Folder Structure

Pdfs\_new

BOE\_Scraping\_Automation.py

BOE

BOE\_summary\_prediction.py

Summary\_model.py

word cloud.py

common

Cleaning\_summary\_.py

Fed\_Scraping\_Automation.py



Ecb\_csv\_Download.py

ECB

BOE\_word\_cloud.py

Fed\_summary\_prediction.py

Ecb\_ word\_cloud.py

FED

Run\_selenium.py

prediction.py

Ecb\_ summary\_prediction.py

Ecb\_PDF\_Link\_scraping.py

fed\_ word\_cloud.py

Folder Structure

best-checkpoint (1).ckpt

model

ALL\_FED\_SPEECH\_SUMMARY\_DATA.xlsx

ALL\_ECB\_SPEECH\_SUMMARY\_DATA .xlsx

ALL\_BOE\_SPEECH\_SUMMARY\_DATA.xlsx

BoeWordCloud

wordcloud

fed\_SpeechData.xlsx

last\_update\_date\_for\_wordcloud.xlsx

FedWordCloud

EcbWordCloud

ECB\_SpeechData.csv

ECB\_PDF\_Link.xlsx

BOE\_SpeechData.xlsx

Upload\_to\_s3.py

Summary\_prediction

Scraped\_Data

S3

Send\_s3\_link.py

## ***BACKEND CODE WORK THROUGH***

1. **BOE :**
   1. **Boe\_Scraping\_Automation.py**
      1. This is a Python class named Boe\_Scraping, designed to scrape speeches from the Bank of England website.
      2. The **\_\_init\_\_** method initializes the class by setting instance variables such as the URL of the Bank of England speeches page, a Selenium object for running a web driver, and a filename for the scraped data
      3. The **run\_boe\_scraping** method is responsible for web scraping the BOE (Bank of England) speeches and generating an Excel file with the data obtained from the web scraping. The method scrapes the speeches, extracts their details, and writes them to an Excel file
   2. **Boe\_summary\_prediction.py**
      1. This is a Python class called **boe\_Summary\_Prediction** which is used to predict quotes from speech contents in an Excel file called "ALL\_BOE\_SPEECH\_SUMMARY\_DATA.xlsx". The class has three instance variables: **filename**, **boe\_scrapedata\_path**, and **boe\_summarydata\_path** which are set to file paths for the **ALL\_BOE\_SPEECH\_SUMMARY\_DATA.xlsx**,**BOE\_SpeechData.xlsx**, and **ALL\_BOE\_SPEECH\_SUMMARY\_DATA.xlsx** files, respectively.
      2. It reads data from the "boe\_summarydata\_path" and "boe\_scrapedata\_path" Excel files and extracts the latest data from the scraped data. It uses a machine learning model to predict the quotes and then cleans the data before appending it to the summary file.
   3. **Boe\_word\_cloud.py**
      1. This is a Python class named "boe\_word\_cloud" that generates word cloud images for a new generation of quotes in the "wordcloud/BoeWordCloud/" directory.
      2. The class has a method named "run\_boe\_wordcloud" that reads new quote predictions from the "summary\_prediction/ALL\_BOE\_SPEECH\_SUMMARY\_DATA.xlsx" file and generates word clouds using the "Word\_cloud\_generation" class.
      3. The output of this class is a set of word cloud images saved in the "wordcloud/BoeWordCloud/" directory. The class also has instance variables that store the path to the summary file.
2. **ECB :**
   1. **Ecb\_csv\_Download.py**
      1. This is a Python class named **Ecb\_Scraping** that defines web scraping functionality for the European Central Bank (ECB) website. The class has an **\_\_init\_\_** method that initializes a **Selenium\_Run** object to launch the Chrome driver and load the ECB download link.
      2. The **download\_csv** method downloads the ECB\_SpeechData.csv file from the ECB Archive by finding the download link using Selenium and then saving the file to the "Scraped\_Data" directory.
   2. **Ecb\_PDF\_link\_scraping.py**
      1. This is a Python class called **Ecb\_PDF\_Scraping** used to scrape the European Central Bank (ECB) website for PDF files containing speeches or other key information. The class uses the Selenium web driver to automate browsing the ECB website and extract links to PDF files. The extracted links are then saved in an Excel file located in the "Scraped\_Data" folder.
      2. This is the class constructor method. It initializes the instance variables of the class, including the ECB website link, the Selenium web driver, and the output Excel file path.
   3. **Ecb\_summary\_prediction.py**
      1. This script defines a class named Ecb\_Summary\_Prediction, which is used to generate quote predictions from speech contents in a specified Excel file. The class has an attribute "filename" that represents the path to the Excel file that contains the summary data.
      2. The class has a method named "summary\_prediction" that reads the summary data from the Excel file, reads speech data from a CSV file, and performs some data cleaning operations on the speech data. It then passes the cleaned data to a model\_prediction object to generate predictions and uses a Prediction\_Summary\_Cleaning object to clean the resulting predictions. Finally, it concatenates the cleaned predictions with the summary data and writes the concatenated data to the Excel file specified by the "filename" attribute.
   4. **Ecb\_word\_cloud.py**
      1. The **ecb\_word\_cloud** class is designed to generate and save word cloud images for the new quotes in the **ALL\_ECB\_SPEECH\_SUMMARY\_DATA.xlsx** file. The class contains a single method named **run\_ecb\_wordcloud()** which creates an instance of the **Word\_cloud\_generation** class and calls the **word\_cloud()** method to generate the word cloud image.
      2. The **run\_ecb\_wordcloud()** method takes no arguments and the path to the summary data file is set as a class attribute. The word cloud image is saved in the **wordcloud/EcbWordCloud/** directory. If an exception occurs during the execution of the method, it is printed to the console.
3. **FED :**
   1. **Fed\_Scraping\_Automation.py**
      1. This is a Python class named **Fed\_Scraping** used to scrape new speeches from the website of the Federal Reserve Bank of the United States, and update the previously scraped data in the file fed\_SpeechData.xlsx.
      2. **‘\_\_init\_\_**’: Initializes the class by setting the URL for the Federal Reserve Bank website, creating a Selenium web driver object, and setting the file name for the output data file fed\_SpeechData.xlsx.
      3. **run\_fed\_scraping**: This method automates the process of scraping new speeches from the Federal Reserve Bank website, and updating the previously scraped data in the output data file fed\_SpeechData.xlsx
   2. **Fed\_summary\_prediction.py**
      1. This script defines a class **Fed\_Summary\_Prediction** which contains a method **summary\_prediction()** that updates quote predictions from speech contents in a given excel file named **ALL\_FED\_SPEECH\_SUMMARY\_DATA.xlsx**.
      2. The **summary\_prediction()** method first reads in the data from the **fed\_SpeechData.xlsx** and **ALL\_FED\_SPEECH\_SUMMARY\_DATA.xlsx** files using **pandas.read\_excel()**. It then converts the 'date' column in the **fed\_summary\_data** DataFrame to a datetime object.
      3. Finally, the cleaned DataFrame is concatenated with the original **fed\_summary\_data** DataFrame using **pandas.concat()**, and the resulting DataFrame is written to the **ALL\_FED\_SPEECH\_SUMMARY\_DATA.xlsx** file using **pandas.to\_excel()**
   3. **Fed\_word\_cloud.py**
      1. This is a class named **fed\_word\_cloud** which has a single method called **run\_fed\_wordcloud()**. The purpose of this method is to generate a word cloud of the new quotes predicted in the **summary\_prediction/ALL\_FED\_SPEECH\_SUMMARY\_DATA.xlsx** file and save it to the **wordcloud/FedWordCloud** folder
      2. The class imports **Word\_cloud\_generation** from **common. Word\_cloud**. The method **run\_fed\_wordcloud()** instantiates an object of **Word\_cloud\_generation** and calls the **word\_cloud()** method. The **word\_cloud()** method generates a word cloud from the given excel file and saves the image file to the specified folder.
4. **Common :**
   1. **cleaning\_summary.py**
      1. The **Prediction\_Summary\_Cleaning** class is responsible for cleaning and formatting Quote prediction by adding double quotation marks to each quote tag.
      2. The class has a single method named **clean\_summary** which takes a dataframe **data** as input and returns the cleaned dataframe.
   2. **Prediction.py**
      1. The **model\_prediction** class is used to load a custom-trained model and perform summarization tasks for speech text. The class has three methods: **\_\_init\_\_**, **load\_model**, **summerize**, and **prediction**
      2. The **\_\_init\_\_** method initializes the **MODEL\_NAME** and **tokenizer**. The **load\_model** method loads the pre-trained model from the model folder. The **summerize** method performs the summarization task after generating encodings and prepares a summary
      3. The **prediction** method loads the **summerize** method and does the prediction of the speech text getting from the dataframe. The **speech\_text\_col** variable is used to get the last column of the data.
      4. The for loop is used to iterate through the data index and get the summary of the speech text, which is then stored in the **quotes** column. Finally, the method returns the updated data.
   3. **Run\_selenium.py**
      1. This is a Python class named **Selenium\_Run**, which is used to start the Selenium Chrome driver for web scraping. It imports several modules from the **selenium** package, such as **webdriver**, **Keys**, **By**, **expected\_conditions**, **Select**, and **ActionChains**, as well as **time**, **Options**, **which**, and **ChromeDriverManager** from other packages.
      2. The **run\_Fed\_driver\_chrome** method starts the Chrome driver with several options and opens the website using the provided **link**. It then waits for 6 seconds before refreshing the page and returning the driver object.
   4. **Summary\_model.py**
      1. This code defines a PyTorch Lightning module called **SpeechSummaryModel**, which is used to train a T5 model for speech summarization.
      2. The **\_\_init\_\_** method initializes the T5 model and tokenizer and defines the **MODEL\_NAME** to use the **t5-large** model.
      3. The **forward** method defines the forward pass of the model, taking **input\_ids**, **attention\_mask**, **decoder\_attention\_mask**, and **labels** as input and returning **loss** and **logits.**
      4. The **training\_step**, **validation\_step**, and **test\_step** methods define the steps for training, validation, and testing respectively. Each method takes **batch** and **batch\_idx** as input and performs the forward pass of the model, computes the loss, logs the loss value, and returns the loss.
      5. Finally, the **configure\_optimizers** method defines the optimizer used for training the model, which is AdamW with a learning rate of 0.0001.
   5. **Word\_cloud.py**
      1. The **color\_func** method defines the color of each keyword in the word cloud based on its sentiment.
      2. The **word\_cloud** method generates the word cloud using the data from the Excel file and saves the generated image as a png file. The method checks the date of the data in the Excel file against the last date for which the word cloud was generated and generates the word cloud only for the new data. It then updates the last date of the word cloud generation in the Excel file. If there is an error during the process, the method prints the error message.
5. **Model :**
   1. **best-checkpoint (1).ckpt –** This is the custom trained weights and biases saved in the **best-checkpoint (1).ckpt file.**
6. **S3 :**
   1. **Send\_s3\_link.py**
      1. This file is used to send all summary prediction files links in S3 bucket.
   2. **Upload\_to\_s3.py**
      1. This file is used to upload all zip images in S3 bucket.
7. **Scraped\_Data :**
   1. This Folder contains all scraped info of all 3 banks.
8. **Summary\_prediction**:
   1. This folder contains all Quote Prediction excel files from all banks
9. **Wordcloud :**

a. It contains all png files of word clouds of each banks and last update excel file where each last date is being updated when running the wordcloud python files.

Run\_selenium.py

Summary\_model.py

word cloud.py

common

prediction.py

Cleaning\_summary\_.py

fed\_ word\_cloud.py

Fed\_summary\_predictionpy

Fed\_Scraping\_Automation.py



FED