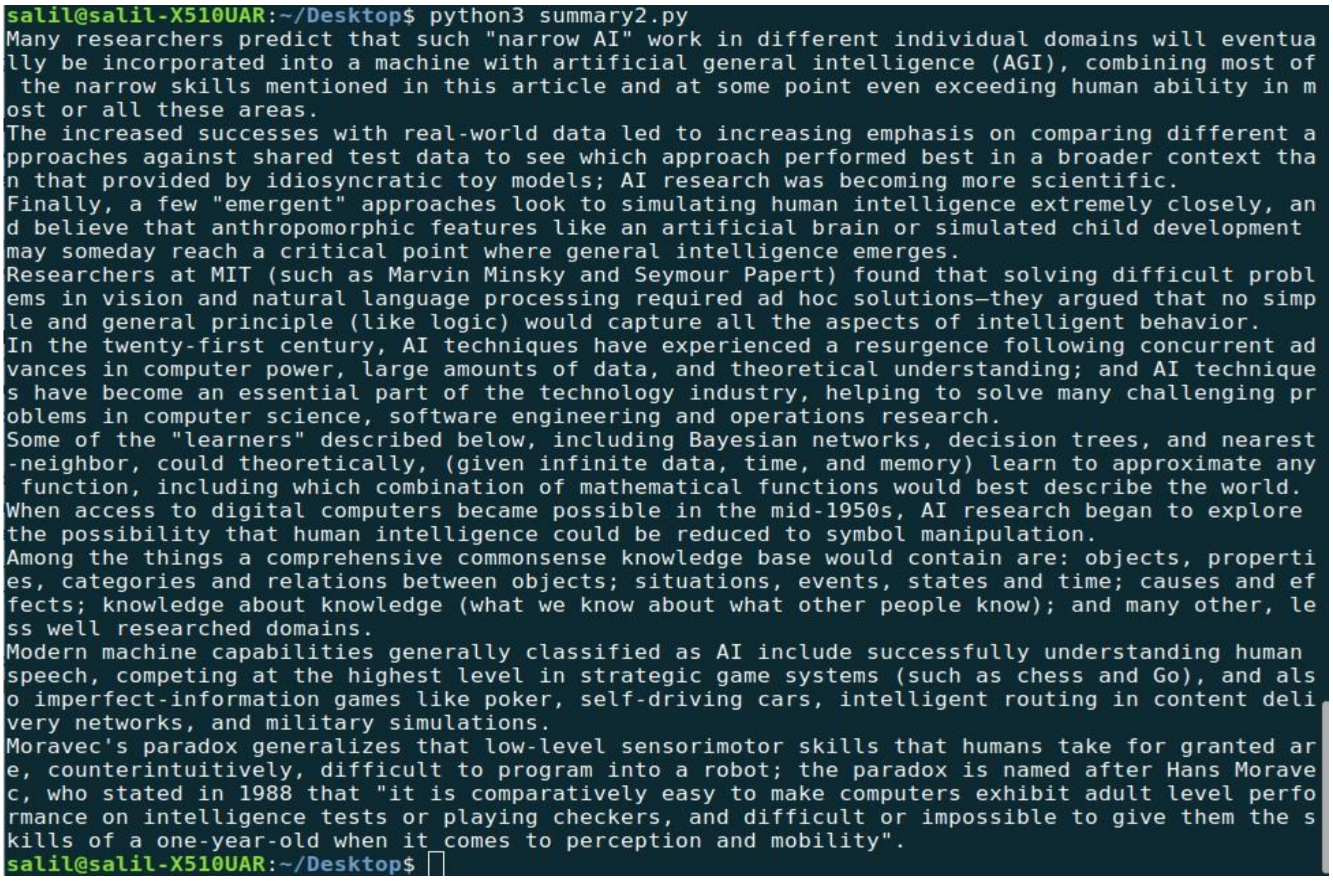
B. Output OF THE TEXT RANK ALGORITHM SUMMARISER

For THE Paper:



```
salil@salil-X510UAR:~/Desktop$ python3 summary2.py
They will initially leverage Open Assets Protocol which is a Bitcoin based colored coins' implem
entation .Bitfinex is Hong Kong based world's leading Bitcoin exchange which one of the most adv
anced cryptocurrencies exchange offering margin trading, advanced order types & margin funding f
or low-risk returns , .IBM Blockchain service on Bluemix provides four-node development and test
 blockchain network to write applications and deploy chain code immediately instead of creating
Blockchain network from the scratch.
The Bitcoin transactions require a time window before they are confirmed which is an issue since
 speedy transaction is expected from digital currency in the modern world.But using marker addre
ss also called as green address approach we can reduce the time taken for confirmation of Bitcoi
n transaction.
It supports Multisignature (Multisig) transaction which sends funds from a multi-signature addre
ss and referred as M-of-N transactions which is associated with N private keys and requires sign
atures from at least M keys .Lock times is a bitcoin feature which makes a transaction not allow
ed to the network until a certain time.
Another example could be smart vehicles which can diagnose any problem and schedule to pay for i
ts maintenance.II.WHAT IS BLOCKCHAINBlockchain is a transaction database which contains informat
ion about all the transactions ever executed in the past and works on Bitcoin protocol.
It creates a digital ledger of transactions and allows all the participants on network to edit t
he ledger in a secured way which is shared over distributed network of the computers.For making
any changes to the existing block of data, all the nodes present in the network run algorithms t
o evaluate, verify and match the transaction information with Blockchain history.
The bitcoins can be sent to ~34-character P2SH address and recipient needs signatures of several
 people to spend bitcoins or a password or there could be entirely unique requirements.Byzantine
 consensus problem is used in a large peer-to-peer network , , .
Bitcoin is peer-to-peer permission-less network which allows every user to connect to the networ
k and send new transaction to verify and create new blocks.Satoshi Nakamoto described design of
Bitcoin digital currency in his research paper posted to cryptography listserv in 2008.
The marker addresses use Bitcoin's own communication channel and parties can establish trust sin
e a receiver is already waiting for an address to send money.Pay to script hash (P2SH) transacti
ons is supported by Bitcoin which allow transactions to be sent to a script hash instead of a pu
blic key hash .
This paper talked about peer- to-peer electronic cash transfer directly from one party to anothe
r without going through the channel of financial institutions .Ethereum is a public Blockchain b
ased distributed computing platform which runs smart applications without any downtime or third
party interference.
The locked transaction will spend the coins it uses as inputs at a certain time in the future, u
nless those coins are first spent by a prior transaction.Micropayment channels use both multi-si
g technology and a lock time.
```

For the web article:



```
salil@salil-X510UAR:~/Desktop$ python3 summary2.py
Many researchers predict that such "narrow AI" work in different individual domains will eventua
lly be incorporated into a machine with artificial general intelligence (AGI), combining most of
 the narrow skills mentioned in this article and at some point even exceeding human ability in m
ost or all these areas.
The increased successes with real-world data led to increasing emphasis on comparing different a
pproaches against shared test data to see which approach performed best in a broader context tha
n that provided by idiosyncratic toy models; AI research was becoming more scientific.
Finally, a few "emergent" approaches look to simulating human intelligence extremely closely, an
d believe that anthropomorphic features like an artificial brain or simulated child development
may someday reach a critical point where general intelligence emerges.
Researchers at MIT (such as Marvin Minsky and Seymour Papert) found that solving difficult probl
ems in vision and natural language processing required ad hoc solutions—they argued that no simp
le and general principle (like logic) would capture all the aspects of intelligent behavior.
In the twenty-first century, AI techniques have experienced a resurgence following concurrent ad
vances in computer power, large amounts of data, and theoretical understanding; and AI technique
s have become an essential part of the technology industry, helping to solve many challenging pr
oblems in computer science, software engineering and operations research.
Some of the "learners" described below, including Bayesian networks, decision trees, and nearest
-neighbor, could theoretically, (given infinite data, time, and memory) learn to approximate any
 function, including which combination of mathematical functions would best describe the world.
When access to digital computers became possible in the mid-1950s, AI research began to explore
the possibility that human intelligence could be reduced to symbol manipulation.
Among the things a comprehensive commonsense knowledge base would contain are: objects, properti
es, categories and relations between objects; situations, events, states and time; causes and ef
fects; knowledge about knowledge (what we know about what other people know); and many other, le
ss well researched domains.
Modern machine capabilities generally classified as AI include successfully understanding human
speech, competing at the highest level in strategic game systems (such as chess and Go), and als
o imperfect-information games like poker, self-driving cars, intelligent routing in content deli
very networks, and military simulations.
Moravec's paradox generalizes that low-level sensorimotor skills that humans take for granted ar
e, counterintuitively, difficult to program into a robot; the paradox is named after Hans Morave
c, who stated in 1988 that "it is comparatively easy to make computers exhibit adult level perfo
rmance on intelligence tests or playing checkers, and difficult or impossible to give them the s
kills of a one-year-old when it comes to perception and mobility".
salil@salil-X510UAR:~/Desktop$
```

RESULT ANALYSIS

We implemented two techniques to summarize documents and web articles:
a. A generic extractive summarization method
b. Text Rank Algorithm based summarization

Parameter	**Comment**	
	Extractive Summarization	**Text Rank Based Summarization**
Accuracy	After reading the summaries of the web article and the IEEE paper that were generated it was observer that the summary was a little less adhering to the subject matter of the original documents. There were some words in the summary that perhaps were not the best choice to be present in the summary but overall, the summary was decent enough. The accuracy of the summary would be around the 80-85% mark.	When comparing the summaries that were generated by this algorithm with those that were generated by the extractive method, these summaries were very impressive. Not only did this method produce summaries that stuck to the subject matter of the original text, they also contained very appropriate words pertaining to the subject. Each of the sentences also made a lot more sense. Overall the accuracy rating for this method is around the 95-100% mark.
Efficiency	The program for this method runs a lot faster and gives an almost instantaneous output summary. The program only has to execute simple tasks like splitting of paragraphs into sentences, sentence to word tokenisation and frequency of occurrence calculations for individual words which run is a constant or even linear time. The **running time complexity was found to be O(1) for**	The program for this algorithm takes a while to execute and generate the output summary. In addition to the mandatory steps like splitting of paragraphs into sentences, sentence to word tokenisation and the preprocessing steps like removal of special characters,numbers and stopwords all of which run in constant time the algorithm requires a special process of generating a similarity matrix between sentences and then creating a graph of

	most cases and O(n) if the input text was very lengthy. Thus the faster speed and higher efficiency comes at a trade off for the accuracy of the output summary.	nodes. On this graph a rank calculation step is performed. For a matrix of order n x n a graph of n nodes is generated and the calculation will take O(n) time. Hence the final **running time complexity is O(n^2).** The lower efficiency is worth it as the output summary is better and highly accurate.
Length of Output	This method generated summaries that were on an average 30-40 words shorter than the summaries generated by the text rank algorithm. This is understandable given the fact that the program was running very quickly in a constant time.	This algorithm was generating larger summaries compared to the extractive method which is justified and appreciated given the fact that the program takes a while longer to execute.
Precision	This method gave a precision score of 83%.	This method gave a precision score of 90%.
Recall	This method gave a precision score of 79%.	This method gave a precision score of 87%.
F-Measure	This method gave a precision score of 85%.	This method gave a precision score of 95%.

CONCLUSION

After performing our mini project, the concept of text summarization in natural language processing has become very clear. There are several techniques available to perform a summarization but choosingthe perfect technique depends on the situation, context and type of text data that is to be summarized. In general, there two broad categories of summarization techniques that are employed nowadays namely, abstractive text summarization and extractive text summarization. In our project we implemented a general extractive summarization as well as an algorithm known as the text rank algorithm. After coding the above two methods we tested both programs on web articles as well as onscientific papers. When reading the summaries being generated, it could be observed that the text rank algorithm produced a summary that sounded more meaning, accurate and adhering to the original text when compared to the extractive text summarization program. In general, both the techniques can be improved on and modified to suit the purpose and type of text on which the summarization procedurehas to be performed.

REFERENCES

1. *https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f*

2. *https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/*

3. *https://medium.com/luisfredgs/automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25*

4. *https://www.machinelearningplus.com/nlp/text-summarization-approaches-nlp-example/*

5. *https://stackabuse.com/text-summarization-with-nltk-in-python/*

6. *https://www.upgrad.com/blog/text-summarization-in-nlp/*

7. *https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70*

8. *https://www.geeksforgeeks.org/python-text-summarizer/*

ACKNOWLEDGEMENT

We would like to thank our teacher, Dr. Phiroj Shaikh for his invaluable guidance throughout this project. Without his inputs this project would not have been a success. From teaching us all the stages involved in natural language processing like lexical analysis, morphological analysis, syntactic analysis, semantic analysis and pragmatics to actually implementing these concepts in a programming language, Dr. Phiroj Shaikh has been of tremendous help.

<u>Project Team Members :</u>

1. Reuben Crasto B. E. – (Roll No. 10)
2. Lance Dias B. E. – (Roll No. 16)
3. Salil Fernandes B. E. – (Roll No. 22)