MINOR-1

Bidirectional RNN (Model)

- Bidirectional recurrent neural network (BRNN) is made to process sequential data.
- Recurrent Neural Networks (RNNs) are a particular class of neural networks that was created with the express purpose of *processing sequential input, including speech, text, and time series data.*
- Bidirectional Recurrent Neural Network (Bi-RNN) refers to the capability of processing input data in both forward and backward directions.
- More details at-(https://www.geeksforgeeks.org/bidirectional-recurrent-neural-network/).

Algorithm (LSTM & GRU):

- **LSTM Long Short-Term Memory** a type of recurrent neural network (RNN) architecture that processes input data in both forward and backward directions.
- LSTM recurrent unit tries to “remember” all the past knowledge that the network is seen so far and to “forget” irrelevant data.
- More details at-(https://www.geeksforgeeks.org/long-short-term-memory-networks-explanation/).
- **Gated Recurrent Unit (GRU)** is a type of recurrent neural network (RNN) architecture that models sequential data. GRUs are faster than LSTMs, but LSTMs are more accurate for longer sequences.
- More details at -(https://www.geeksforgeeks.org/gated-recurrent-unit-networks/).

1. **Data Loading and Preprocessing:**
   - You read a CSV file containing news data (`WELFake_Dataset.csv`) into a Pandas DataFrame.
   - Merged the "title" and "text" columns into a new column called "tot_news."
   - Extracted labels and news for fake and real news separately.
2. **Tokenization and Model Loading:**
   - Tokenized the combined news text using the Keras `Tokenizer` class.
   - Loaded a pre-trained Keras model (`test4.keras`) using `load_model` from Keras.
3. **News API Integration:**
   - Utilized the News API (`newsapi-python`) to get news articles based on search keywords.
   - Extracted the URL and title of the top news article.
4. **Web Scraping and Text Analysis:**
   - Scraped the text content of the news article using BeautifulSoup.

- Analyzed sentiment using TextBlob.
- Classified topics using spaCy.

5. **Visualization:**
   - Created a word cloud based on the classified topics using the `WordCloud` library.
   - Displayed the word cloud using Matplotlib.
6. **Prediction and Classification:**
   - Preprocessed the new text by tokenizing and padding it.
   - Used the pre-trained model to predict whether the news article is fake or real.
   - Mapped the predicted label to the corresponding class ("fake" or "real").
7. **Function for Text Classification:**
   - Defined a function (`classify_a_text`) for classifying a given text using the loaded model.
8. **Setup and Imports:**

- **Checks Python version:** Ensures compatibility.
- **Installs libraries:** `newsapi-python` for news fetching, `spacy` for language processing.
- **Imports necessary libraries:**
  - `keras` for deep learning model loading.
  - `text` and `sequence` from `keras.preprocessing` for text preparation.
  - `numpy` for numerical operations.
  - `NewsApiClient` for interacting with News API.
  - `requests` for making HTTP requests.
  - `BeautifulSoup` for parsing HTML content.
  - `TextBlob` for sentiment analysis.
  - `WordCloud` for generating word clouds.
  - `spacy` for topic classification.
  - `matplotlib.pyplot` for creating visualizations.
  - `pandas` for data manipulation.
  - `train_test_split` from `sklearn.model_selection` for dataset splitting.

9. **Data Loading and Preprocessing:**

- **Loads dataset:** Reads a CSV file containing news articles and their labels (fake/real).
- **Creates a mapper:** Maps numerical labels to "fake" and "real" for readability.
- **Combines title and text:** Merges these columns for analysis.
- **Splits data into training and testing sets:** Prepares data for model training and evaluation.
- **Creates a tokenizer:** Learns vocabulary and converts text to numerical sequences.
- **Loads a pre-trained fake news detection model:** Loads a model from a file for prediction.

10. **News Fetching Functions:**

- `get_news_url`:
  - Takes search keywords as input.
    - Uses News API to fetch relevant news articles.
    - Returns the URL and title of the top article or a dummy URL if no results are found.
- `get_news_text`:
  - Takes a news article URL as input.
    - Fetches the HTML content using `requests`.
    - Parses the HTML using `BeautifulSoup` to extract the article text.
    - Returns the extracted text.

11. **Text Analysis Functions:**

- `analyze_sentiment`:
  - Takes text as input.
    - Uses `TextBlob` to analyze its sentiment (positive, negative, or neutral).
    - Returns the sentiment score.
    - `classify_topic`:
      - Takes text as input.
    - Uses `spaCy` to identify nouns as potential topics.
    - Returns a list of extracted topics.

12. **Main Script:**

- **Prompts for search keywords:** Asks the user for input.
- **Fetches news article details:** Uses `get_news_url` and `get_news_text` to retrieve URL, title, and text.
- **Performs sentiment analysis:** Uses `analyze_sentiment` to determine sentiment.
- **Classifies topics:** Uses `classify_topic` to identify topics.
- **Displays results:** Prints article title, link, sentiment, topics, and text.
- **Generates word cloud:** Visualizes topics using `WordCloud`.
- **Preprocesses text for prediction:** Converts text to numerical sequences using the tokenizer.
- **Predicts label using the model:** Classifies the article as "fake" or "real".
- **Prints predicted label:** Displays the model's prediction.

6. **Additional Function for Text Classification:**

- `classify_a_text`:
  - Takes arbitrary text as input.
  - Preprocesses the text.
  - Uses the model to predict its label ("fake" or "real").

- Prints the predicted label.

Project Link-(https://www.kaggle.com/code/ganeshtailuru/bidirectional-rnn-news-summary-analysis).