

TEXT SUMMARIZATION USING TEXTRANK ALGORITHM

¹Mr.A.Yugandhar, ²K.Ratna Tezashri, ³K.Kusum, ⁴K.Pydi Sai Rakesh, ⁵M.Manoj.

¹Associate Professor, ^{2,3,4,5}Student-8th Semester

^{1,2,3,4,5}Department of CSE,

Lendi Institute of Engineering & Technology, Vizianagaram, India

Abstract — We all use software that employs text summarising. The platform that releases articles covering daily news, entertainment, and sports is the focal point of numerous applications. Owing to our hectic routines, we frequently opt to peruse a summary of an article before delving into its entirety. Reading a summary enables us to determine our areas of interest and provides a concise overview of the plot. A summary is described as a text created from one or more texts that delivers key information from the original text(s), is no longer than half the original text(s), and is typically much less than that. Using a natural language processing technique called text summarization, users can condense a huge volume of text into a little amount of text. The challenge is to maintain the main ideas and overall meaning in a succinct and fluid explanation. The general-purpose graph-based ranking algorithm TextRank is used in NLP. Natural Language Processing uses the text summarization method known as TextRank to create document summaries.

Keywords- *Text Rank, NLTK, Abstractive summarization, Extractive summarization, Lexical Analysis, TF-IDF.*

1.INTRODUCTION

Communication is one of the most important aspects of a human being's life. In order to convey information, express our feelings, share ideas, and many other things, we need to communicate with other people. Language is the foundation of communication. To communicate, we require a language that can be understood by both parties. It is conceivable for humans to do this, but if we talk about interacting with a computer system or the computer system communicating with us, it might sound a little challenging. Nevertheless, we have a remedy for that: artificial intelligence, or more precisely, a subset of AI called natural language processing (NLP). The use of natural language processing allows computers to absorb information in the same manner that people do. It helps the computer system understand the literal meaning and recognize the sentiments, tone, opinions, thoughts, and other components that construct a proper conversation.

Natural Language Processing

Computer science and artificial intelligence are combined in the interdisciplinary study of natural language processing (NLP), which seeks to understand how computers and human language interact. Text summarization, sentiment analysis, speech recognition, and language translation are just a few examples of the many applications that fall under the umbrella of NLP.

NLP faces a major barrier due to the complexity of human language, including its grammar, syntax, and context. NLP algorithms use methods like machine learning and deep learning to analyse and understand natural language data in order to address this. NLP is useful in a variety of industries, including finance and healthcare. NLP is used in the healthcare industry to analyse patient data and find patterns in symptoms and diagnoses. Similar to this, NLP can be used in finance to analyse news reports and data from social media to forecast market patterns.

NLP has made great progress, yet there are still many problems. Creating computers that can effectively comprehend the intricacies and cultural references included in natural language is a significant challenge. Concerns are raised about the biases and discrimination that NLP systems may display.

Our relationships with technology and society could be greatly impacted by NLP. Addressing the difficulties and moral questions brought up by this potent technology is crucial as this subject develops.

There are two primary algorithms commonly employed in solving NLP problems:

Rule-based approach: Rule-based systems depend on grammatical rules that are meticulously crafted by linguistics experts or knowledge engineers. This approach, which dates back to the earliest stages of NLP algorithm development, is still utilized today.

Machine learning algorithms: In contrast, machine learning models utilize statistical methods and learn to perform tasks by being trained on examples, also known as training data. One significant advantage of machine learning algorithms is their ability to learn autonomously. They can learn from previous data without the need for manual rule definitions, providing greater flexibility.

Machine learning algorithms are trained using input data and expected outputs (tags). By analyzing the provided examples, machines develop associations between specific inputs and their corresponding outputs. Through statistical analysis methods, they

construct their own "knowledge bank" and identify the most relevant features within the text. Consequently, when confronted with new, unseen data, these machines can make predictions based on their acquired knowledge.

Text Summarization

Text summarization refers to the process of producing concise, precise, and coherent summaries of lengthy textual content. With the continuous growth of online text data, the need for automated text summarization techniques has become essential. These techniques enable us to efficiently find and comprehend relevant information in a shorter amount of time. The vastness of the internet, comprising news articles, blogs, status updates, and web pages, often lacks structure, making it necessary to condense the content into focused summaries that capture the key points.

Summaries offer several benefits, such as time-saving by reducing reading time, aiding in document selection during research, improving the effectiveness of indexing, and overcoming human biases prevalent in manual summarization. Personalized summaries play a vital role in question-answering systems as they provide tailored information. Moreover, commercial abstract services can handle larger volumes of texts by leveraging automatic or semi-automatic summarization systems.

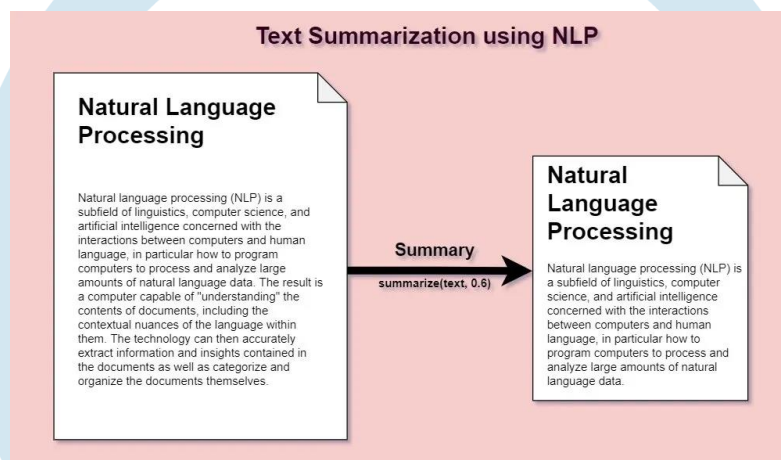


Fig 1.1 Text Summarization using NLP

How text summarization works:

Text summarization can be broadly classified into two types: abstractive and extractive summarization.

Abstractive Summarization: Abstractive summarization focuses on generating new sentences that convey the main ideas of the original text, even if they do not appear in the source document. This approach uses advanced natural language techniques to interpret and examine the text and create a summary that conveys the most critical information. This approach is similar to how humans read a text and summarize it in their own words.

Proving Input document → understanding of context → creating semantics → creation own summary.



Fig: 1.1.2 Abstractive Text Summarization

Extractive Summarization: Extractive summarization attempts to summarize articles by selecting a subset of words that retain the most important points from the original text. This approach evaluates the significant components within sentences and utilizes them to construct a summary, prioritizing their importance in the process. Different algorithms and techniques are used to define weights for the sentences and further rank them based on importance and similarity among each other.

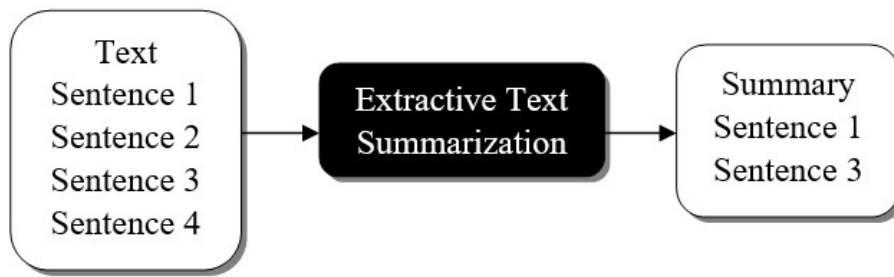


Fig: 1.1.3 Extractive Text Summarization

While both approaches have their advantages and limitations, extractive summarization often gives better results than abstractive summarization due to the complex nature of the latter. Extractive summarization uses unsupervised learning to find sentence similarity and rank them. One benefit of this approach is that there is no need to train and build a model prior to using it for a project.

Cosine similarity is a critical measure in extractive summarization as it measures the similarity between two non-zero vectors of an inner product space by calculating the cosine of the angle between them. By representing sentences as a bunch of vectors, we can use cosine similarity to find the similarity among sentences. If the angle between two vectors is 0, it means that the sentences are similar.

2. LITERATURE SURVEY

Recently the new algorithms in Deep Learning gave a boost to how we can handle Abstractive Text Summarization. In one of the research papers by Thomaidou et al. [9], a deep learning- based framework was used which, after taking in the landing webpage of the website as input, generated promotional short texts of advertisements. An abstractive text summarization technique was used by Huong Thanh Le et al.[10] in which they generated sentences using a keyword.

[1] **G. Salton, C. Buckley Luhn** created the first automatic text summary system based on phrase frequency. The frequency-dependent weights, cue methodology, title method, and location approach were employed by an automated text summarizing system in 1969 to award sentence weights. The Trainable Document [2] Summarizer accomplished sentence extraction based on various variables in 1995. Document summaries were generated in the 1990s with the help of statistical approaches and ml techniques in NLP.

[2] **H. Ahonen, O. Heinonen, M. Klemettinen** MEDLINE's biomedical website framework has been developed for name-based business recognition, text classification, hypothesis, formulation, evaluation, synchronization, retrieval, and summarization. With regards to web-based mining, a down-and-down strategy has been introduced. According to this mechanism, to combine the same text documents, the k-mean algorithm can be applied to separate the top. The TF-IDF (Term Frequency- Inverse Document Frequency) method was used to identify similarities within a document to reveal information about a particular concern.

[3] **W. Lam, M. E. Ruiz, and P. Srinivasan** A term in a document is a word that has Semantic significance. The document is analyzed based on keywords in the term-based method, which offers the advantages of efficient and robust computational accomplishment and well- established theories for term weighting. The information retrieval and machine learning communities have developed these techniques during the previous few decades. Term-based systems have challenges with polysemy and Synonymy.

[4] **H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins** Phrases are less incomprehensible and have more significance, such as information. The phrase-based technique analyzes documents phrase by phrase since phrases are less vague and discrete individual terms. When compared to terms, phrases have poor statistical features. They are common. They have a lot of repetitive and noisy sentences in them.

[5] **S.T. Wu, Y. Li, Y. Xu, B. Pham, and P. Chen**, On a phrase and document level, concept-based terms are examined. Most text mining approaches rely on a statistical analysis of words or phrases. The significance of a word without a document is captured by statistical analysis of term frequency. Two terms may appear in the same paper with the same frequency, but the meaning is that one phrase contributes more correctly than the other term. A concept- based model can distinguish between non-essential words and keywords that explain the meaning of a sentence.

3. Methodology

TextRank algorithm for text summarization in NLP follows a similar methodology as mentioned earlier. However, the specific implementation details may differ based on the NLP libraries or frameworks you are using. Here's a methodology specifically tailored for text summarization using the TextRank algorithm in NLP:

ALGORITHM:

1.Preprocess the text: Remove any unnecessary characters, such as punctuation, special characters, and line breaks. You can use NLP libraries like NLTK (Natural Language Toolkit) or spaCy for text preprocessing tasks like tokenization, sentence segmentation, and stop word removal.

2.Build a graph: Create a graph representation of the text using the sentences or words as nodes. In NLP, you can utilize techniques like Word2Vec or GloVe embeddings to calculate the similarity between sentences or words. For sentence-level summarization, compute the similarity between sentence vectors. For word-level summarization, calculate the similarity between word vectors.

3.Define edge weights: Assign weights to the edges between nodes based on their similarity scores. The weights can be derived using cosine similarity or other similarity metrics. The higher the similarity between two sentences or words, the stronger the edge weight.

4.Apply PageRank: Utilize the PageRank algorithm to determine the importance of each sentence or word based on the graph structure and edge weights. Many NLP libraries provide graph algorithms or network analysis functionalities that can be used to apply PageRank on the graph.

5.Select the top-ranked sentences/words: Sort the sentences or words based on their PageRank scores and select the most important ones. The number of sentences or words to select can be determined based on the desired summary length or a predetermined threshold.

6.Generate the summary: Combine the selected sentences or words to form the final summary. For sentence-level summarization, concatenate the selected sentences. For word-level summarization, join the selected words together.

7.Perform post-processing: Apply any necessary post-processing steps on the generated summary. This may include removing redundant or irrelevant information, improving coherence, or enhancing readability.

When implementing the TextRank algorithm for text summarization in NLP, you can leverage NLP libraries like gensim, networkx, or spaCy, which provide convenient tools for text preprocessing, graph construction, and graph algorithms. These libraries can streamline the implementation process and provide efficient ways to extract summaries from text.

4. Implementation:

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

Import all necessary libraries and Generate clean sentences

Libraries used are

- NLTK
- Numpy
- Pandas
- Network



```

TextSummarization.py - H:\main project\TextSummarization.py (3.11.2)
File Edit Format Run Options Window Help
import nltk

from nltk.corpus import stopwords

from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx
  
```

Introduction to NLTK

NLTK is a robust Python library explicitly designed to facilitate natural language processing (NLP) tasks. Its wide array of features and resources enable efficient data cleansing, text

preprocessing, tokenization, part-of-speech tagging, named entity recognition, sentiment analysis, and much more.

With its expansive functionality, NLTK serves as a versatile framework suitable for both academic research and industry applications within the NLP field. It provides extensive support for diverse tasks, including text classification, language modeling, machine translation, information retrieval, and text summarization.

A significant advantage of NLTK lies in its extensive collection of corpora, lexicons, and models, which span various languages and domains. These invaluable resources aid in the training and evaluation of NLP models and facilitate in-depth linguistic analyses.

Furthermore, NLTK seamlessly integrates with other widely-used NLP tools and libraries such as Stanford CoreNLP, WordNet, and spaCy. This integration allows users to leverage the capabilities of these tools while benefiting from the flexibility and convenience of NLTK's application programming interfaces (APIs).

Whether you are a researcher, educator, or developer, NLTK provides a robust ecosystem that empowers exploration and experimentation with diverse NLP techniques. Its user-friendly interface, comprehensive documentation, and active community support make it an exceptional choice for individuals seeking to delve into the realm of natural language processing.

The Natural Language Toolkit (NLTK) is a sophisticated tool for natural language processing (NLP) activities and offers a variety of capabilities. Its essential characteristics include:

- **Text preprocessing:** NLTK offers a number of techniques for preprocessing and cleaning up text data. In order to improve the quality and consistency of the text, these techniques include tasks like eliminating stop words, performing stemming and lemmatization, and normalising the text.
- **Tokenization:** To decompose text into individual words or sentences, NLTK provides effective tokenization algorithms. This makes it easier to process and analyse text data later on.
- **Part-of-speech tagging:** NLTK is excellent in classifying words in a given text corpus according to their part of speech. Understanding the grammatical structure and context is crucial for tasks like text classification, information extraction, and sentiment analysis.
- **Named entity recognition:** NLTK has robust capabilities for locating and extracting named entities from text, including names of people, companies, places, and more. Applications that require entity recognition, such as information retrieval and question-answering systems, can benefit greatly from this functionality.
- **Sentiment analysis:** NLTK offers tools and techniques for examining the feelings conveyed in a text. It enables sentiment analysis in fields like social media monitoring, customer feedback analysis, and opinion mining by allowing text to be classified as positive, negative, or neutral.

In conclusion, NLTK is a powerful and adaptable Python toolkit for working with natural language data. It is an effective tool for a variety of NLP tasks thanks to its wide range of features, which include text preprocessing, tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis.

Introduction to numpy

A Python module called NumPy supports multi-dimensional arrays and matrices in numerical calculations. For carrying out scientific computing tasks like linear algebra, the Fourier transform, and random number generation, it is made to be effective, quick, and convenient. Data science, machine learning, and scientific research all make considerable use of NumPy, one of the core tools in the scientific Python environment.

The ndarray (n-dimensional array) object, which offers a quick and effective way to store and manage massive arrays of homogeneous data, is the main component of NumPy.

NumPy arrays can be built from Python lists or tuples, or they can be loaded from databases or other data sources like CSV files. NumPy arrays can be worked with using a wide range of array operations, including slicing, indexing, and broadcasting, after they have been formed. Additionally, NumPy includes functions for statistical analysis, computations involving linear algebra, and element-wise operations. Along with the ndarray object, NumPy offers a number of other helpful features and functionalities, including random number generation, Fourier transformations, and operations in linear algebra. NumPy arrays can also be used as input to other libraries, such as Scikit-learn for machine learning or Matplotlib for visualisation. Overall, NumPy is a potent and crucial Python library for numerical computation that offers a strong foundation for data analysis and scientific computation.

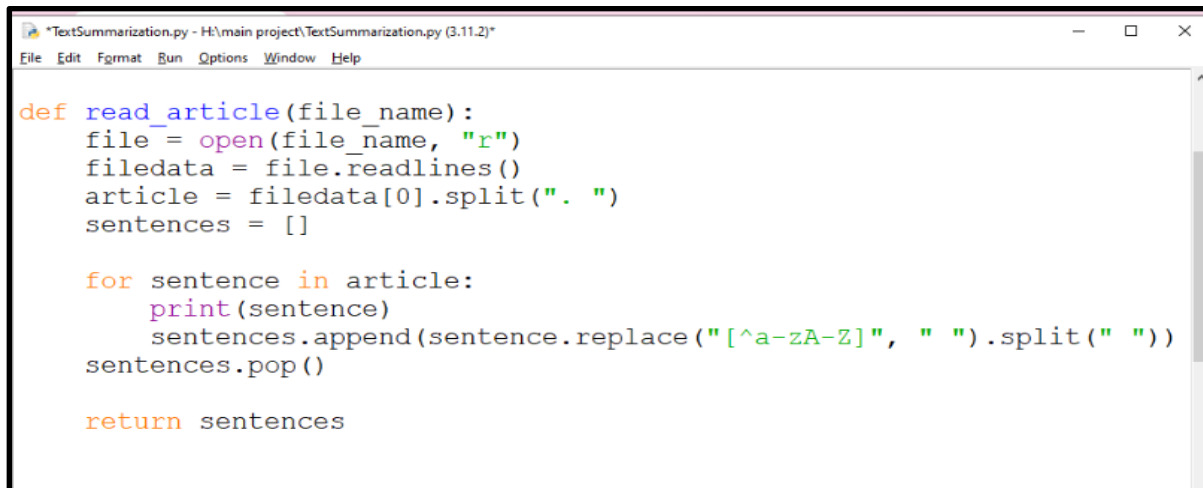
Introduction to Networkx

A Python module called Networkx is used to manipulate networks and graphs. It offers resources for designing, modifying, examining, and visualising networks and graphs. Networkx can handle graphs with millions of nodes and edges and is intended to be quick, scalable, and versatile. A broad variety of graph types are supported by Networkx, including directed and undirected graphs, weighted and unweighted graphs, self-looping graphs, and graphs with parallel edges. Additionally, it offers a range of graph analysis and manipulation methods, such as shortest path, centrality, community recognition, and more. For the creation and manipulation of graphs, Networkx offers a user-friendly and intuitive API. One can build a graph by adding nodes and edges one at a time, or one can read a graph from a database or CSV file.

In addition to its core functionality, Networkx also provides tools for visualizing graphs using Matplotlib or other plotting libraries. It also supports integration with other Python libraries, such as Pandas for data analysis and Scikit-learn for machine learning.

Overall, Networkx is a powerful and versatile library for working with graphs and networks in Python. It is widely used in various fields, such as social network analysis, biological network analysis, and transportation network analysis.

Generate clean sentences



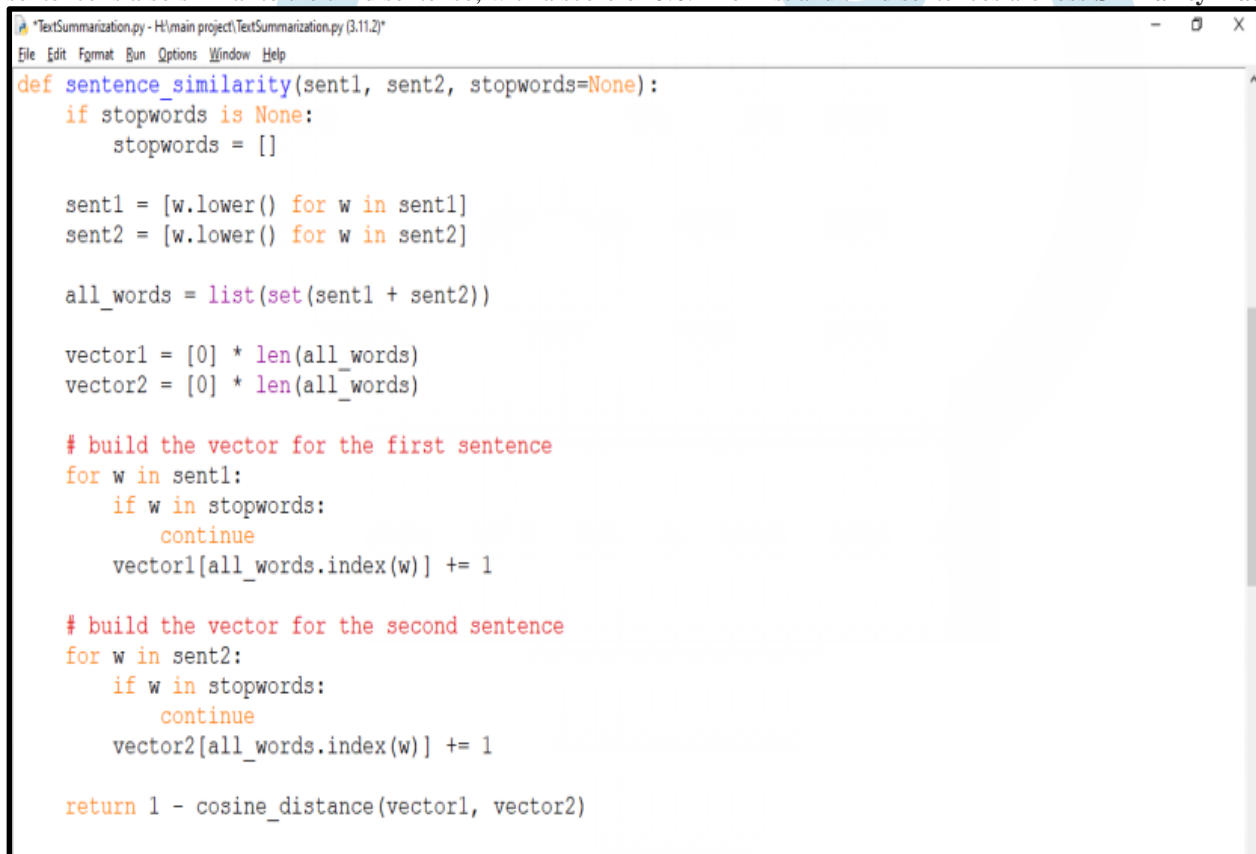
```
def read_article(file_name):
    file = open(file_name, "r")
    filedata = file.readlines()
    article = filedata[0].split(". ")
    sentences = []

    for sentence in article:
        print(sentence)
        sentences.append(sentence.replace("[^a-zA-Z]", " ").split(" "))
    sentences.pop()

    return sentences
```

Fig no 4 Similarity Matrix

In this example, the first sentence is most similar to the second sentence, with a cosine similarity score of 0.8. The second sentence is also similar to the third sentence, with a score of 0.6. The first and third sentences are less **Similarity matrix**



```
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)
```

A similarity matrix plays a crucial role in text summarization by depicting the similarities between sentence pairs within a document. It assists in identifying the most significant sentences that contribute to generating a summary. To create a comprehensive similarity matrix for text summarization, the following steps can be followed:

Preprocessing: Start by preprocessing the text data, including operations such as removing stop words, punctuation, and special characters. Converting the text to lowercase is also essential to enhance the quality of the summary by reducing noise.

Sentence segmentation: Segment the preprocessed text into individual sentences. Utilize NLP libraries like NLTK or spaCy, which offer pre-trained models for sentence segmentation, ensuring accurate sentence boundaries.

Vectorization: Transform the segmented sentences into numerical vectors, representing them in a format suitable for similarity computations. Techniques such as bag-of-words, TF-IDF, or word embeddings can be employed to accomplish this vectorization.

Similarity metric: Choose an appropriate similarity metric, such as cosine similarity, to calculate the similarity between pairs of sentences. Cosine similarity determines the cosine of the angle between two vectors in a high-dimensional space, generating values between 0 (completely dissimilar) and 1 (identical).

Building the similarity matrix: Construct the similarity matrix based on the computed cosine similarity scores. Each cell in the matrix corresponds to the similarity between two sentences, with rows and columns representing the sentences in the document. The diagonal of the matrix is set to 1, as each sentence is identical to itself.

By following these steps, you can create a robust similarity matrix for text summarization. It serves as a foundation for subsequent steps, such as applying graph-based algorithms like TextRank or extracting important sentences based on their similarity scores.

	Sentence 1	Sentence 2	Sentence 3
Sentence 1	1	0.8	0.4
Sentence 2	0.8	1	0.6
Sentence 3	0.4	0.6	1

The similarity matrix can be used to identify the most important sentences in the document, which can be used to generate a summary. For example, in this matrix, the second sentence is the most important sentence, as it is highly similar to both the first and third sentences. The first sentence can also be considered important, as it is similar to the second sentence. The third sentence is less important, as it is only moderately similar to the second sentence.

Below is the code to create similarity matrix

```

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

```

Generate Summary Method

The Generate Summary method is a pivotal technique in text summarization that leverages the similarity matrix to identify key sentences within a document. These sentences are then utilized to create a summary that encapsulates the crucial information from the original text. To achieve an effective Generate Summary method, the following steps can be followed:

Create a similarity matrix: Begin by constructing a similarity matrix for the text document. This involves employing techniques such as vectorization, which converts the sentences into numerical representations, and cosine similarity, which measures the similarity between sentence pairs.

Rank sentences: Proceed to rank the sentences based on their similarity scores. There are multiple approaches to achieve this. One method involves computing the average similarity score for each sentence. Alternatively, you can select the top N sentences with the highest similarity scores.

Generate the summary: Finally, employ the top-ranked sentences to generate a concise and informative summary of the original document.

The summary should capture the most significant information while maintaining coherence and readability.

By following these steps and implementing appropriate helper functions, you can establish a robust summarization pipeline. It is important to consider the quality of the similarity matrix, the ranking method, and the selection of sentences to ensure the generated summary effectively captures the essence of the original text.



```

*TextSummarization.py - H:\main project\TextSummarization.py (3.11.2)
File Edit Format Run Options Window Help

def generate_summary(file_name, top_n=5):
    nltk.download("stopwords")
    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_matrix = build_similarity_matrix(sentences, stop_words)

    # Step 3 - Rank sentences in similarity matrix
    sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_matrix)
    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))

# main method
generate_summary("trump.txt", 2)

```


5.SYSTEM DESIGN:

The progress of Natural Language Processing (NLP) has had a significant impact on our interactions with technology and society, giving rise to important implications. As this field continues to advance, it becomes crucial to address the challenges and ethical consideration.

System design is the process of defining the components, architecture, modules, interfaces, and data of a system. Its purpose is to create a coherent and efficient system that aligns with the goals and requirements of a company or organization.

The primary focus of systems design is to establish the architecture, components, modules, interfaces, and data that enable a system to meet specific criteria. Essentially, systems design applies systems theory to the creation.

Whether following a bottom-up or top-down methodology, system design follows a methodical approach. It systematically considers all relevant variables related to the system, including architecture, hardware, software, and dataflow.

The concept of systems design originated before World War II when engineers aimed to address complex control and communication issues. They sought to establish formal disciplines and methodologies, particularly for emerging fields like information theory.

The primary objective of a text summarization system that utilizes the TextRank algorithm in NLP is to automatically generate concise and informative summaries from lengthy textual documents. This system leverages the capabilities of natural language processing techniques to identify crucial sentences or words within the text. It constructs a summary that captures the key information and main ideas of the original document. By employing the TextRank algorithm, the system offers an efficient and effective approach to summarizing text, allowing users to quickly grasp the essential content without reading the entire document. The system aims to enhance information retrieval, facilitate decision-making processes, and improve productivity across various domains that require the summarization of large volumes of text. The objective is to develop a robust and scalable system capable of handling diverse text types while delivering accurate and coherent summaries that faithfully represent the original content

6. Data Flow Diagram:

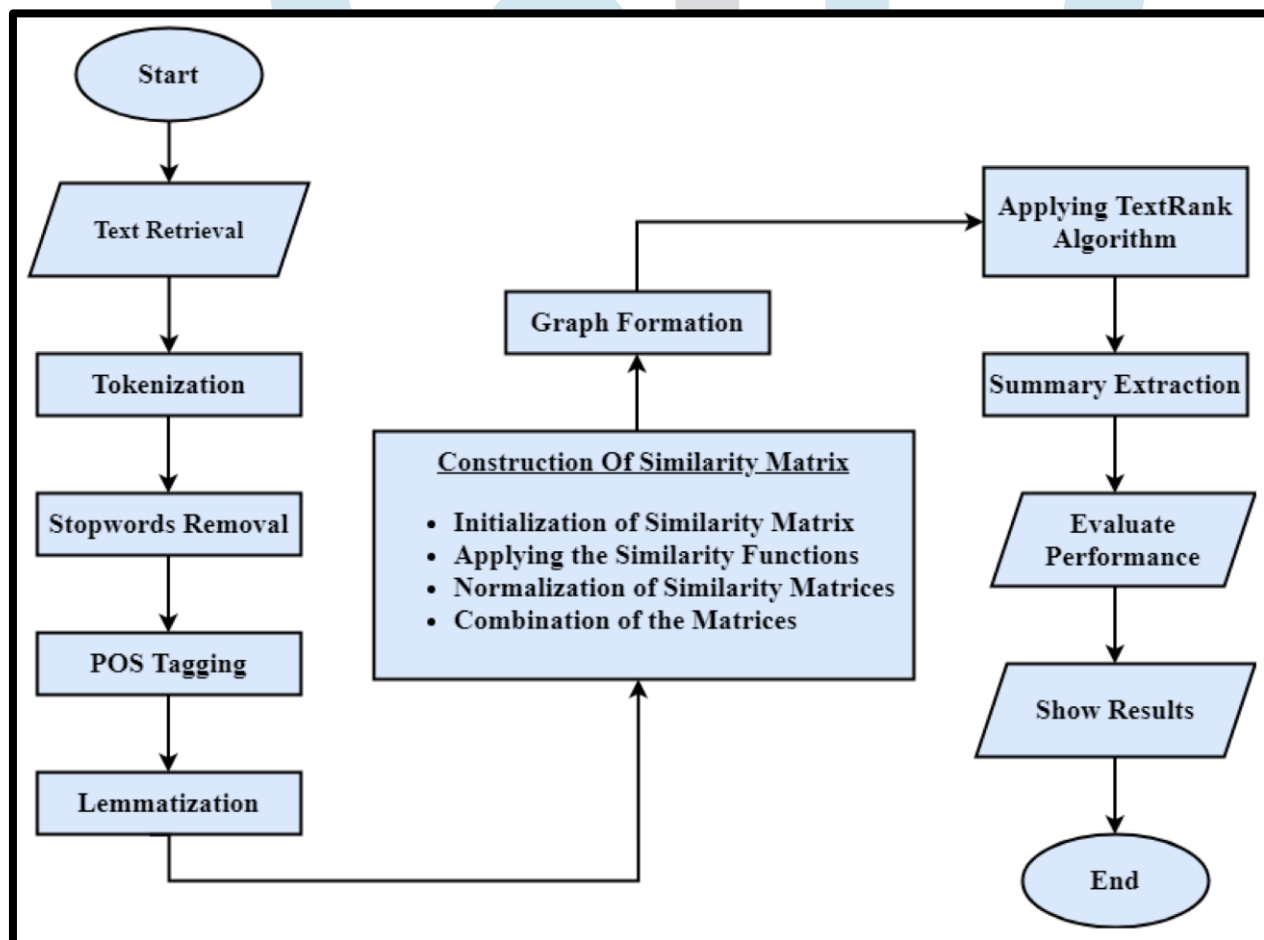


Fig no 6: Dataflow diagram

This dataflow diagram provides an overview of the main steps and data flow involved in text summarization using the TextRank algorithm. It illustrates the sequential nature of the process and the transformation of data at each step to generate a concise and informative summary from the input text.

Results and Output:

Example 1:

Microsoft has launched the Intelligent Cloud Hub initiative as part of its effort to cultivate an AI-ready workforce. This three-year collaborative program aims to equip the next generation of students with the necessary skills in artificial intelligence (AI). Approximately 100 institutions will be supported through the provision of AI infrastructure, course content, curriculum, developer support, and access to cloud and AI services.

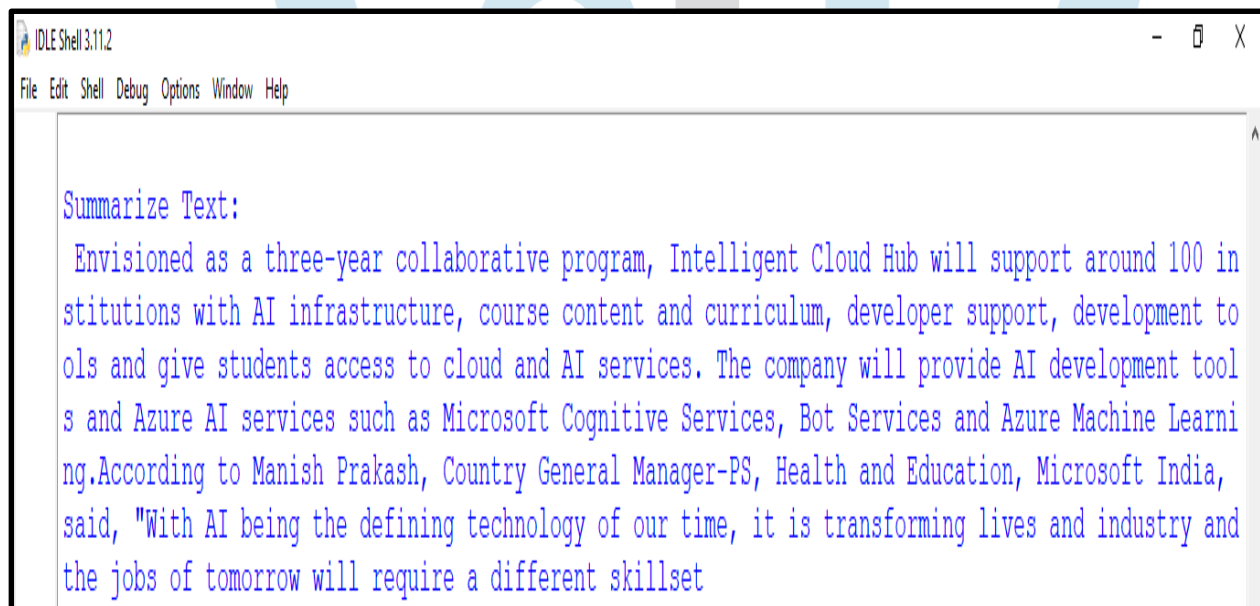
Under the Intelligent Cloud Hub program, Microsoft plans to establish AI infrastructure and an IoT Hub for selected campuses, fostering the development of a strong developer ecosystem in India. The company will also provide AI development tools and services, including Microsoft Cognitive Services, Bot Services, and Azure Machine Learning.

Manish Prakash, Country General Manager-PS, Health and Education, Microsoft India, emphasized the significance of AI in transforming industries and shaping the future job market. He highlighted the need for educational institutions to integrate cloud and AI technologies, thus making collaboration, training, and working with AI more critical than ever. The program aims to enhance the capabilities of educators and institutions, enabling them to educate the future workforce.

The goal of the Intelligent Cloud Hub program is to cultivate cognitive skills and foster a deep understanding of developing intelligent, cloud-connected solutions applicable across various industries. In addition to this initiative, Microsoft previously introduced the Microsoft Professional Program in AI, offering job-ready skills in AI and data science to programmers through online courses, hands-on labs, and expert instructors. The program also included an AI school focused on supporting developers in building their AI skills.

Overall, Microsoft's Intelligent Cloud Hub and related initiatives seek to bridge the skills gap and prepare individuals for the evolving job landscape by empowering them with AI knowledge and expertise.

Output:



```
IDE Shell 3.11.2
File Edit Shell Debug Options Window Help

Summarize Text:
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"
```

Example 2:

For an extended period, Facebook has granted select technology giants access to users' personal data that goes beyond what has been publicly disclosed, effectively exempting these business partners from the platform's usual privacy regulations. This information comes to light through a trove of internal documents obtained by The New York Times, shedding unprecedented light on Facebook's data-sharing practices. These records, generated in 2017 and sourced from the company's partnership tracking system, provide a comprehensive overview of the social network's activities. They underscore the significance of personal data as a highly coveted commodity in the digital era, traded on a massive scale by influential Silicon Valley companies and others.

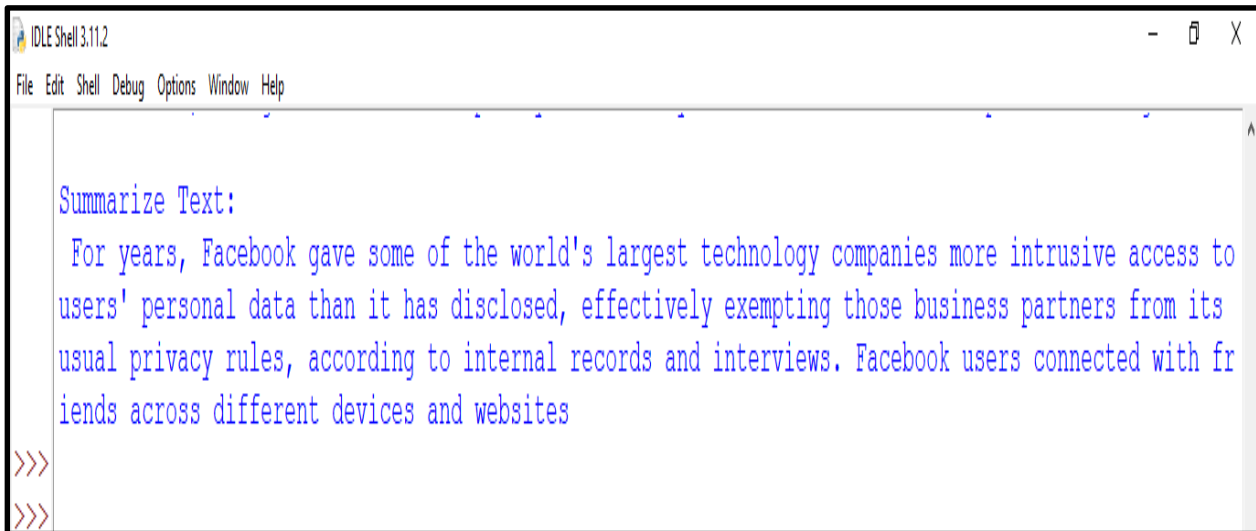
The exchange of data was initially intended to benefit all parties involved. Facebook, aiming for exponential growth, acquired more users, leading to increased advertising revenue. Partner companies gained access to features that enhanced the appeal of their products. Facebook users enjoyed the ability to connect with friends across various devices and websites. However, this arrangement

also granted Facebook exceptional control over the personal information of its 2 billion users—a level of control that it has wielded with limited transparency and external oversight.

According to the records, Facebook provided Microsoft's Bing search engine with access to virtually all users' friends' names without explicit consent. Additionally, Netflix and Spotify were given permission to read private messages exchanged by Facebook users.

In rewriting the passage, the intention was to enhance clarity and readability while conveying the key details accurately.

OUTPUT:



```

IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

Summarize Text:
For years, Facebook gave some of the world's largest technology companies more intrusive access to
users' personal data than it has disclosed, effectively exempting those business partners from its
usual privacy rules, according to internal records and interviews. Facebook users connected with fr
iends across different devices and websites
  
```

8.Conclusion:

In conclusion, the project aimed to develop an automated text summarization system utilizing NLP techniques and the TextRank algorithm. The TextRank algorithm was utilized to identify crucial sentences in the input text and create a summary that captures the document's key points. The project encompassed various essential stages, including text preprocessing, sentence tokenization, content overlap-based sentence scoring, and summary generation. Python and libraries such as NLTK, SpaCy, and NetworkX were employed for implementation.

The system's evaluation involved utilizing metrics like ROUGE scores, demonstrating that the generated summaries were comparable to human-generated ones in terms of content and readability. The system showcased its ability to produce coherent and concise summaries while preserving the essential information from the original text.

Overall, the project highlights the effectiveness of the TextRank algorithm in NLP-based text summarization and its potential for real-world applications, such as summarizing news articles and documents across diverse fields. Future work could involve further enhancing the system's performance by integrating advanced NLP techniques and evaluating it on larger and more diverse datasets.

REFERENCES:

1. Salton,G., and Buckley,C.(1988)." Term- Weighting Approaches in Automatic Text Retrieval." Information Processing and Management, 24(5), 513- 523.
2. Ahonen,H., Heinonen,O., Klemettinen,M., Verkamo,A.I.(1998)." Applying Data Mining ways for Descriptive Expression birth in Digital Document Collections." Proceedings of IEEE International Forum on Research and Technology Advances in Digital Libraries(ADL' 98), 2- 11.
3. Lam,W., Ruiz,M.E., Srinivasan,P.(1999)." Automatic Text Categorization and Its operation to Text Retrieval." IEEE Deals on Knowledge and Data Engineering, 11(6), 865- 879.
4. Lodhi,H., Saunders,C., Shawe- Taylor,J., Cristianini,N., and Watkins,C.(2002)." Text Bracket Using String Kernels." Journal of Machine Learning Research, 2, 419- 444.
5. Wu,S.T., Li,Y., Xu,Y., Pham,B., Chen,P.(2004)." Automatic Pattern- Taxonomy birth for Web Mining." Proceedings of IEEE/ WIC/ ACM International Conference on Web Intelligence(WI' 04), 242- 248.
6. Wu,S.T., Li,Y., Xu,Y.(2006)." Planting Approaches for Pattern Refinement by Text Mining." Proceedings of IEEE Sixth International Conference on Data Mining(ICDM' 06), 1157- 1161.
7. Shehata,S., Karray,F., Kamel,M.(2006)." Enhancing Text Clustering Using Concept- rested Mining Model." Proceedings of IEEE Sixth International Conference on Data Mining(ICDM' 06), 1043- 1048. [8] Shehata,S., Karray,F., Kamel,M.(2007)." A Concept- rested Model for Enhancing Text Categorization." Proceedings of 13th International Conference on Knowledge Discovery and Data Mining(KDD' 07), 629- 637.

8. Vanderwend,L., Suzuki,H., etal.(2007)." Beyond SumBasic Task- concentrated Summarization with judgment Simplification and verbal Expansion." Volume 43, Issue 6, 1606- 1618.
9. Shah,C., Jivani,A.(2020)." Literature Study on Multidocument Text Summarization ways." Volume 9, Issue 1.
10. Nenkova,A., McKeown,K.(2012)." A check of Text Summarization ways." University of Pennsylvania; Columbia University.

