

APPENDIX 1

NATURAL LANGUAGE PROCESSING USING SHORT TEXT SUMMARIZATION

PROJECT REPORT

Submitted by

Vaishnavi. S

Bhavani. P

Sangeetha. J

In partial fulfilment for the award of the degree

Of

BACHOLRE OF ENGINEERING

In

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MAM COLLEGE OF ENGINEERING

ANNA UNIVERSITY : CHENNAI 600 025

JANUARY 2023

APPENDIX 2

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**NATURAL LANGUAGE
PROCESSING USING SHORT TEXT SUMMARIZATION**”

is the bonafide work of “Vaishnavi. S ,Bhavani. P , Sangeetha. J ”

who carried out the project work under my supervision

Artificial intelligence and data science

Artificial Intelligence And Data Science

Trichy Chennai Trunk Road,

Trichy Chennai Trunk Road,

Siruganur, Tamil Nadu 621105

Siruganur, Tamil Nadu 621105

Contents

Abstract

Acknowledgments

1. Introduction

1.1 Domain

1.2 project intro

1.3 Problem Statement & Objectives

1.4 Hardware and software requirements

2. Literature Survey

2.1 Survey of Existing System/SRS

2.2 Architecture

2.3 Algorithm and Process Design

2.4 Existing system

2.5 Proposed System

2.6 How Text Summarization Works

2.7 Advantages Of Text Summarization

2.8 coding

3. Conclusion

3.1 Use cases

3.2 Benefits

3.3 Test Result

3.4 Performance Comparison Among
Datasets

3.5 Experiments and future results

3.6 End to End Applications

3.7 USE CASES

3.8 Conclusion and future work

Reference

ABSTRACT

Text Summarization is condensing the source text into a shorter version preserving its information content and overall meaning. It is very difficult for human beings to manually summarize large documents of text. Text Summarization methods can be classified into extractive and abstractive summarization. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. The importance of sentences is decided based on statistical and linguistic features of sentences. An abstractive summarization method consists of understanding the original text and re-telling it in fewer words. It uses linguistic methods to examine and interpret the text and then to find the new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original text document. This volume of text is an invaluable source of information and knowledge which needs to be effectively summarized to be useful. In this review, the main approaches to automatic text summarization are

described. We review the different processes for summarization and describe the effectiveness and shortcomings of the different methods. To perform abstractive summarization an encoder-decoder neural network with an attention model is used.

ACKNOWLEDGEMENT :

I extend my sincere thanks to MAM college of Engineering Which provided me with the opportunity to fulfil our wish and achieve our Goal. I would like to express deep debt to Prof. Makalakshmi project guide for Her vital suggestion, meticulous guidance and constant motivation, makes This project successful up till this point. They spent a lot of time with us and Gave all the related information and expertise about report writing

1. INTRODUCTION

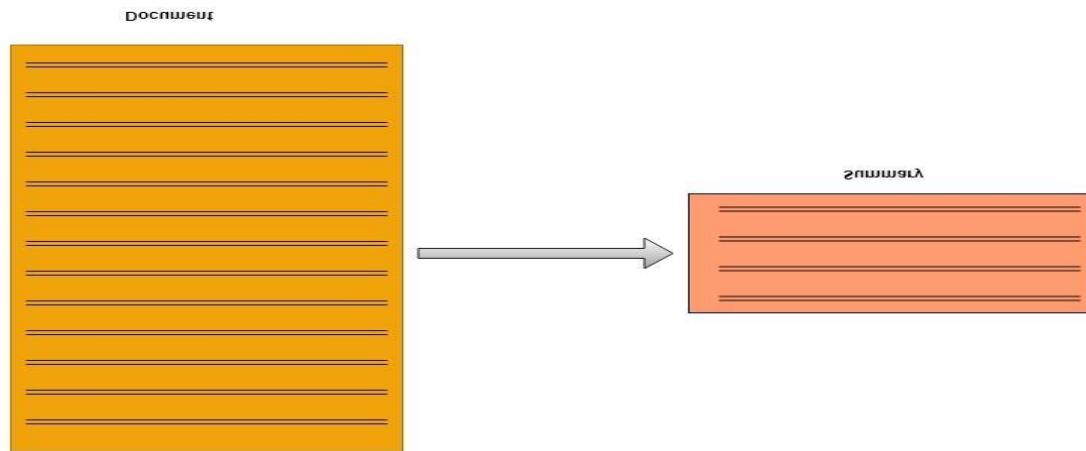
1.1 DOMAIN:

In the modern Internet age, textual data is ever increasing. Need some way to condense this data while preserving the information and meaning. We need to summarize textual data for that. Text summarization is the process of automatically generating natural language summaries from an input document while retaining the important points. It would help in easy and fast retrieval of information.

Stages of Text Summarization: -

- (i)Content Selection: - Choose Sentences to extract from large Chunks Text.
- (ii) Information Ordering: - Choose an order to place Summary.

(iii) Sentence Realization: - Clean up the sentences



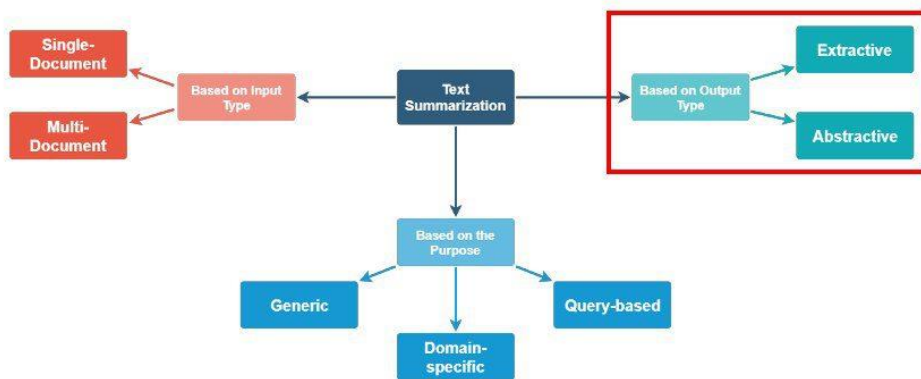
1.2 PROJECT INTRO:

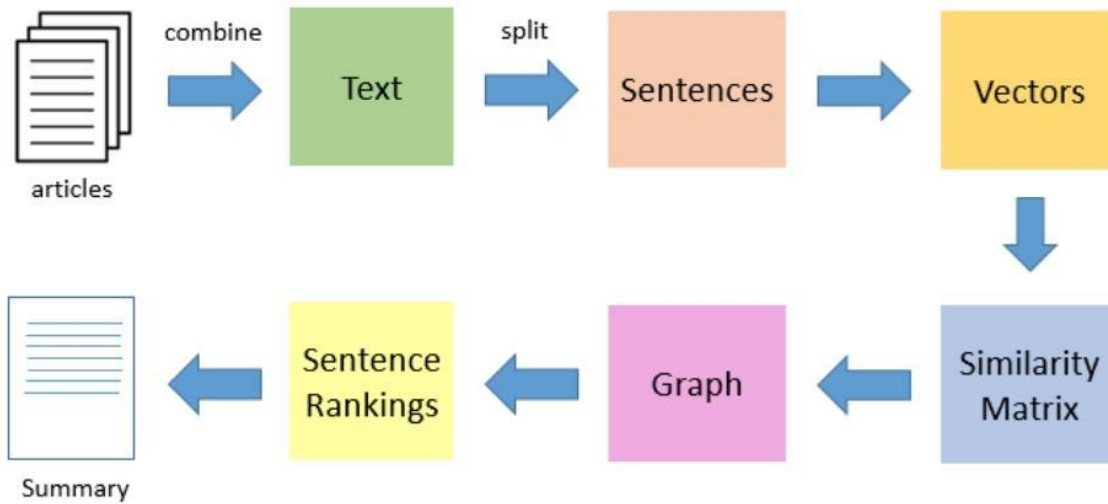
There are two prominent types of summarization algorithms.

Extractive summarization systems form summaries by copying parts of the source text through some measure of importance and then combine sentences together to render a summary. Importance of sentence is based on linguistic and statistical features.

Abstractive summarization systems generate new phrases, possibly rephrasing or using words that were not in the original text. Naturally abstractive approaches are harder. For perfect abstractive summary, the model should first truly understand the document and then try to express that understanding in short possibly using new words and phrases. Much harder than extractive. Has 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. Also, it promises grammatically correct and coherent summary. But they often do not summarize long and complex texts well as they are very restrictive.





1.3 PROBLEM STATEMENT & OBJECTIVES

To create a text summarizer which summarises the text or the content of the paragraph in minimum words without changing its meaning. This system is made using NLP based model which is branch of machine learning. This text summarizer also summarizes text from the web links and also summarises text from PDF document.

Objectives:

- Summaries reduce reading time.
- When researching documents, summaries make the selection process easier.
- Automatic summarization improves the effectiveness of indexing.

- Automatic summarization algorithms are less biased than human summarizers.
- Personalized summaries are useful in question-answering systems as they provide personalized information.
- Using automatic or semi-automatic summarization systems enables commercial abstract services to - increase the number of text documents they can process.

PROBLEM STATEMENT:

We have used NLP, which seeks to summarize articles by picking a collection of words that hold the most essential information, can address this problem with the help of extractive summarizer. This approach takes a significant portion of a phrase and utilizes it to create a summary. To define sentence verbs and subsequently rank them in terms of significance and similarity, a variety of algorithms and approaches are utilized. There is a great need for text summary techniques to address the amount of text data available online to help people find the right information and use the right information quickly. In addition, the implementation of text summaries reduces reading time, speeds up the

process of researching information, and increases the information that may not be in one field. This research paper focuses on the frequency-based approach for text summarization. The steps involved in text summarizer are Sentence and word tokenization and then calculating sentence score on the basis of TF-IDF score which is being used to select the most important sentences to retain the information and merge it to form a summary.

STEP-1: Import all necessary libraries

NLTK (Natural Language toolkit) is a widely used Library while we are working with text in python. Stop Words contain a list of English stop words, which need To be removed during the pre-processing step

STEP-2: Generate clean sentences

Text processing is the most important step in Achieving a constant and positive approach result. The Processing steps removes special digits, word, and Characters.

STEP-3: Calculate TF-IDF and generate a matrix

We'll find the TF and IDF for each word in a Paragraph.

$TF(t) = (\text{Frequency of } t \text{ from document}) / (\text{total no. Of } t \text{ in the document})$

$IDF(t) = \log_e (\text{total no. Of documents} / \text{No. Of Documents with } t \text{ it})$ Now, we will be generating a new matrix after Multiplying the calculated TF and IDF values.

STEP-4: Score the sentences

Here, we use TF-IDF word points in a sentence to give Weight to a paragraph. However, Sentence scoring varies with different algorithms.

STEP-5: Generate the summary

This is the last stage of text summarization. Top Sentences are calculated based on the score and Retention rate given to the user are included in the Summary and finally, a summary is created.

1.4 HARDWARE AND SOFTWARE REQUIREMENTS:

Software requirements	
Programming Language:	Python
Operating system:	Windows7, Windows 8 and high

	versions, Linux, Mac OS
--	-------------------------

Hardware Requirements
Laptop/PC with 4GB RAM
Processor-i3 or higher version, AMD 3 or higher version
Storage-5 GB max
Internet Connection
1 GHz speed

Hardware interfaces:

The solution involves extensive use of several hardware devices.

These devices include;

- Internet modem
- LAN

- Windows/Linux/Mac OS

Software interfaces:

We are using Django framework for creating this web application.

We are using python programming language for backend programming. Python uses NLTK library for natural language processing to solve the problem

2. Literature Survey

2.1 Survey of Existing System

We have investigated the existing surveys of the ATS domain, and a few of them are presented to prove the significance of this paper. Most surveys covered the former methods and research on ATS. However, recent trends, applicability,

effects, limitations, and challenges of ATS techniques were not present. Table 1 summarizes and compares the existing survey on ATS. Mishra reviewed (2000-2013) years of studies and found some methods such as hybrid statistical and ML approaches. The researchers did not include cognitive aspects or evaluations of the impact of ATS. Allahyari investigated different processes such as topic representation, frequency-driven, graph-based, and machine learning methods for ATS. This research only includes the frequently used strategies. El-Kassas described graph-based, fuzzy logic-based, concept oriented, ML approaches, etc., with their advantages or disadvantages. This research did not include abstractive or hybrid techniques. Saranyamol offered a thorough survey for analysts by introducing various aspects of ATS such as structure, strategies, datasets, evaluation metrics, etc. Gambhir attempted to analyse a hybrid approach including two text summarization methods. This study missed many contemporary techniques for review. The research of Gholamrezazadeh represents a comprehensive and comparative study of extractive methods in ATS of the last decade. Several multilingual approaches have also been discussed. Andhale provided a taxonomy of text summarization methods and a variety of techniques. Although the author has covered some time-consuming processes of ATS, recent, more efficient methods such as machine learning were missed. Abualigah conducted research on how to handle multiple documents and massive web data for text summarization. Lastly, the paper contains a comparative table with recent studies without details. Bharti

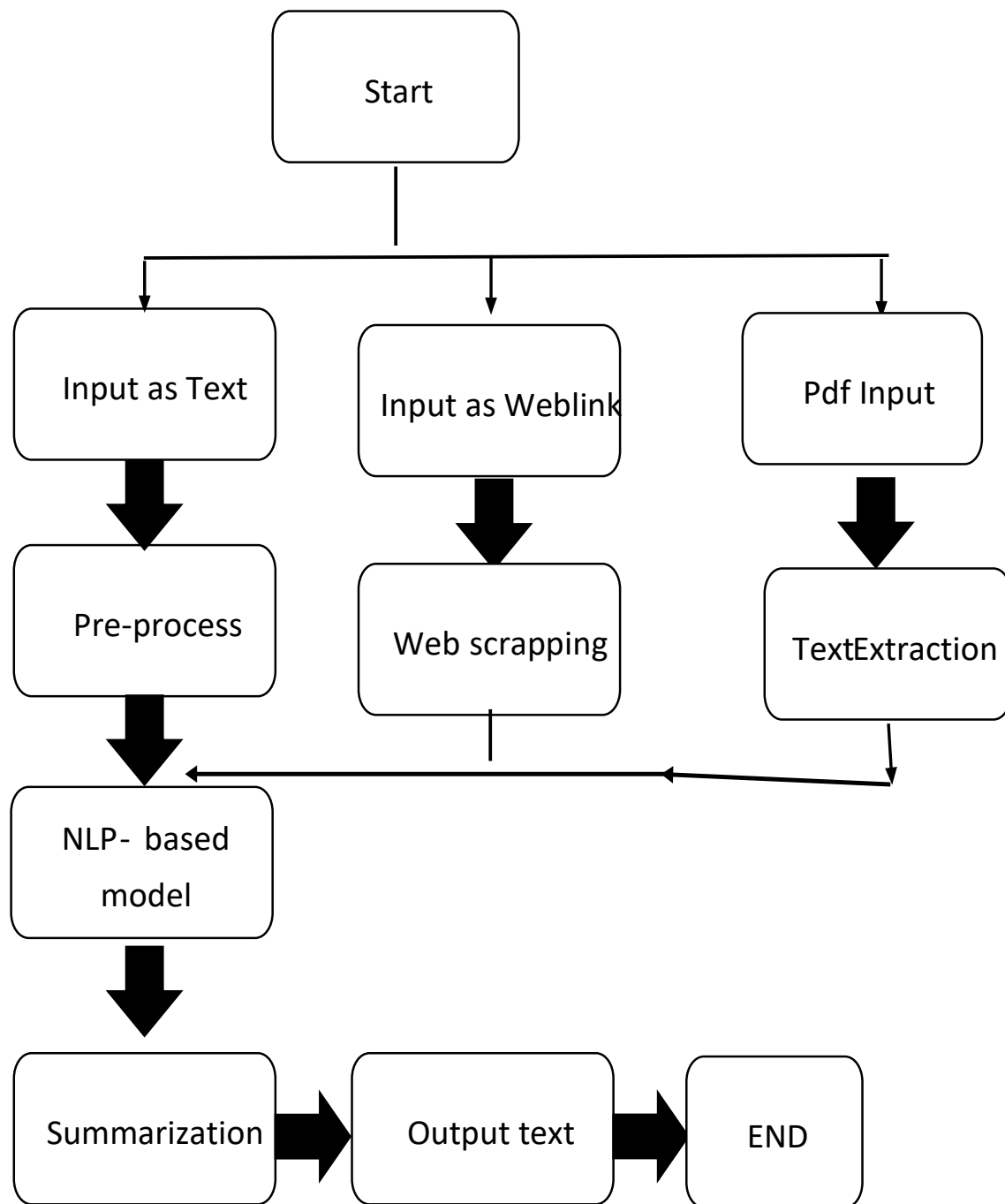
presented a survey of research papers based on automated keyword extraction methods and techniques. It covers ideas about multiple databases that are used for document summarization

1	2018, arXiv (Cornell University)	They Presented a fully data driven approach for automatic text summarization. They proposed and evaluated the model on standard datasets which show results comparable to the state of the art models without access to any linguistic information	they have assumed that summary length to be generated should be less than 'page_len'.
2	2017,arXiv (Cornell University)	They first pre-train the generative model by generating summaries given the source text. Then they pre-train the discriminator by providing positive examples from the human-generated summaries and the negative examples produced from the pre-trained generator. After the	The evaluation metric is different from the training loss. The input of the decoder in each time step is often from the true summary during the training. In the testing phase, the input of the next time step is the previous work.

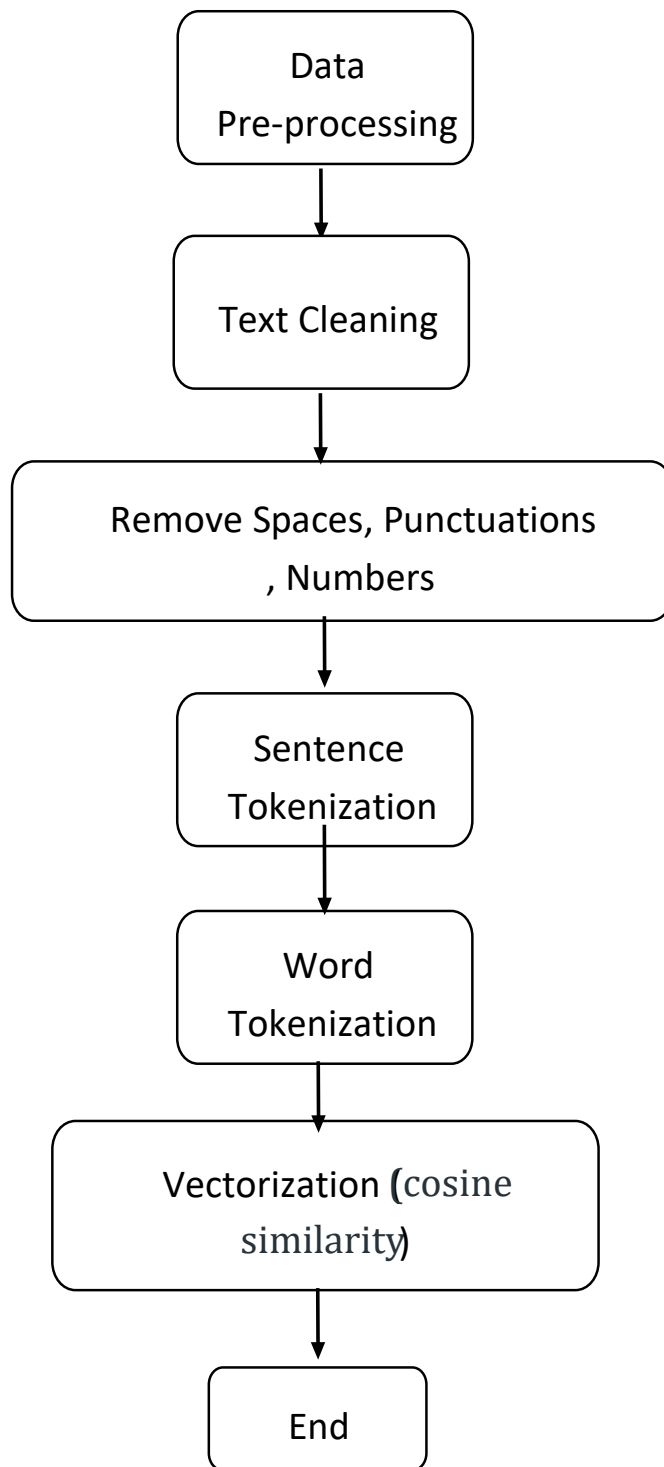
		pre-training, the generator and discriminator are trained alternatively.	
3	2019, arXiv (Cornell University)	They apply their model to CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains news articles (39 sentences on average) paired with multi-sentence summaries, and show that they outperform the state-of-the-art abstractive system by at least 2 ROUGE points.	It is unable to attain higher level of abstraction.
4	2020,JMLR (Journal of Machine Learning Research)	It reduces each document in the corpus to a vector of real numbers, each of which represents ratios of counts. After suitable normalization, this term frequency count is compared to an inverse document frequency count, which measures the number of occurrences of a word in the entire corpus .The end result is a	They could also consider partially exchangeable models in which they condition on exogenous variables; thus, for example, the topic distribution could be conditioned on features such as “paragraph” or “sentence,” providing a more powerful text model that makes use of information obtained from a

		<p>term-by-document matrix X whose columns contain the tf-idf values for each of the documents in the corpus.</p> <p>The tf-idf scheme reduces documents of arbitrary length to fixed length lists of numbers.</p>	parser.
--	--	---	---------

2.2 ARCHITECTURE:



MODEL WORKING:



2.3 Algorithm and Process Design

NLP:

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and written -- referred to as natural language. It is a component of artificial intelligence

TF-IDF (Term Frequency — Inverse Document Frequency)

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

TF-IDF is used with cosine similarity. TF-IDF (Term Frequency — Inverse Document Frequency) gives weights to individual words based on their uniqueness compared to the document's overall vocabulary. Words with higher weights (more unique) often have more importance or provide more meaning to the document.

TOKENIZATION:

Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph

To use this, we built a function that takes in an article's text, tokenizes each sentence

(data frame rows), creates a vocabulary without stop words for the individual document (data frame columns) and finally gives TF-IDF weights to each individual word in the vocab for each sentence.

COSINE SIMILARITY:

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

Similarity = $(A.B) / (||a||.||B||)$ where A and B are vectors.

Specifically, it measures the similarity in the direction or orientation of the vectors ignoring differences in their magnitude or scale. Both vectors need to be part of the same inner product space, meaning they must produce a scalar through inner product multiplication.

The similarity of two vectors is measured by the cosine of the angle between them. Using the TF-IDF weights for each sentence, I convert each row into a vector and store them in a matrix. Next, I find the cosine-similarity of each TF-IDF vectorized sentence pair

PROCESS FLOW :

Finally, after finding the cosine-similarity for all vectorized pairs, I average the weights of each vector, and return the indexes of the vectors with the highest averages. These indexes are then used to pull out the sentences from the original text for the summarization. The sentences with the highest average weights will capture the unique and important sentences from the original text (although like everything, it's not always perfect).

2.4 EXSISTING SYSTEMS

The text summarizations involve two approaches which are extractive and abstractive summarizations including summarization of single document and multi documents based on the requirements of extractive and abstractive summarizations. There are many methodologies and techniques used in the process of text summarization where the main objective is to reduce the size of the output by maintaining the coherence and accurate meaning of the original input.

2.5 PROPOSED SYSTEM

The objective is to automate the summarization of documents. The proposed system works on the extractive summarization-based approach. The system calculates the frequency weightage of each word in the sentence in the entire document and authenticates for the parts of speech of the words then allocates a total score for the sentences. The system then incorporates the clustering technique for extracting the final summary sentences.

The advantage of this approach is that in this, the Clustering phase the clusters are formed based upon the sentence scores and are segregated into lowest and highest weighted sentences from which the final phase provides the output based upon the highest scored clusters which give meaningful and efficient summaries. k-means clustering is an approach of quantization vector, initially from signal processing. Its objective is to divide and observe into k clusters where the cluster belongs to each observation with the closest mean cluster centroid or cluster centres. It is helping as a prototype of the cluster, which leads to the division of the data space into Voronoi cells. As a result. k-means clustering reduces intra-cluster differences, and the regular Euclidean distances would have the toughest Weber issues hence we squared Euclidean distances and the squared errors are optimized by mean.

2.6 How text summarization works:

In general there are two types of summarization, abstractive and extractive summarization.

ABSTRACTIVE SUMMARIZATION :

Abstractive methods select words based on semantic understanding, even those words did not appear in the source documents. It aims at producing important material in a new way. They interpret and examine the text using advanced natural language techniques in order to generate a new shorter text that conveys the most critical information from the original text.

It can be correlated to the way human reads a text article or blog post and then summarizes in their own word.

Input document → understand context → semantics → create own summary.

Extractive Summarization:

Extractive methods attempt to summarize articles by selecting a subset of words that retain the most important points. This approach weights the important part of sentences and uses the same to form the summary. Different algorithm and techniques are used to define weights for the sentences and further rank them based on importance and similarity among each other.

Input document → sentences similarity → weight sentences → select sentences with higher rank.

The limited study is available for abstractive summarization as it requires a deeper understanding of the text as compared to the extractive approach.

Purely extractive summaries often times give better results compared to automatic abstractive summaries. This is because of the fact that abstractive summarization methods cope with problems such as semantic representation,

Inference and natural language generation which is relatively harder than data-driven approaches such as sentence extraction.

There are many techniques available to generate extractive summarization. To keep it simple, I will be using an unsupervised learning approach to find the sentences similarity and rank them. One benefit of this will be, you don't need to train and build a model prior start using it for your project.

(a) Extractive Summarization

Source Text: Tom and Jerry went by bicycle to listen to a lecture on campus. While in class, Tom got a call and headed back home.

Summary: Tom and Jerry listen lecture campus. Tom headed home.

(a) Abstractive Summarization

Source Text: Tom and Jerry went by bicycle to listen to a lecture on campus. While in class, Tom got a call and headed back home.

Summary: Tom headed home after listening to a lecture with Jerry.

ACCERN

2.7 Advantages of Text Summarization

Instantly effective

It takes time and effort to read the entire article, deconstruct it, and separate the significant concepts from the raw text. It takes at least 15 minutes to read a 500-word article.

In a fraction of a second, automatic summary software summarises texts of 500-5000 words. This enables the user to read less data while still getting the most critical information and drawing sound judgments.

It Functions in Any Language

Many summarization software can work in any language, which is a capability that most humans lack. Because summarizers are based on linguistic models, they can automatically summarise texts in a wide range of languages, from English to Russian. As a result, they're great for persons who read and work with multilingual information.

Productivity is increased

Not only do some software summarise documents, but they also summarise web pages. This boosts productivity by accelerating the surfing process. Instead of reading entire news stories that are full of irrelevant information, summaries of such websites can be detailed and accurate while yet being only 20% of the original article's size.

2.8 CODING:

Next, Below is our code flow to generate summarize text:-

Input article → split into sentences → remove stop words → build a similarity matrix → generate rank based on matrix → pick top N sentences for summary.

Let's create these methods.

1.Import all necessary libraries:

```
From nltk.corpus import stopwords
```

```
From nltk.cluster.util import cosine_distance
```

```
Import numpy as np
```

Import networkx as nx

2.Generate clean sentences:

```
Def read_article(file_name):  
    File = open(file_name, "r")  
    Filedata = file.readlines()  
    Article = filedata[0].split(". ")  
    Sentences = []  
    For sentence in article:  
        Print(sentence)  
        Sentences.append(sentence.replace("[^a-zA-Z]", "").split(" "))  
    Sentences.pop()  
    Return sentences
```

3. Similarity matrix

This is where we will be using cosine similarity to find similarity between sentences.

```
Def build_similarity_matrix(sentences, stop_words):  
    # Create an empty similarity matrix  
    Similarity matrix = np.zeros((len(sentences), len(sentences)))  
  
    For idx1 in range(len(sentences)):  
        For idx2 in range(len(sentences)):  
            If idx1 == idx2: #ignore if both are same sentences  
                Continue
```

```

        Similarity_matrix[idx1][idx2] =
sentence_similarity(sentences[idx1], sentences[idx2], stop_words)

Return similarity_matrix

```

4. Generate Summary Method

Method will keep calling all other helper function to keep our summarization pipeline going. Make sure to take a look at all # Steps in below code.

```

Def generate_summary(file_name, top_n=5):

    Stop_words = stopwords.words('english')

    Summarize_text = []

    # Step 1 – Read text and tokenize

    Sentences = read_article(file_name)

    # Step 2 – Generate Similarity Martix across sentences

    Sentence_similarity Martix = build_similarity_matrix(sentences,
stop_words)

    # Step 3 – Rank sentences in similarity martix

    Sentence_similarity_graph =
nx.from_numpy_array(sentence_similarity Martix)

    Scores = nx.pagerank(Sentence_similarity_graph)

    # Step 4 – Sort the rank and pick top sentences

    Ranked_sentence = sorted(((scores[i],s) for I,s in
enumerate(sentences)), reverse=True)

    Print("Indexes of top ranked_sentence order are “, ranked_sentence)

    For I in range(top_n):

        Summarize_text. Append(“ “.join(ranked_sentence[i][1]))

    # Step 5 – Offcourse, output the summarize texr

```

```
Print("Summarize Text: \n", “. “.join(summarize_text))
```

All put together, here is the complete code:

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
From nltk.corpus import stopwords
```

```
From nltk.cluster.util import cosine_distance
```

```
Import numpy as np
```

```
Import networkx as nx
```

```
Def read_article(file_name):
```

```
    File = open(file_name, "r")
```

```
    Filedata = file.readlines()
```

```
    Article = filedata[0].split(" ")
```

```
    Sentences = []
```

```
    For sentence in article:
```

```
        Print(sentence)
```

```
        Sentences.append(sentence.replace("[^a-zA-Z]", " ").split(" "))
```

```
    Sentences.pop()
```

```
    Return sentences
```



```
Def sentence_similarity(sent1, sent2, stopwords=None):
```

```
    If stopwords is None:
```

```
        Stopwords = []
```

```
    Sent1 = [w.lower() for w in sent1]
```

```
    Sent2 = [w.lower() for w in sent2]
```

```
    All_words = list(set(sent1 + sent2))
```

```
    Vector1 = [0] * len(all_words)
```

```
    Vector2 = [0] * len(all_words)
```

```
    # build the vector for the first sentence
```

```
    For w in sent1:
```

```
        If w in stopwords:
```

```
            Continue
```

```
        Vector1[all_words.index(w)] += 1
```

```
    # build the vector for the second sentence
```

```
    For w in sent2:
```

```
        If w in stopwords:
```

```
            Continue
```

```
        Vector2[all_words.index(w)] += 1
```

```
Return 1 – cosine_distance(vector1, vector2)
```

```
Def build_similarity_matrix(sentences, stop_words):
```

```
    # Create an empty similarity matrix
```

```
    Similarity_matrix = np.zeros((len(sentences), len(sentences)))
```

```
    For idx1 in range(len(sentences)):
```

```
        For idx2 in range(len(sentences)):
```

```
            If idx1 == idx2: #ignore if both are same sentences
```

```
                Continue
```

```
                Similarity_matrix[idx1][idx2] =  
sentence_similarity(sentences[idx1], sentences[idx2], stop_words)
```

```
    Return similarity_matrix
```

```
Def generate_summary(file_name, top_n=5):
```

```
    Stop_words = stopwords.words('english')
```

```
    Summarize_text = []
```

```
    # Step 1 – Read text and split it
```

```
    Sentences = read_article(file_name)
```

```
    # Step 2 – Generate Similarity Matrix across sentences
```

```
Sentence_similarity_martix = build_similarity_matrix(sentences,
stop_words)
```

```
# Step 3 – Rank sentences in similarity martix
```

```
Sentence_similarity_graph =
nx.from_numpy_array(sentence_similarity_martix)

Scores = nx.pagerank(Sentence_similarity_graph)
```

```
# Step 4 – Sort the rank and pick top sentences
```

```
Ranked_sentence = sorted(((scores[i],s) for I,s in
enumerate(sentences))), reverse=True)
```

```
Print("Indexes of top ranked_sentence order are ", ranked_sentence)
```

```
For I in range(top_n):
```

```
Summarize_text.append(" ".join(ranked_sentence[i][1]))
```

```
# Step 5 – Offcourse, output the summarize text
```

```
Print("Summarize Text: \n", " ".join(summarize_text))
```

```
# let's begin
```

```
Generate_summary( "msft.txt", 2)
```

Let's look at it in action.

The complete text from an article titled Microsoft Launches Intelligent Cloud Hub To Upskill Students In AI & Cloud Technologies

In an attempt to build an AI-ready workforce, Microsoft announced Intelligent Cloud Hub which has been launched to empower the next generation of students with AI-ready skills. Envisioned as a three-year collaborative program, Intelligent Cloud Hub will support around 100 institutions with AI infrastructure, course content and curriculum, developer support, development tools and give students access to cloud and AI services. As part of the program, the Redmond giant which wants to expand its reach and is planning to build a strong developer ecosystem in India with the program will set up the core AI infrastructure and IOT Hub for the selected campuses. The company will provide AI development tools and Azure AI services such as Microsoft Cognitive Services, Bot Services and Azure Machine Learning. According to Manish Prakash, Country General Manager-PS, Health and Education, Microsoft India, said, “With AI being the defining technology of our time, it is transforming lives and industry and the jobs of tomorrow will require a different skillset. This will require more collaborations and training and working with AI. That’s why it has become more critical than ever for educational institutions to integrate new cloud and AI technologies. The program is an attempt to ramp up the institutional set-up and build capabilities among the educators to educate the workforce of tomorrow.” The program aims to build up the cognitive skills and in-depth understanding of developing intelligent cloud connected solutions for applications across industry. Earlier in April this year, the company announced Microsoft Professional Program In AI as a learning track open to the public. The program was developed to provide job ready skills to programmers who wanted to hone their skills in AI and data science with a series of online courses which featured hands-on labs and expert instructors as well. This program also included developer-focused AI school that provided a bunch of assets to help build AI skills.

And the summarized text with 2 lines as an input is

Envisioned as a three-year collaborative program, Intelligent Cloud Hub will support around 100 institutions with AI infrastructure, course content and curriculum, developer support, development tools and

give students access to cloud and AI services. The company will provide AI development tools and Azure AI services such as Microsoft Cognitive Services, Bot Services and Azure Machine Learning. According to Manish Prakash, Country General Manager-PS, Health and Education, Microsoft India, said, “With AI being the defining technology of our time, it is transforming lives and industry and the jobs of tomorrow will require a different skillset.

5 techniques for text summarization in Python

Here are five approaches to text summarization using both abstractive and extractive methods.

1. Gensim

Gensim is an open-source topic and vector space modeling toolkit within Python programming language.

First, the user needs to utilize the `summarization.summarizer` from Gensim as

is based on a variation of the TextRank algorithm.

Since TextRank is a graph-based ranking algorithm, it helps narrow down importance of vertices in graphs based on global information drawn from said graphs.

Here's an example code to summarize text from Wikipedia:

```
From gensim.summarization.summarizer import summarize
From gensim.summarization import keywords
Import wikipedia
Import en_core_web_sm
```

To import the wikipedia content:

```
Wikisearch = wikipedia.page("")
Wikicontent = wikisearch.content
Nlp = en_core_web_sm.load()
Doc = nlp(wikicontent)
```

To summarize based on percentage:

```
Summ_per = summarize(wikicontent, ratio = "")
Print("Percent summary")
Print(summ_per)
```

To summarize based on word count:

```
Summ_words = summarize(wikicontent, word_count = "")
Print("Word count summary")
Print(summ_words)
```

There are two ways of extracting text using TextRank: keyword and sentence extraction.

Keyword extraction can be done by simply using a frequency test, but this would almost always prove to be inaccurate. This is where TextRank automates the process to semantically provide far more accurate results based on the corpus.

Sentence extraction, on the other hand, studies the corpus to summarize the most valid sentences pertaining to the subject matter and phonologically arranges it.

2. Sumy

Sumy is another library in Python that uses various algorithms to perform text summarization.

Let's take a look at a few of them.

LexRank

LexRank is a graphical-based summarizer. The code is as follows:

```
From sumy.summarizers.lex_rank import LexRankSummarizer
Summarizer_lex = LexRankSummarizer()
```

```
# Summarize using sumy LexRank
Summary= summarizer_lex(parser.document, 2)
Lex_summary="""
For sentence in summary:
Lex_summary+=str(sentence)
Print(lex_summary)
```

```
Print(text_summary)
```

Luhn

Developed by an IBM researcher of the same name, Luhn is one of the oldest summarization algorithms and ranks sentences based on a frequency criterion for words.

Here's the code for the algorithm:

```
From sumy.summarizers.luhn import LuhnSummarizer
Summarizer_1 = LuhnSummarizer()
Summary_1 =summarizer_1(parser.document,2)
```

```
For sentence in summary_1:
Print(sentence)
```

LSA

Latent semantic analysis is an automated method of summarization that utilizes term frequency with singular value decomposition. It has become one of the most used summarizers in recent years.

The code is as follows:

```
From sumy.summarizers.lsa import LsaSummarizer
Summarizer_lsa = LsaSummarizer()
# Summarize using sumy LSA
Summary =summarizer_lsa(parser.document,2)
Lsa_summary="""
For sentence in summary:
Lsa_summary+=str(sentence)
Print(lsa_summary)
```

TextRank

And last but not least, there is TextRank which works exactly the same as in Gensim.

Here's the code for this algorithm:

```
# Load Packages
From sumy.parsers.plaintext import PlaintextParser
From sumy.nlp.tokenizers import Tokenizer

# For Strings
```



```
Parser = PlaintextParser.from_string(text,Tokenizer("english"))
From sumy.summarizers.text_rank import TextRankSummarizer
```

```
# Summarize using sumy TextRank
Summarizer = TextRankSummarizer()
Summary =summarizer_4(parser.document,2)
Text_summary="""
```

```
For sentence in summary:
Text_summary+=str(sentence)
Print(text_summary)
```

When using each of these summarizers, you will notice that they summarize text differently. It's better to try them all to figure out which one works best in different situations.

3. NLTK

The 'Natural Language Toolkit' is an NLP-based toolkit in Python that helps with text summarization.

Here's how to get it up and running.
Import the required libraries using the code below:

```
Import nltk
From nltk.corpus import stopwords
From nltk.tokenize import word_tokenize, sent_tokenize
```

Input your text for summarizing below:

```
Text = """ """
```

Next, you need to tokenize the text:

```
stopWords = set(stopwords.words("english"))
words = word_tokenize(text)
```

Now, you will need to create a frequency table to keep a score of each word:

```
freqTable = dict()
for word in words:
    word = word.lower()
    if word in stopWords:
        continue
    if word in freqTable:
        freqTable[word] += 1
    else:
        freqTable[word] = 1
```

Next, create a dictionary to keep the score of each sentence:

```
Sentences = sent_tokenize(text)
sentenceValue = dict()

for sentence in sentences:
    for word, freq in freqTable.items():
        if word in sentence.lower():
            if sentence in sentenceValue:
                sentenceValue[sentence] += freq
            else:
                sentenceValue[sentence] = freq

sumValues = 0
for sentence in sentenceValue:
    sumValues += sentenceValue[sentence]
```

Now, we define the average value from the original text as such:

```
Average = int(sumValues / len(sentenceValue))
```

And lastly, we need to store the sentences into our summary:

```

Summary = ''
For sentence in sentences:
If (sentence in sentenceValue) and (sentenceValue[sentence] > (1.2 * average)):
Summary += " " + sentence
Print(summary)

```

4. T5

To make use of Google's T5 summarizer, there are a few prerequisites.

First, you will need to install PyTorch and Hugging Face's Transformers. You can install the transformers using the code below:

Pip install transformers

Next, import PyTorch along with the AutoTokenizer and AutoModelWithLMHead objects:

Import torch

From transformers, import AutoTokenizer, AutoModelWithLMHead

Next, you need to initialize the tokenizer model:

```

Tokenizer = AutoTokenizer.from_pretrained('t5-base')
Model = AutoModelWithLMHead.from_pretrained('t5-base', return_dict=True)

```

From here, you can use any data you like to summarize. Once you have gathered your data, input the code below to tokenize it:

```

Inputs = tokenizer.encode("summarize: " + text,
Return_tensors='pt',
Max_length=512,
Truncation=True)

```

Now, you can generate the summary by using the model.generate function on T5:

```
Summary_ids = model.generate(inputs, max_length=150, min_length=80,  
                             length_penalty=5., num_beams=2)
```

Feel free to replace the values mentioned above with your desired values. Once it's ready, you can move on to decode the tokenized summary using the `tokenizer.decode` function:

```
Summary = tokenizer.decode(summary_ids[0])
```

And there you have it: a text summarizer with Google's T5. You can replace the texts and values at any time to summarize various arrays of data.

5. GPT-3

GPT-3 is a successor to the GPT-2 API and is much more capable and functional. Let's take a look at how to get it running on Python with an example of downloading PDF research papers.

First, you will need to import all dependencies as listed below:

```
Import openai  
Import wget  
Import pathlib  
Import pdfplumber  
Import numpy as np
```

You will then need to install `openai` to interact with GPT-3, so make sure you have an API key. You can get one [here](#).

You will also need `wget` to download PDFs from the internet. This will further require `pdfplumber` to convert it back to text. Install all three with `pip`:

```
Pip install openai  
Pip install wget  
Pip install pdfplumber
```


ACCERN

Input



Extractive
Summarization

Text Preprocessing

Feature Extraction

Sentence Scoring

Extractive
Summarization Method

1. Frequency
2. Sentence Position
3. Cue words
4. Sentence length.
5. Important entitles(NER)
6. Tfidf scores

Input



Abstractive
Summarization

Text Preprocessing

Sequence to
Sequence Model

Bahdanau
attention Result

Abstractive
Summarization Method

```
{ "Summary":  
  { "Abstractive":  
    "Rephrasing the  
    sentence to get  
    the gist"  
  }  
  { "Extractive":  
    { "text": [ "sentence",  
               "sentence 2" ], "score":  
      [ 2, 1 ]  
    }  
  }  
}
```

3.CONCLUSION

3.1 USE CASES

Financial Research with NLP:

Financial and investment decisions require an in-depth investigation and classification of a significant quantity of information. This is true for both individual investors and investment firms. This is where an automatic text summarization designed for evaluating and condensing financial information can come in handy.

People who are unfamiliar with extracting data from financial statistics or reports, for example, can use automated text summarization for providing a concise summary of those financial reports.

Watch this 3-min video to see how asset managers can leverage the Accern NoCodeNLP Platform to receive fast and accurate insights from text data to fuel timely investment decisions.

Media Monitoring with NLP

Assume you need to learn about the current state of an industry from a variety of publications and media. But you hardly have time to scan all the headlines, let alone read all of them and get to the meat of their arguments.

In this circumstance, text summarization can help you scan more information by extracting the summary of numerous news articles and other media

3.2 BENEFITS :

1 . Scalable and Quick

Manually summarizing a short document is fairly easy, but what if you have an article or paper that is hundreds or thousands of pages long?

Rather than having dozens of employees manually going through thousands of documents, you can automate this analysis with the Accern NoCodeNLP Platform. Our software will analyze all your input text and source documents and provide you with a summary text.

2.Leverages Existing Tools

To take advantage of text summarization, you don't have to build machine learning infrastructure in-house or hire a data science team. The technology exists and it's accessible to everyone.

Most CEOs or directors may get confused just hearing the buzzwords: artificial intelligence, machine learning models, NLP tasks, text rank algorithm, etc.

But there's absolutely no reason for feeling discouraged when No-Code NLP platforms are available.

The summarization of documents and transformation of data, words, and sentences into decisions is possible and already used in a variety of industries with AI / ML / NLP platforms like ours.

My point here is, you do not have to fully understand how text summarization works or be an expert. We can supply you with the technology that is able to

effortlessly extract key points from a given document and provide a detailed summary using NLP.

Our text summarization algorithms are easy to use and available to make your research and business decision-making process more efficient and actionable.

3.Understand Your Customers Better

You can never go wrong with having a better understanding of your customers.

The data on your customers may come in many forms, such as spreadsheets, chat messages, emails, etc. And of course it might come in different languages too.

In customer relationship management, text summarization refers to compressing all the above-mentioned customer data, and turning it into abridged summaries that you can present in a meeting or use in other business processes.

Since NLP can extract insight from text data, this makes it a perfect tool for keeping track of customer feedback, determining sentiment, whether it's positive or negative, and to what degree.

This allows organizations to monitor reviews in real-time and flag the most important or time-sensitive comments, provide timely feedback, and ignore irrelevant information.

4.Summarize the Text in Different Formats

Whether you are conducting research or launching a new product, one of the first steps is analyzing your market and competitors.

How do you do that given the volume and velocity of information and remain productive?

Natural language processing helps you obtain summarized text extracted from your competitor's web pages, market research documents, industry-related articles, etc. Having a clear idea of the market and your competitors helps you determine actionable steps for presenting your product or refining your business strategy. This helps you stand out amongst the competitors and maintain a competitive advantage in the market.

The Accern NLP platform can provide you with the most relevant sentences that you can use to communicate your product, important points to focus on and give you a deep understanding of your environment.

People who are not familiar with extracting data from financial statistics or reports, for example, can use automated text summarization for capturing a synopsis of those statistics and reports

. Ensure all Critical Information is Covered

The human eye can overlook important details, while standalone software is more accurate. What every reader needs is the ability to extract what is useful to them from any piece of material.

The automated text summarizing approach makes it easy for the user to read all the most important sentences in a document.

5 . Run Sentiment Analysis

Sentiment analysis uses artificial intelligence and natural language processing to identify, extract, and analyze textual data to understand the overall attitude and emotional tone of a text.

Once the data is evaluated, a sentiment score will be generated to determine whether the data is positive, negative, or neutral and to what extent.

Sentiment analysis is used across many industries, including by hedge funds for analysing financial news to predict stock market trends and movements. It is also used by traders and investment bankers to research, extract data on and analyze ESG compliance and mergers and acquisitions news.

With sentiment analysis (downloadable white paper), financial teams can even evaluate consumer sentiments around specific companies.

3.3 TEST RESULT

Short Input

While performing the testing for Smaller inputs we get an error of minimum Summary value Where it denotes about the word's frequency is not Greater than required frequency to calculate the Summary.

Foreign Language

While giving input in any Language, it successfully performs the Summarization process and a meaningful summary Is obtained.

Improper URL

If the given URL hasn't a defined URL And a sequential data which can be summarize then It displays the error as mentioned below since the Web scrapper can't get the exact data from the URL From which our summary could be generated.

Illogical Text

If any illogical or meaningful text Is given as an input, then the summary won't come As it will not make sense to generate a summary of Punctuation marks or any stop words. As given Below the output is generated where it shows that The

given text could be stop words which gets Eliminated in the pre-processing phase of Summarization.

Input Article (Description)	Original Summary (Synopsis)	Generated Summary (Synopsis)
hi support please see alarm root mgojunre show chassis alarms noforwarding alarms currently active alarm time class description pht major fpc major errors error code attached rsi ... remained clean closing technical case per support needed customer contact agreed close marione jgramirez	FPC 5 Major Error - Error code 65537	fpc major errors
high cpu utilization observed one routers cpu utilization user percent eopqqtu hrstwi show system processes extensive ... hyperlink idkb customer contact pushkar sanchit tarun	rma details need to refresh the rma number of table view	[RMA Details] In Task details page, the notes API is not being called on clicking "Refresh" button
switch chassis member issue integrating switches issue description exc continuously creates ... current network status resolved resolution detail pr customer contact elad	they have 3 switch 2 chassis is not member have issue integrating the switches	switch not functioning
hello one power supplies failed mx router beginning saw bunch warnings like pemaltiusgettemp ... archive case issue description issue description pem faulty current network status stable resolution detail processed rma rma delivered customer contact sergey yemelyanovich	Power Supply failed	ex power supply down
routing engines visible forwarding table issue description routes seen forwarding table routinginstance current network status issue resolved configuration changes resolution detail instance type specified routinginstance caused forwarding table show routes customer contact jacek	routing engines are not visible in the forwarding table	routing engine issue

3.4 Performance Comparison Among Datasets :

Figure 22 shows the comparison among the model's performances on different datasets in Terms of ROUGE-1 and BLEU scores. Figure 22 shows that our model has the best performance

On the JIRA Dataset. The two metrics ROUGE-comparing and BLEU scores are computed by comparing

The generated summary with the actual summary word by word. However, abstractive

Summarization is not about creating the same summary as the actual summary, because two

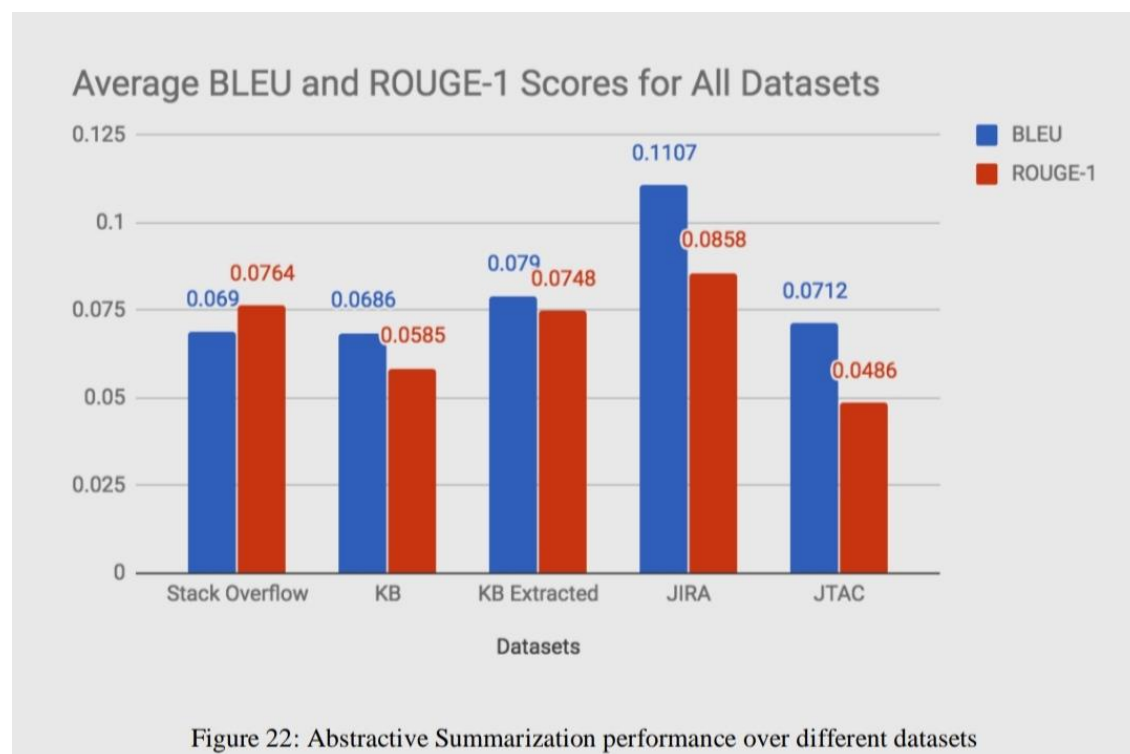
Summaries can be different when compared word by word while still summarizing the article

Properly. So, while the two metrics give some context about the generated summary, it is

Important to manually look at some of the generated summaries and not base our evaluation of

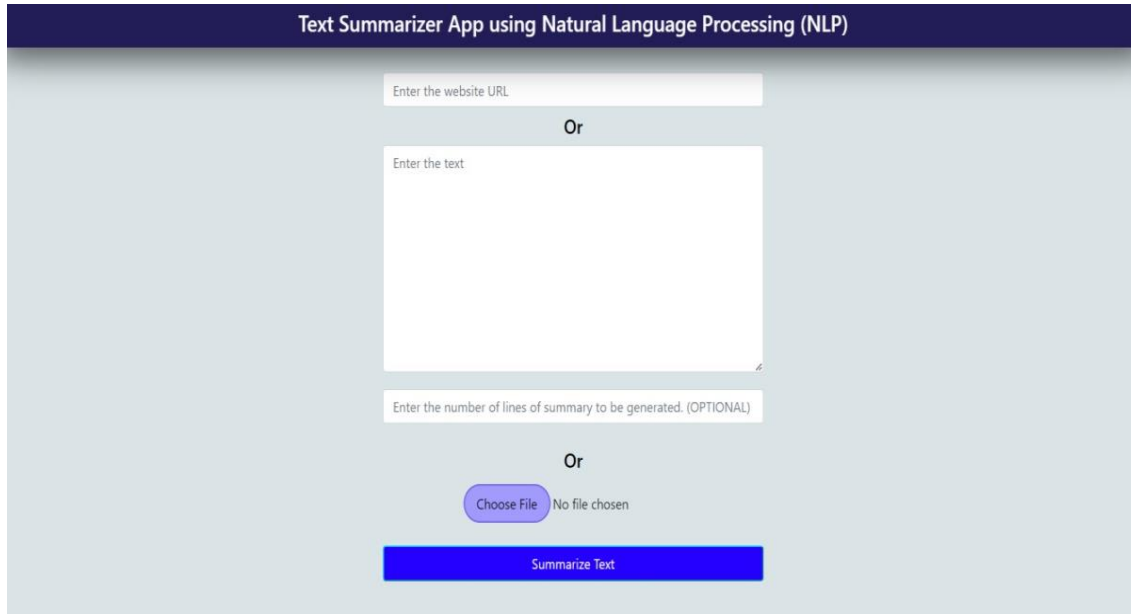
Due to our limited resources, the small size and noisy datasets, our scores do not reflect

Great performance. On an average, with our resources, a model training on 15,000 articles over



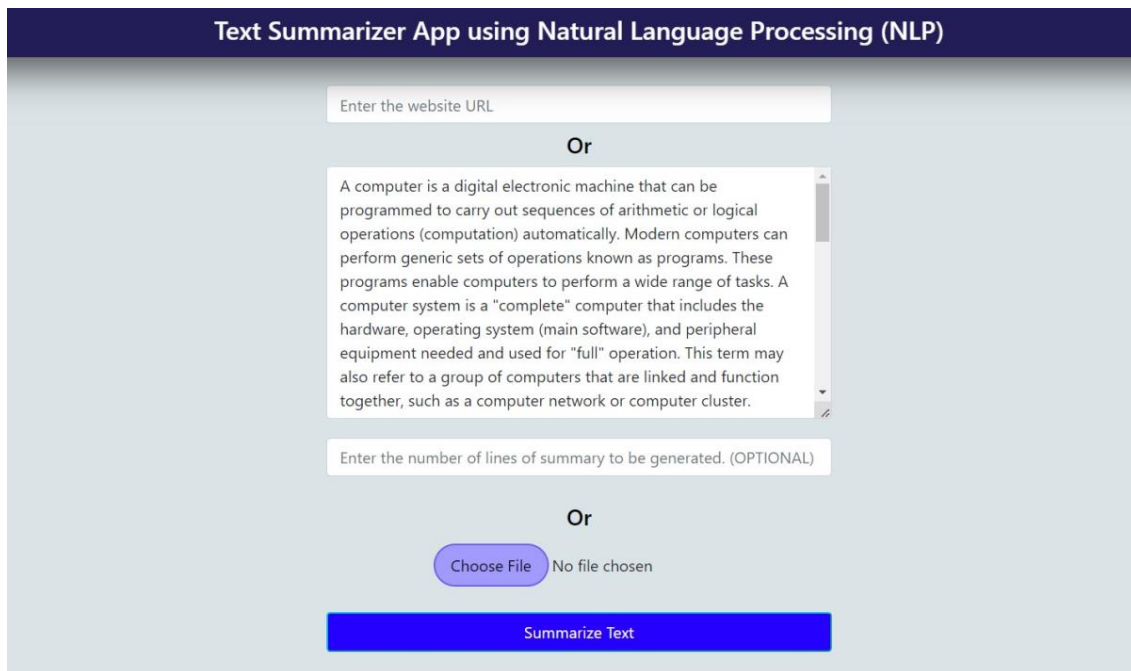
3.5 Experiment and Results for Validation

GUI of project:

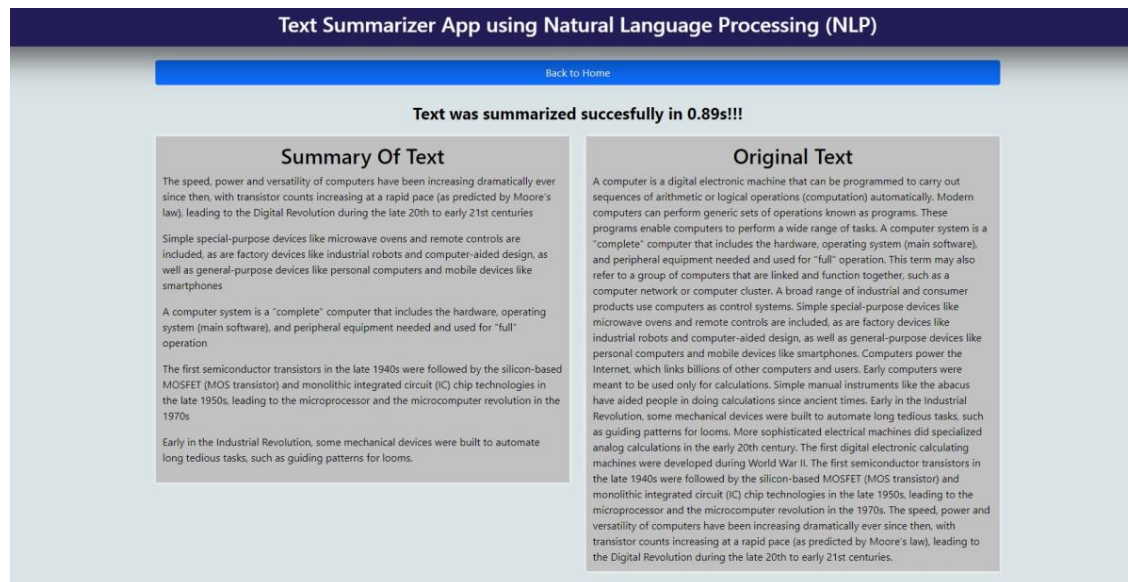


The screenshot shows the 'Text Summarizer App using Natural Language Processing (NLP)' interface. It features a dark blue header with the app's name. Below the header, there are three input options separated by 'Or' labels. The first option is a text box labeled 'Enter the website URL'. The second option is a larger text area labeled 'Enter the text'. The third option is a text box labeled 'Enter the number of lines of summary to be generated. (OPTIONAL)'. Below these, there is a 'Choose File' button and a 'No file chosen' label. At the bottom, there is a large blue button labeled 'Summarize Text'.

INPUT AS TEXT :



This screenshot shows the same 'Text Summarizer App' interface, but with text entered into the 'Enter the text' field. The text is a paragraph about computers: 'A computer is a digital electronic machine that can be programmed to carry out sequences of arithmetic or logical operations (computation) automatically. Modern computers can perform generic sets of operations known as programs. These programs enable computers to perform a wide range of tasks. A computer system is a "complete" computer that includes the hardware, operating system (main software), and peripheral equipment needed and used for "full" operation. This term may also refer to a group of computers that are linked and function together, such as a computer network or computer cluster.' The 'Summarize Text' button is still visible at the bottom.



SUMMARIZING APP:

<https://play.google.com/store/apps/details?id=com.aya.textsummarizer>

3.6 End-to-End Application

Finally, we developed an end-to-end web application to demonstrate the concept of text summarization . The web application was hosted on a Microsoft Azure web server. Once logged into the web app, one could enter an input text, select the appropriate model they want the text to run on and get back the summary. The once the user clicks the summarize button, the HTML page makes a POST request to the backend server which has preloaded all the appropriate models. After the request is received, the server runs the input against the model, fetches the result and sends it as the response to the web client. The web page is then updated to reflect the output. Developing an end-to-end application helped demonstrate the results and the capabilities of text summarization efficiently. The backend server of the tool could be used by the Chatbot to fetch real-time summaries once the model is accurate enough

3.7 Use Case

Financial Research with NLP

Financial and investment decisions require an in-depth investigation and classification of a significant quantity of information. This is true for both individual investors and investment firms. This is where an automatic text summarization designed for evaluating and condensing financial information can come in handy.

People who are unfamiliar with extracting data from financial statistics or reports, for example, can use automated text summarization for providing a concise summary of those financial reports.

Watch this 3-min video to see how asset managers can leverage the Accern NoCodeNLP Platform to receive fast and accurate insights from text data to fuel timely investment decisions.

Media Monitoring with NLP

Assume you need to learn about the current state of an industry from a variety of publications and media. But you hardly have time to scan all the headlines, let alone read all of them and get to the meat of their arguments.

In this circumstance, text summarization can help you scan more information by extracting the summary of numerous news articles and other media.

3.8 CONCLUSION AND FUTURE WORK

Text summarization is an interesting machine learning field that is increasingly gaining attraction. As research in this area continues, we can expect to see breakthroughs that will assist in fluently and accurately shortening long text documents.

Hereby We can say we have successfully completed text summarization using NLP as per problem statement with efficiency. By this project we have solved the problem by the summaries of the text to gain information. We have tried our best to make these summaries as important as possible in the aspect of text intention We can add various features to our web applications like we can take input of almost any text format like (.doc and .docx,.rtf) by uploading it directly in our input box for text summarization We can also integrate features like the voice text acceptance for the text summarization. Example, someone reads out loud the text paragraph from the newspaper or passage from novel which is difficult to understand and needs to be summarized. We have certain limitation while dealing with punctuation marks and spaces so in future we will try to make it as proper as possible.

REFERENCE

Journals / Conference Papers:

- [1] Sinha, Aakash , Abhishek Yadav, and Akshay Gahlot.
"Extractive text summarization using neural networks." arXiv preprint arXiv:1802.10137 (2018).
- [2] Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." *arXiv preprint arXiv:1704.04368* (2017).
- [3] Liu, Linqing, et al. "Generative adversarial network for abstractive text summarization." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. No. 1. 2018.
- [4] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3.Jan (2003): 993-1022.

Web links:

- [5] Comprehensive Guide to Text Summarization using Deep Learning in Python. -
<https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guidetext-summarization-using-deep-learning-python/> o [6] Text

Summarization using Machine Learning. = <https://data-flair.training/blogs/machine-learning-text-summarization/> o [7]

Approaches to Text Summarization: An Overview. =
<https://www.kdnuggets.com/2019/01/approaches-text-summarizationoverview.html>

[8] “Text summarization approaches for NLP ” -
<https://www.machinelearningplus.com/nlp/text-summarizationapproaches-nlp-example/>

o [9] “Beautiful Soup: Build a Web Scraper with Python” accessed
on 07 October 2021 - <https://realpython.com/beautiful-soup-web-scraperpython/>

o [10] “Text Summarization Using Cosine Similarity and Clustering
Approach” - <https://easychair.org/publications/preprint/p2gV>

