**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1) SYSTEM SPECIFICATIONS & REQUIREMENTS**

**3.1.1) NUMPY**

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project.

**3.1.2) PANDA**

Panda is a python library for data analysis. It was started by Wes McKinney in 2008.It has functions for analyzing, cleaning, exploring, and manipulating data. The name “Pandas” has a reference to both “Panel Data”, and “Python Data Analysis” .Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values.

**3.1.3) NLTK**

The Natural Language Toolkit or NLTK, is one of the premier libraries for developing Natural Language Processing (NLP) models. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

**3.1.4) PYTHON**

Python is an interpreted, object-oriented & high-level programming language developed by Guido van Rossum. It was originally released in 1991.Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn’t specialized for any specific problems. Python runs on an interpreter system, meaning that code can be executed as soon as it is written.

**3.1.5) VISUAL STUDIO CODE**

Visual studio code, also commonly referred to as a VS Code, is a source-code editor made by Microsoft with the Electron framework, for windows, Linux and macOS. Visual Studio Code is a free coding editor that helps us to start coding quickly. Any programming language can be coded in VS code. Visual Studio Code has support for many languages including python, java , C++, Javascript, and more.

**3.1.6) RE (REGULAR EXPRESSION)**

The Python "re" module provides regular expression support. In Python a regular expression search is written as: match = re. search(pat, str) The re.search() method takes a regular expression pattern and a string and searches for that pattern within the string. The’ re’ package is a built-in module in Python that provides functions for working with regular expressions, which are powerful tools for pattern matching and text manipulation.

**3.1.7) WARNING**

The "warnings" module provides a way to manage and control warning messages that are issued during the execution of a Python program. In NLP, the "warnings" package can be useful when working with external libraries or APIs. By using the "warnings" package, you can control how these warning messages are handled, whether to display them, ignore them, or raise an exception.

**3.1.8) WORD\_TOKENIZE**

word\_tokenize is a function in Python that splits a given sentence into words using the NLTK library. It is a common pre-processing step in NLP tasks where text data needs to be processed at the word level. One popular tool used for word tokenization in NLP is the Natural Language Toolkit (NLTK), which is a library for Python. NLTK provides a method called ’ word\_tokenize()’ that tokenizes a text string into a list of words.

**3.1.9) TEXTBLOB**

TextBlob is a popular Python library for natural language processing (NLP). It provides a simple and intuitive API for common NLP tasks such as noun phrase extraction, sentiment analysis, language translation, and more. TextBlob is built on top of NLTK (Natural Language Toolkit) and offers an easy-to-use interface for NLP operations.

**3.1.10) WORDCLOUD**

Word clouds are a visualization technique commonly used in natural language processing (NLP) to display the most frequent words or terms in a text corpus. It is great for visualizing unstructured text data and getting insights on trends and patterns.

**3.1.11) STOPWORDS**

Stop words are a set of commonly used words in any language. For example, in English, “the”, “is”, & “and”, would easily qualify as stop words. In NLP, stop words are used to eliminate unimportant words, allowing applications to focus on the important words instead.

**3.2) ALGORITHM**

1. Begin

2. Import necessary libraries:

2.1.numpy (as np): For numerical operations

2.2.pandas (as pd): For data manipulation

2.3.warnings: For handling warnings

2.4.re: For regular expression operations

2.5.nltk: Natural Language Toolkit for text processing

2.6.nltk.tokenize: For tokenization

2.7.nltk.sent\_tokenize: For sentence tokenization

2.8.textblob: For text processing and sentiment analysis

2.9.string: For string operations

2.10.nltk.corpus.stopwords: For stopwords removal

2.11.statistics.mean: For calculating the mean

2.12.heapq.nlargest: For finding the largest elements in a list

2.13.wordcloud: For creating word clouds

3. Define 'stop words' & 'punctuations' for initializing set of stopwords and

punctuations.

4. Define 'warning' module to ignore warning messages.

5. Reads three CSV files 'articles1.csv,'articles2.csv' & 'articles3.csv into three

pandas dataframes 'df\_1','df\_2' & 'df\_3'. Compares the column names of

'df\_1' & 'df\_2' using '==' operator

5.1.Repeat the same for 'df\_2' & 'df\_3'

5.2.Creates a list 'd' containing 'df\_1','df\_2' & 'df\_3'

5.3.Concatenates the DataFrames in the list 'd' using 'pd.concat(d,keys==['x','y','z'])'

5.4.Rename the column 'content' to 'article' in the concatenated dataframe

5.5.Prints the first few rows of the concatenated DataFrame using 'df.head()'.

5.6.Prints the shape of the concatenated DataFrame using 'df.shape'

5.7.Drop the 'Unnamed: 0' column from the concatenated dataframe

5.8.Print the first few rows of the modified DataFrame.

6. import 'seaborn' and 'matplotlib' libraries for data visualization.

6.1.It aims to create two count plots to visualize the distribution of publications and articles according to the year.

6.2.Set the figure size for the plots.

6.3.Set the font scale and style for seaborn.

6.4.Create the first count plot for the distribution of publications.

6.5.Rotate the x-axis labels by 45 degrees for better readability.

6.6.Set the labels and title for the first plot.

6.7.Replace a specific value in the 'year' column of the DataFrame 'df'

6.8.Create the second count plot for the distribution of articles by year

6.9.Set the labels and title for the second plot.

7. Analyse a dataframe 'df' that contains the column 'author'

7.1. calculate the frequency count of each unique value in the 'author'

column of the DataFrame 'df'.

7.2. sets the font scale, style, and grid style for the seaborn library using

'sns.set(font\_scale=1,style='whitegrid')'

7.3. selects the top 80 most frequent authors from the DataFrame using 'df\_author=df.author.value\_counts().head(80)'.

7.4. create a 'barplot' function to create a horizontal barplot. The x-axis represents the count of authors, and the y-axis represents the author names.

7.5. Add labels and titles to the x and y axis

7.6. Adjust the plot appearance using various functions

7.7. Replace the contractions

7.8. Define 'contractions\_dict', it maps certain contractions to their expanded forms.

8. import the 're' module for regular expressions.

8.1.Compile a regular expression pattern ' contractions\_re' using 're.compile'.

8.2. Define a function name called 'cleanhtml' that takes a single parameter 'raw\_HTML'

8.3. Compile another regular expression pattern 'cleanr' to match and remove HTML tags using the '<.\*?>' pattern

8.4. use 're.sub' to substitute the matches of 'cleanr' in 'raw\_ html' with an empty string, effectively removing the HTML tags.

8.5. store the result in 'cleantext'

8.6. return the cleaned text as output of the function.

9. Define a function named 'expand\_contractions' that takes two

parameters:'s' & 'contractions\_ dict'.

9.1. Define an inner function named 'replace' that takes a single parameter 'match'.

9.2. inside 'replace' return the expansion of the matched contraction 'match.group(0)' from the ' contractions\_dist' .

9.3.Define a function named 'preprocessing' that takes a single

parameter 'article'.

9.4. converted the article to lowercase

9.5. remove the HTML tags using ' cleanhtml'

9.6. use regular expression 're' for removing email addressses & URLs

9.7.Replace the non-breaking space character with a regular space character.

9.8. expand the contractions in the article using 'expand\_contractions'.

9.9. Return the preprocessed 'article' as output of the function.

10.Apply a lambda function to the 'article' series for removing possessive forms of words and reducing consecutive spaces.

10.1. Extends the preprocessing steps for the 'article' data by removing punctuation and stop words.

10.2. It applies lambda functions and regular expressions to achieve these modifications.

10.3. Store the result back in the ' article' series.

11. Analyze the frequency and score of sentences in an article using

'word\_frequency' & 'sentence\_score'.

12.Using 'summary(sentence\_score\_Owo)',it calculates the number of sentences to include in the summary by taking 25% of the total number of sentences. It selects the top-scoring sentences using the 'nlargest' function and appends them as a summary to the 'summary list'.

12.1.calculates the word frequencies using the ' word\_frequency' function.

12.2.calculates sentence scores using the 'sent\_token' function.

12.3.Generates the summarized article using the 'summary' function.

12.4.The last part of the code calls the 'article\_summarize' function with a subset of article dataframe.

12.5. Then it prints the actual length of the first article, displays the full article text, prints the length of the summarized article, displays the summarized article, and finally calls the 'word\_cloud' function to generate and display the word cloud visualization.

13.1) Import the necessary libraries:

13.1.1) 'newspaper'-library for scraping articles from websites.

13.1.2) 'reportlab'-library for generating PDF documents.

13.1.3) 'sumy'-library for text summarization.

13.2) Import specific modules from the libraries:

13. 2.1) 'article' class from 'newspaper' library to extract information from newspaper articles.

13.2.2) 'SimpleDocTemplate' , 'paragraph' and 'spacer' classes from 'reportlab.lib.styles' module for creating PDF documents.

13.2.3) 'getSampleStyleSheet' function from 'reportlab.lib.styles' module for defining document styles.

13. 2.4) 'plaintextParser', 'Tokenizer' & 'LsaSummarizer' classes from the 'sumy.parsers.plaintext' & 'sumy.nlp.tokenizers' & 'sumy.summarizers.lsa' modules respectively for text summarization.

13.3) Perform text summarization by initialising an newspaper article.

13.3.1) use a 'parse()' method to extract the article's information.

13.3.2) initialise a 'LsaSummariser' object.

13.3.3) use the 'analyze()' method of the 'LsaSummarizer' to compute the summary.

13.3.4) Get the summarized sentences using the 'get\_summary()' method.

13.4) Generate a PDF document by initialising 'SimpleDocTemplate' object with the desired output file name and page size.

13. 4.1) Create a list of paragraphs and spacers with the summarized sentences and styles.

13.4.2) Build the document using the 'build()' method of the 'SimpleDocTemplate' object.

14) End.