Documentation – News Analyzer Project

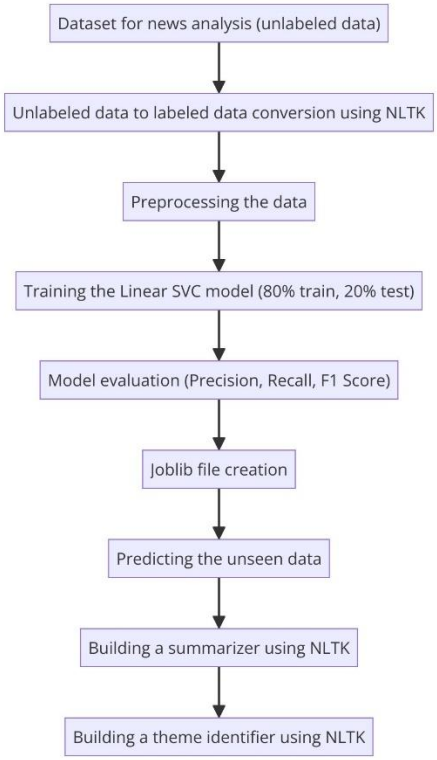
Shantanu Gagare – shantanugagare1998@gmail.com

1. **Problem Statement:**
   Objective:
   The objective of the News Analysis Project is to develop a comprehensive system capable of enhancing the comprehension and summarization of news articles through advanced natural language processing (NLP). The system aims to provide valuable insights into the emotional tone and thematic connections across various articles, thereby aiding stakeholders in making informed decisions based on current news trends.

2. **Challenges to Address:**

   1. Text Cleaning and Preprocessing: The first major challenge is the effective cleaning and preprocessing of news articles. This involves removing unnecessary elements such as punctuation and common stopwords that may obscure the main ideas within the text.
   2. Mood Detection: Another significant challenge is the development of a robust mood detection module that can accurately determine the sentiment (positive, negative, or neutral) of the articles. This module must handle various writing styles and contexts that can affect sentiment analysis.
   3. Thematic Analysis: Identifying and linking common themes across a multitude of articles presents a challenge due to the diversity of topics and perspectives in news writing. The system must be able to detect subtle patterns and themes that may not be immediately apparent.
   4. Integration and Performance: Integrating these modules into a cohesive system that performs efficiently on large datasets of news articles is crucial. The system must process information quickly and provide outputs such as mood ratings, concise summaries, and identified themes in a user-friendly manner.

3. Workflow:



4. Technologies and Libraries used:

- pandas: For data manipulation and reading/writing Excel files.
- nltk: Used for text processing, summarization, and handling stopwords.
- joblib: For loading your pre-trained models.
- scikit-learn: This is for TF-IDF vectorization and possibly other machine learning tasks if extended.
- Numpy: It is likely a dependency for Pandas and scikit-learn, but it is important for handling numerical operations.

```
pip install pandas==2.2.2
pip install nltk==3.8.1
pip install joblib==1.4.2
pip install scikit-learn==1.4.2
pip install numpy==1.26.4
```

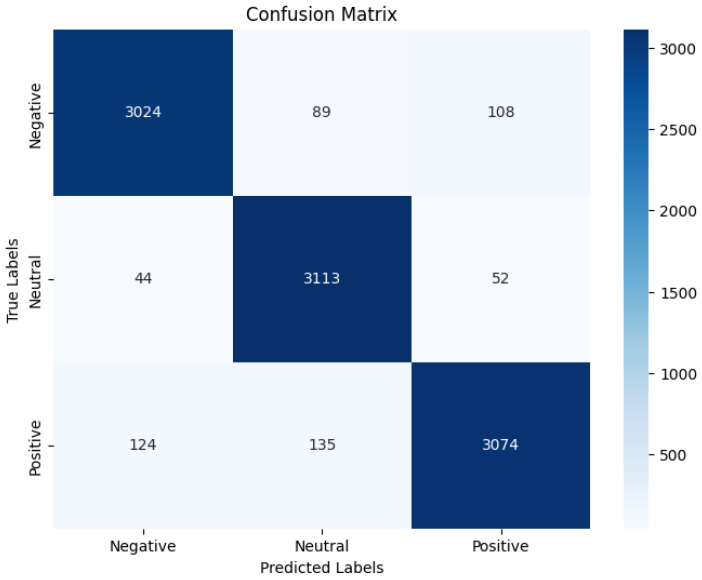
5. **Dataset:**
   Inshorts Dataset - English News - link
   Inshorts is a news service that provides short news summaries from around the web. This dataset contains headlines and a summary of news items. The dataset is English.

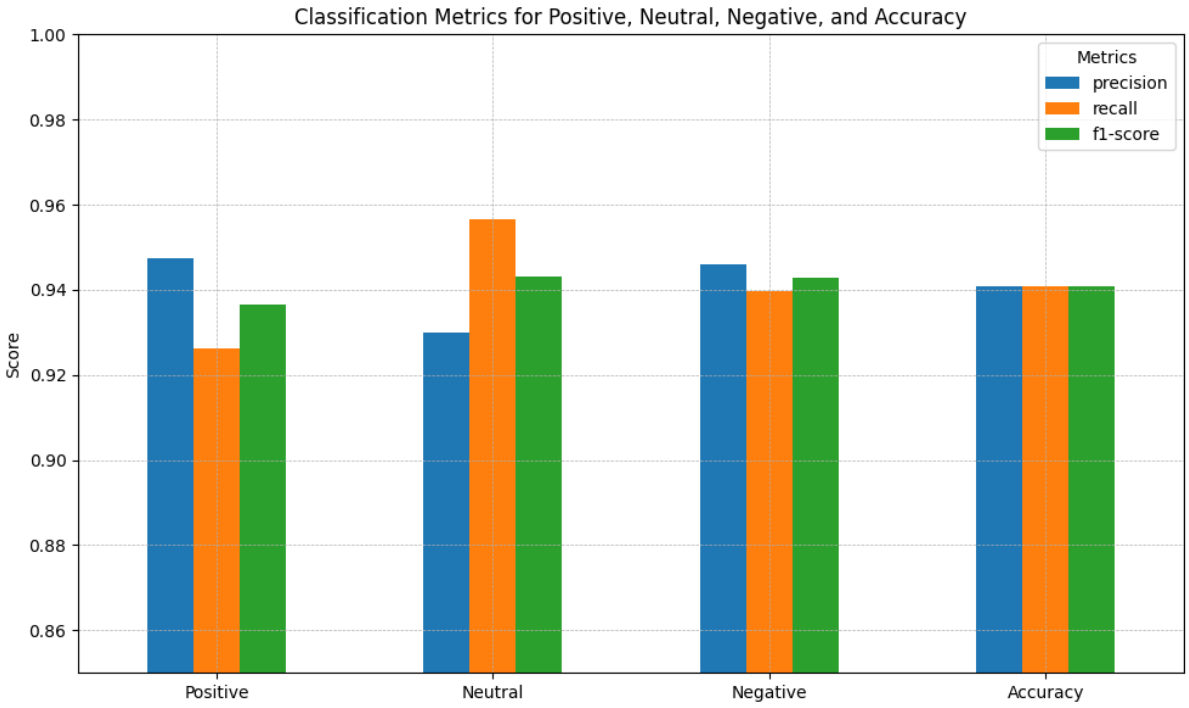
6. **Data Cleaning Step:**
   - Text Normalization: Standardizing text, such as converting all text to lowercase and expanding contractions, to reduce the variety of forms an algorithm must handle.
   - Tokenization: Breaking text down into smaller parts like words or sentences makes processing and analyzing easier.
   - Removing Stopwords: Filtering out common words (like "and", "the", etc.) that offer little value in understanding the text's meaning.
   - Stemming and Lemmatization: Simplifying words to their base or root form to reduce the complexity of the language data.
   - Handling Noise: Removing irrelevant characters, such as special symbols, numbers, or HTML tags, which do not contribute to text analysis.
   - Unicode Normalization: Ensuring all text is in a consistent format (like UTF-8) to avoid encoding issues.

7. **Sentiment Analysis (Labeled data generation from unlabeled data):**
   - Importing Libraries:
   - nltk: Natural Language Toolkit (NLTK) is used for text manipulation and provides support for tokenization and stopword handling.
   - string: This standard Python library provides a collection of string constants, including a string of all punctuation characters.
   - swifter: A library that optimizes Pandas operations by accelerating apply functions. It automatically decides whether to use a single-core, multi-core, or Dask (distributed computing) approach based on data size.
   - Downloading NLTK Resources:
   - punkt: A tokenizer model used to divide a text into a list of sentences by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences.

- stopwords: A collection of "stop words" which are generally the most common words in a language and often do not contribute significant meaning in text analysis.
- Preparing Stopwords and Punctuation for Removal:
- Converts the list of stopwords from NLTK and all punctuation characters into sets for fast access during the cleaning process.
- Defining the Text Cleaning Function:
- Lowercase Conversion: Standardizes the text to lowercase to ensure uniformity, reducing the complexity for downstream processing.
- Tokenization: Splits the text into individual words, making them easier to analyze and manipulate.
- Removing Punctuation and Stopwords: Filters out unwanted tokens that are either punctuation or recognized stopwords, which are not typically useful in many NLP tasks.
- Rejoining Tokens: Converts the list of clean tokens back into a single string for further analysis or modeling.
- Applying Text Cleaning in DataFrame:
- Uses swifter to apply the clean_text function across a DataFrame column efficiently. This approach is designed to handle large datasets by utilizing multi-core processing.
- Sentiment Analysis Application (Assumed Functionality):
- It's noted that the cleaned text is then used for sentiment analysis, although the specific implementation details of get_sentiment are not provided. This function is assumed to analyze the sentiment of the text and categorize it accordingly.

8. Model Evaluation:





9. **Article Summarization and Theme Finding**:

Text Summarization (nltk_summarize):

- This function takes a piece of text as input and generates a summary of it.
- It first tokenizes the text into words and removes stopwords (commonly occurring words like "the", "is", etc.) and punctuation marks.
- Using NLTK's FreqDist, it calculates the frequency of each word in the text.

- For each sentence in the text, it calculates a score based on the sum of frequencies of words present in that sentence.
- Finally, it selects the top N sentences with the highest scores to form the summary.
- This method essentially identifies the most important sentences in the text based on the frequency of important words.

Theme Finding (get_relevant_category):

- This function aims to categorize the text into predefined themes based on the occurrence of specific keywords associated with each theme.
- It takes the input text and a dictionary of categories with their associated keywords and weights.
- It tokenizes the text into words and removes stopwords.
- For each category, it calculates a score by summing the product of the frequency of each keyword in the text and its weight.
- The category with the highest score is considered the most relevant category for the text.
- This method essentially matches the text with the predefined themes based on the presence and importance of specific keywords.

10. Complete Pipeline:

Input:

Excel file containing news articles.

Pipeline Process:

Text Preprocessing:
Removal of stopwords, punctuation, and stemming.
Text Summarization:
Generation of a summary for each article.
Sentiment Analysis:
Prediction of sentiment (positive, negative, neutral) for each article.
Theme Categorization:
Classify articles into predefined themes based on keywords.

Output:

Excel file ('final_processed_articles.xlsx') with the following columns for each article:

'Article': Original news article.
'Summary': Summarized version of the article.
'Sentiment': Predicted sentiment (positive, negative, neutral).
'Theme': Categorized theme based on the content of the article.

11. How to run the code?

Install Required Libraries:

Ensure you have Python installed on your system.

Install the required libraries by running:

pip install pandas joblib nltk scikit-learn

Additionally, download NLTK resources by running the following Python script:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

Prepare Input Data:

Create an Excel file containing news articles. Each article should be stored in a separate row under the' Article' column. Save this file with an appropriate name (e.g., 'news_articles.xlsx').

Run the Code:

Essential Files:

- svm_sentiment_model.joblib: This file contains the trained Support Vector Machine (SVM) model for sentiment analysis.

- tfidf_vectorizer.joblib: This file contains the TF-IDF vectorizer used to transform text data for input into the SVM model.

Please make sure that both svm_sentiment_model.joblib and tfidf_vectorizer.joblib are included in the project directory or in the specified path to enable successful execution of the final code.

Ensure that the provided code is saved in a Python script (e.g., 'news_processing_pipeline.py').

Update the file path in the process_articles() function to point to your input Excel file:

process_articles('news_articles.xlsx')

Run the script using Python:

python news_processing_pipeline.py

Check Output:

After the script finishes execution, it will generate an Excel file named 'final_processed_articles.xlsx'.

Open this file to view the processed news articles and their summaries, sentiment predictions, and categorized themes.