| **Name : Yash Sarang** | **Class/Roll No. : D16AD / 47** | **Grade :** |
| --- | --- | --- |

**Title of Experiment :**

Implementation of News summarizer using NLP

**Problem Statement :**

The problem is to develop a News Summarizer using Natural Language Processing (NLP) techniques. The goal is to create an automated system that can extract the most important information from a news article and generate a concise summary, making it easier for users to grasp the main points of the news quickly.

**Description / Theory :**

Description/Theory: News summarization is a process that involves condensing the essential information from a news article while retaining its core message and meaning. There are two primary approaches to news summarization: extractive and abstractive summarization. In extractive summarization, important sentences or phrases are selected directly from the article to form the summary. On the other hand, abstractive summarization involves paraphrasing and generating novel sentences to create a summary.

**Working of Model / Flowchart** :

Input (News Article URL and Number of Sentences for Summary):

* + The user provides a URL of a news article and specifies the number of sentences they want in the summary.

Fetching and Preprocessing the Article:

* + The summarize\_article() function first fetches the article content from the provided URL using the Article class from the newspaper3k library.
  + It then parses the article content and performs natural language processing (NLP) on it using the nlp() method to extract entities and keywords. This helps in improving the accuracy of the summary.

Tokenization of Sentences:

* + The article text is tokenized into individual sentences using the sent\_tokenize() function from the NLTK library.
  + This step is essential to analyze the content at the sentence level.

Vectorization of Sentences:

* + A CountVectorizer is created and fitted with the tokenized sentences.
  + The CountVectorizer converts the sentences into numerical vectors, representing the frequency of words in each sentence.

Cosine Similarity Calculation:

* + The code calculates the cosine similarity between the vectors of all sentence pairs using the cosine\_similarity() function from the scikit-learn library.
  + Cosine similarity measures the cosine of the angle between two non-zero vectors and is used to determine the similarity between sentences.

Sentence Ranking:

* + The cosine similarity matrix is summed row-wise to calculate the score for each sentence.
  + Higher scores indicate that a sentence is more similar to other sentences in the article.
  + The sentences are then sorted in descending order based on their scores, and the top-ranked sentences are selected.

Output (Summarized Text):

* + Finally, the top-ranked sentences are combined to generate the summary of the news article.
  + The summarized text is then returned by the summarize\_article() function and printed for the user to read.

| **Program:**  The code provided demonstrates the algorithm by fetching a news article from a given URL, tokenizing it into sentences, vectorizing the sentences, calculating cosine similarity, ranking the sentences based on the similarity scores, and generating a summary by selecting the top-ranked sentences.  !pip install nltk newspaper3k  import nltk  from newspaper import Article  from nltk.tokenize import sent\_tokenize  from sklearn.metrics.pairwise import cosine\_similarity  from sklearn.feature\_extraction.text import CountVectorizer  nltk.download('punkt')  # Function to fetch and summarize news article  def summarize\_article(url, num\_sentences=3):  # Download the article  article = Article(url)  article.download()  article.parse()  article.nlp()  # Tokenize the article into sentences  sentences = sent\_tokenize(article.text)  # Create a CountVectorizer to convert sentences to vectors  vectorizer = CountVectorizer().fit\_transform(sentences)  vectors = vectorizer.toarray()  # Calculate cosine similarity between sentence vectors  similarity\_matrix = cosine\_similarity(vectors)  # Sort sentences by similarity score  sentence\_scores = [sum(similarity\_matrix[i]) for i in range(len(sentences))]  ranked\_sentences = sorted(((sentence\_scores[i], s) for i, s in enumerate(sentences)), reverse=True)[:num\_sentences]  # Get the summarized text  summarized\_text = " ".join([s[1] for s in ranked\_sentences])  return summarized\_text  # Sample news article URL  article\_url = "https://www.theguardian.com/sport/live/2023/oct/05/england-v-new-zealand-cricket-world-cup-2023-opener-live-updates"  # Summarize the news article  summary = summarize\_article(article\_url)  # Display the summary  print("Article Summary:")  print(summary)  **Output:**  Article Summary:  Updated at 09.02 EDT 10h ago 08.56 EDT 4th over: New Zealand 19-1 (Conway 10, Ravindra 8) The reason it’s a game of two ends is because Curran has got the length right – aiming for the top of off – and he’s moving the ball both ways. Photograph: Andrew Boyers/Reuters Updated at 10.13 EDT 9h ago 09.28 EDT 11th over: New Zealand 92-1 (Conway 44, Ravindra 47) Buttler keeps the faith with Wood, but again the first ball of his over is struck for four – a classical straight drive from Conway. 8h ago 10.35 EDT 28th over: New Zealand 208-1 (Conway 108, Ravindra 97) Moeen returns and, as too often, the first ball of the over goes for four – a pull from Ravindra, who has been quietly motoring along at a run a ball. |
| --- |

**Algorithm:**

**Extractive Summarization Algorithm using Cosine Similarity and Sentence Ranking**

Input: News article URL and desired number of sentences for summary.

Fetch and Preprocess the Article:

* + Fetch the news article content from the provided URL.
  + Tokenize the article into sentences using sent\_tokenize() from the NLTK library.

Vectorization of Sentences:

* + Create a CountVectorizer to convert each sentence into a numerical vector.
  + Convert the sentences to vectors and form a matrix of sentence vectors.

Cosine Similarity Calculation:

* + Calculate the cosine similarity between the vectors of all sentence pairs.
  + Cosine similarity is used to measure the similarity between sentences based on their vector representations.

Sentence Ranking:

* + Calculate the sentence scores by summing the cosine similarity scores for each sentence.
  + Higher scores indicate sentences that are more similar to others and are likely to contain essential information.

Select Top Sentences:

* + Rank the sentences based on their scores in descending order.
  + Select the top num\_sentences sentences as the summary based on the highest scores.

Output: Summarized Text:

* + Combine the selected top sentences to form the summarized version of the article.

**Results and Discussions :**

The results will include the generated summary for a given news article. The discussion will focus on the accuracy and conciseness of the summary, comparing it to the original article. The evaluation may include metrics such as Rouge scores to measure the quality of the summary.

**Conclusion:**

News summarization using NLP is a valuable tool for quickly obtaining the main points of a news article. Extractive summarization provides an effective way to generate summaries by selecting significant sentences. However, abstractive summarization, though more challenging, can potentially create more coherent and natural-sounding summaries. Implementing an automated news summarizer enhances information accessibility and can be used in various applications to improve news consumption. Further research and development can lead to more advanced summarization techniques and improved summary quality.

**\*\*\*\*\*\***