Drew Lickman

CSCI 4820-001

Project #7

Due 12/3/24

AI Disclaimer: A.I. Disclaimer: Work for this assignment was completed with the aid of artificial intelligence tools and comprehensive documentation of the names of, input provided to, and output obtained from, these tools is included as part of my assignment submission.

Using SIF /scratch/user/u.jp60244/sif/csci-2024-Spring.sif

Custom NLP Project using 3 Hugging Face Pipelines

Dr. Sal Barbosa, Department of Computer Science, Middle Tennessee State University

---

Project Description

This project is used to analyze the transcripts of the Federal Open Market Committees (FOMC)

Takes about 30 minutes to run the entire program

The Problem:

I chose this project because I believe it is important for people to get a quick and easy-to-understand analysis of the FOMC meetings. The FOMC "reviews economic and financial conditions, determines the appropriate stance of monetary policy, and assesses the risks to its long-run goals of price stability and sustainable economic growth" (https://www.federalreserve.gov/monetarypolicy/fomc.htm)

The Dataset:

The dataset I used is the FOMC transcripts from each of their meetings. I created (with Claude 3.5 Sonnet (New)) a web scraper to read the FOMC website and download the PDFs

The Solution:

1. Download PDF transcripts from the official FOMC website using `fomc-crawler.py`
2. Convert the PDFs to text files with `pdf-to-txt.py`
3. Utilize a slightly modified version of tabularisai's robust-sentiment-analysis (distil)BERT-based Sentiment Classification Model `https://huggingface.co/tabularisai/robust-sentiment-analysis` for sentiment analysis
4. Summarize each document via pipeline of Falconsai's text_summarization Fine-Tuned T5 Small for Text Summarization Model `https://huggingface.co/Falconsai/text_summarization`
5. Answer the question "What is the current status of the economy?" from each meeting by using consciousAI's question-answering-roberta-base-s-v2 for Question Answering `https://huggingface.co/consciousAI/question-answering-roberta-base-s-v2`

Load web proxy for TAMU FASTER system

In [1]:
```
import os
os.environ['http_proxy'] = 'http://10.72.8.25:8080'
os.environ['https_proxy'] = 'http://10.72.8.25:8080'
```

The following pip installs may be necessary to run the web scraper and pdf-to-text converter:

In [2]:
```
# For Web Scraper and PDF-to-TXT:
!pip install requests tqdm beautifulsoup4
!pip install pdfplumber
!pip install tqdm
```

```
!pip install datasets

# If you encounter an error, you may not have Windows Long Path support enabled.
# You can find information on how to enable this at https://pip.pypa.io/warnings/enable-long-paths
!pip install transformers
!pip install nbformat=4.2.0
!pip install ipywidgets
```

In [3]:
```
import os
import re
import nltk
import torch
import numpy as np
import torch.nn as nn
import torch.optim as optim
from datasets import load_dataset
import plotly.graph_objects as go
from   nltk.tokenize import sent_tokenize
from   transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline

#nltk.download('punkt') # comment after downloading
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
2024-11-27 12:13:37.914941: I tensorflow/core/util/port.cc:111] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-27 12:13:39.022237: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2024-11-27 12:13:39.022278: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2024-11-27 12:13:39.034101: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2024-11-27 12:13:39.580805: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-11-27 12:13:41.482962: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```

Web Scraping

To retrieve fresh data, you must run `./data/fomc-crawler.py` and `./data/pdf-to-txt.py` to download all the FOMC transcript PDFs first, then convert the PDFs to TXT

Scrape FOMC Transcripts from https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm

Please wait about 1 to 3 minutes

Code written by Claude 3.5 Sonnet (New)

In [4]:
```
!python ./data/fomc-crawler.py
# Outputs to ./data/fomc_transcripts
```

Conversion

Convert PDFs to TXT

Please wait 1 to 3 minutes

Code written by Claude 3.5 Sonnet (New)

In [5]:
```
!python ./data/pdf-to-txt.py
# Outputs to ./data/extracted_text
```

In [6]:
```
# Data directory
TEXT_DIR = "./data/extracted_text" # Local FOMC transcript data as .txt

# Summary directory
SUMMARY_DIR = "./data/summaries"

#  Save text files and their data to a dictionary
txt_fileNames = [txt for txt in os.listdir(TEXT_DIR) if txt.endswith('.txt')]
```

```
txt_data = [open(os.path.join(TEXT_DIR, file), 'r', encoding='utf-8').read() for file in txt_fileNames]

textDict = {fileName: data for fileName, data in zip(txt_fileNames, txt_data)}

print(f"{len(txt_fileNames)} documents ready for analysis!")

# If I had more time to fix up the code to get it using datasets I would use this
# From https://www.youtube.com/watch?v=en0bIMzyaE4
# transcripts = []
# for t in textDict:
#     transcripts.append({
#         'title': t,
#         'body': textDict[t]
#     })
# import json
# def save_as_jsonl(data, filename):
#     with open(filename, "w") as f:
#         for transcript in data:
#             f.write(json.dumps(transcript) + "\n")
# save_as_jsonl(transcripts, "train.jsonl")
# data_files = {"train": "train.jsonl"}
# dataset = load_dataset("json", data_files=data_files)
# print(dataset)
```

```
46 documents ready for analysis!
```

Below is a helper function that splits input text into chunks due to limited context sizes of the semantic analyzer and summarizer.

Written by Claude 3.5 Sonnet (New)

In [7]:
```
def chunk_text(text, max_chunk_size):
    """
    Split text into chunks based on sentences to respect max token limit.
    Tries to keep sentences together while staying under the token limit.
    """
    sentences = sent_tokenize(text)
    chunks = []
    current_chunk = []
    current_length = 0

    for sentence in sentences:
        # Rough approximation of tokens (words + punctuation)
        sentence_length = len(sentence.split())

        if current_length + sentence_length > max_chunk_size:
            if current_chunk:  # Save current chunk if it exists
                chunks.append(' '.join(current_chunk))
                current_chunk = [sentence]
                current_length = sentence_length
            else:  # Handle case where single sentence exceeds max_chunk_size
                chunks.append(sentence)
                current_chunk = []
                current_length = 0
        else:
            current_chunk.append(sentence)
            current_length += sentence_length

    # Add the last chunk if it exists
    if current_chunk:
        chunks.append(' '.join(current_chunk))

    return chunks
```

BERT-based Sentiment Analysis

tabularisai's robust-sentiment-analysis used via pipeline:

Modified to be chunked for longer input texts

also outputs probability distribution, rather than just the highest result

Please wait 2 to 4 minutes

In [8]:
```
model_name = "tabularisai/robust-sentiment-analysis"
```

```python
sentimentAnalysis = pipeline(model=model_name, device=device)
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)

# Pipeline from Hugging Face (copied from example on page, had to modify to get probability distribution)
def predict_sentiment(text):
        inputs = tokenizer(text.lower(), return_tensors="pt", truncation=True, padding=True, max_length=512)
        with torch.no_grad():
                outputs = model(**inputs)

        probabilities = torch.nn.functional.softmax(outputs.logits, dim=-1)
        predicted_class = torch.argmax(probabilities, dim=-1).item()

        probs_list = probabilities[0].tolist()
        sentiment_map = {0: "Very Negative", 1: "Negative", 2: "Neutral", 3: "Positive", 4: "Very Positive"}

        # Create a dictionary of sentiment labels and their probabilities
        sentiment_probs = {
                                                sentiment_map[i]: prob
                                                for i, prob in enumerate(probs_list)
                                                }

        return {
                        'predicted_class': sentiment_map[predicted_class],
                        'probabilities': sentiment_probs
                        }

# Function written by Claude 3.5 Sonnet (New) to allow the pipeline to handle longer input text
def analyze_long_text(text, max_chunk_size):
        """
        Analyze sentiment of long text by breaking it into chunks and averaging results.
        """
        # Clean text
        text = text.replace('\n', ' ').strip()

        # Split into chunks using existing chunk_text function
        chunks = chunk_text(text, max_chunk_size)

        # Analyze each chunk
        chunk_sentiments = {"Very Negative": 0, "Negative": 0, "Neutral": 0, "Positive": 0, "Very Positive": 0}
        valid_chunks = 0

        for chunk in chunks:
                try:
                        result = predict_sentiment(chunk) # Uses modified pipeline
                        for sentiment, prob in result['probabilities'].items():
                                chunk_sentiments[sentiment] += prob
                        valid_chunks += 1
                except Exception as e:
                        print(f"Error processing chunk: {e}")
                        continue

        # Average the sentiments
        if valid_chunks > 0:
                for sentiment in chunk_sentiments:
                        chunk_sentiments[sentiment] /= valid_chunks

        # Determine overall sentiment
        max_sentiment = max(chunk_sentiments.items(), key=lambda x: x[1])

        return {
                        'predicted_class': max_sentiment[0],
                        'probabilities': chunk_sentiments
                        }

# Updated sentiment analysis loop
sentimentCount = {"Very Negative": 0, "Negative": 0, "Neutral": 0, "Positive": 0, "Very Positive": 0}
sentimentProbs = {"Very Negative": [], "Negative": [], "Neutral": [], "Positive": [], "Very Positive": []}
for txt in textDict:
    try:
        result = analyze_long_text(textDict[txt], max_chunk_size=256)
        print(f"File: {txt}")
        print(f"Predicted Sentiment: {result['predicted_class']}")
        print("Probability Distribution:")
        for sentiment, prob in result['probabilities'].items():
            print(f"  {sentiment}: {prob * 100:.2f}%")
            sentimentCount[sentiment] += prob           # Save the probability to get the averages
            sentimentProbs[sentiment].append(prob)      # Save each probability for each sentiment
        print()
    except Exception as e:
        print(f"Error processing {txt}: {e}")
```

```
/home/u.al234966/.local/lib/python3.11/site-packages/huggingface_hub/file_download.py:797: FutureWarning: `resum
e_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you wa
nt to force a new download, use `force_download=True`.
  warnings.warn(
File: FOMCpresconf20240501.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.30%
  Negative: 8.90%
  Neutral: 59.33%
  Positive: 19.83%
  Very Positive: 8.65%

File: FOMCpresconf20240612.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.33%
  Negative: 9.23%
  Neutral: 55.40%
  Positive: 22.44%
  Very Positive: 9.60%

File: FOMCpresconf20221102.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.11%
  Negative: 8.03%
  Neutral: 54.66%
  Positive: 22.30%
  Very Positive: 11.91%

File: FOMCpresconf20230503.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.14%
  Negative: 4.35%
  Neutral: 65.01%
  Positive: 20.28%
  Very Positive: 9.23%

File: FOMCpresconf20231213.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.82%
  Negative: 6.82%
  Neutral: 65.97%
  Positive: 16.23%
  Very Positive: 9.16%

File: FOMCpresconf20190501.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.88%
  Negative: 8.70%
  Neutral: 65.71%
  Positive: 17.21%
  Very Positive: 6.50%

File: FOMCpresconf20210428.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.30%
  Negative: 12.64%
  Neutral: 60.98%
  Positive: 14.92%
  Very Positive: 7.16%

File: FOMCpresconf20201105.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.97%
  Negative: 4.81%
  Neutral: 62.29%
  Positive: 19.50%
  Very Positive: 11.43%

File: FOMCpresconf20220504.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.98%
  Negative: 7.73%
  Neutral: 62.20%
  Positive: 15.82%
```

```
  Very Positive: 11.26%

File: FOMCpresconf20230920.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.68%
  Negative: 6.46%
  Neutral: 63.06%
  Positive: 19.50%
  Very Positive: 9.31%

File: FOMCpresconf20211103.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.85%
  Negative: 5.92%
  Neutral: 67.91%
  Positive: 16.17%
  Very Positive: 8.15%

File: FOMCpresconf20230726.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.66%
  Negative: 7.09%
  Neutral: 55.68%
  Positive: 22.97%
  Very Positive: 10.59%

File: FOMCpresconf20220615.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.11%
  Negative: 7.17%
  Neutral: 55.43%
  Positive: 21.36%
  Very Positive: 12.94%

File: FOMCpresconf20220727.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.36%
  Negative: 14.77%
  Neutral: 59.01%
  Positive: 14.07%
  Very Positive: 7.79%

File: FOMCpresconf20190320.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.92%
  Negative: 8.50%
  Neutral: 56.45%
  Positive: 22.29%
  Very Positive: 8.84%

File: FOMCpresconf20191030.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.58%
  Negative: 5.67%
  Neutral: 56.67%
  Positive: 23.32%
  Very Positive: 12.76%

File: FOMCpresconf20230614.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.56%
  Negative: 5.52%
  Neutral: 64.05%
  Positive: 19.41%
  Very Positive: 9.46%

File: FOMCpresconf20190731.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.06%
  Negative: 8.54%
  Neutral: 64.95%
  Positive: 16.42%
  Very Positive: 8.03%
```

```
File: fomcpresconf20240731.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.28%
  Negative: 10.55%
  Neutral: 68.93%
  Positive: 11.31%
  Very Positive: 5.93%

File: FOMCpresconf20210317.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 5.64%
  Negative: 10.87%
  Neutral: 63.54%
  Positive: 13.64%
  Very Positive: 6.31%

File: FOMCpresconf20200429.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 12.30%
  Negative: 11.99%
  Neutral: 47.37%
  Positive: 18.27%
  Very Positive: 10.07%

File: FOMCpresconf20200916.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.35%
  Negative: 6.83%
  Neutral: 59.22%
  Positive: 18.53%
  Very Positive: 12.07%

File: FOMCpresconf20240131.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 0.65%
  Negative: 3.14%
  Neutral: 61.27%
  Positive: 22.47%
  Very Positive: 12.46%

File: FOMCpresconf20240918.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.09%
  Negative: 6.12%
  Neutral: 65.36%
  Positive: 18.67%
  Very Positive: 7.75%

File: FOMCpresconf20220921.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.73%
  Negative: 8.00%
  Neutral: 56.99%
  Positive: 20.75%
  Very Positive: 10.53%

File: FOMCpresconf20220316.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.16%
  Negative: 6.11%
  Neutral: 62.50%
  Positive: 18.67%
  Very Positive: 11.55%

File: FOMCpresconf20240320.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 0.81%
  Negative: 5.92%
  Neutral: 70.81%
  Positive: 16.30%
  Very Positive: 6.16%

File: FOMCpresconf20190918.txt
Predicted Sentiment: Neutral
```

```
Probability Distribution:
  Very Negative: 2.01%
  Negative: 10.05%
  Neutral: 59.31%
  Positive: 22.76%
  Very Positive: 5.87%

File: FOMCpresconf20200729.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 7.01%
  Negative: 14.82%
  Neutral: 55.49%
  Positive: 14.13%
  Very Positive: 8.55%

File: FOMCpresconf20210127.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.71%
  Negative: 10.74%
  Neutral: 46.84%
  Positive: 24.51%
  Very Positive: 13.21%

File: FOMCpresconf20231101.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.13%
  Negative: 7.71%
  Neutral: 60.20%
  Positive: 20.76%
  Very Positive: 9.19%

File: FOMCpresconf20230201.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.01%
  Negative: 5.38%
  Neutral: 60.66%
  Positive: 20.43%
  Very Positive: 10.52%

File: FOMCpresconf20190130.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.87%
  Negative: 10.58%
  Neutral: 63.98%
  Positive: 15.65%
  Very Positive: 6.92%

File: FOMCpresconf20241107.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.73%
  Negative: 7.32%
  Neutral: 71.08%
  Positive: 13.76%
  Very Positive: 5.11%

File: FOMCpresconf20190619.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.36%
  Negative: 5.72%
  Neutral: 66.87%
  Positive: 18.17%
  Very Positive: 7.88%

File: FOMCpresconf20221214.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.58%
  Negative: 9.78%
  Neutral: 52.16%
  Positive: 20.75%
  Very Positive: 12.73%

File: FOMCpresconf20220126.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.06%
```

```
  Negative: 9.45%
  Neutral: 58.48%
  Positive: 15.80%
  Very Positive: 12.21%

File: FOMCpresconf20201216.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.45%
  Negative: 6.85%
  Neutral: 54.78%
  Positive: 19.21%
  Very Positive: 16.71%

File: FOMCpresconf20200129.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.11%
  Negative: 5.43%
  Neutral: 67.35%
  Positive: 17.15%
  Very Positive: 7.96%

File: FOMCpresconf20200610.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 8.45%
  Negative: 11.16%
  Neutral: 43.96%
  Positive: 18.20%
  Very Positive: 18.23%

File: FOMCpresconf20211215.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.37%
  Negative: 9.68%
  Neutral: 58.18%
  Positive: 15.42%
  Very Positive: 12.35%

File: FOMCpresconf20191211.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.77%
  Negative: 9.13%
  Neutral: 63.96%
  Positive: 17.27%
  Very Positive: 7.88%

File: FOMCpresconf20230322.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 4.57%
  Negative: 7.33%
  Neutral: 58.42%
  Positive: 21.36%
  Very Positive: 8.32%

File: FOMCpresconf20210616.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 2.48%
  Negative: 8.46%
  Neutral: 65.81%
  Positive: 13.78%
  Very Positive: 9.47%

File: FOMCpresconf20210728.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 3.44%
  Negative: 9.73%
  Neutral: 67.96%
  Positive: 13.76%
  Very Positive: 5.11%

File: FOMCpresconf20210922.txt
Predicted Sentiment: Neutral
Probability Distribution:
  Very Negative: 1.44%
  Negative: 7.11%
  Neutral: 63.97%
```

```
Positive: 16.17%
Very Positive: 11.31%
```

In [9]:
```
# Print average sentiment confidence
avgSentimentPcts = []
for sentiment in sentimentCount:
        avgSentimentPcts.append(float(f"{sentimentCount[sentiment]/len(textDict) * 100:.2f}"))
        print(f"Average {sentiment}: \t{sentimentCount[sentiment]/len(textDict) * 100:.2f}%")
#print(avgSentimentPcts)
```

```
Average Very Negative:  3.18%
Average Negative:       8.19%
Average Neutral:        60.66%
Average Positive:       18.30%
Average Very Positive:  9.68%
```

In [10]:
```
# Data preparation
sentiments = ["Very Negative", "Negative", "Neutral", "Positive", "Very Positive"]
percentages = avgSentimentPcts
colors = ["#ff4d4d", "#ff8c8c", "#8c8c8c", "#7fbf7f", "#2eb82e"]

# Create the bar chart
barChart = go.Figure(data=[
    go.Bar(
        x=sentiments,
        y=percentages,
        marker_color=colors,
        text=[f'{p}%' for p in percentages],
        textposition='auto',
    )
])

barChart.update_layout(
    title='Average FOMC Sentiment Distribution',
    xaxis_title='Sentiment',
    yaxis_title='Percentage (%)',
    yaxis_range=[0, 100],
    template='plotly_white',
    bargap=0.2
)

barChart.show()

#####

# Create the line chart with 5 different lines for each sentiment
lineChart = go.Figure()

for i, sentiment in enumerate(sentiments):
    lineChart.add_scatter(
        x=list(range(len(sentimentProbs[sentiment]))),
        y=[p * 100 for p in sentimentProbs[sentiment]],
        mode='lines',
        name=sentiment,
        line=dict(color=colors[i])
    )

lineChart.update_layout(
    title='Sentiment Over Time',
    xaxis_title='Time',
    yaxis_title='Percentage (%)',
    template='plotly_white'
)

lineChart.show()
```

Summarization

Falconsai's text_summarization used via pipeline:

Modified to be chunked for longer input texts

Please wait 14 - 18 minutes

In [11]:
```
summarizer = pipeline(model="Falconsai/text_summarization", device=device)

# Function written by Claude 3.5 Sonnet (New) to allow the pipeline to handle longer input text
def summarize_long_text(text, summarizer, max_length_div, min_length_div, max_chunk_size):
    """
    Summarize long text by breaking it into chunks and combining summaries.
```

```python
    """
    # Clean text
    text = text.replace('\n', ' ').strip()

    # Split into chunks
    chunks = chunk_text(text, max_chunk_size)
    chunkLen = len(chunks)
    max_length = chunkLen // max_length_div
    min_length = chunkLen // min_length_div

    # Summarize each chunk
    chunk_summaries = []
    for chunk in chunks:
        try:
            result = summarizer(chunk, max_length=max_length, min_length=min_length) # Pipeline from Hugging Fa
            chunk_summaries.append(result[0]['summary_text'])
        except Exception as e:
            print(f"Error processing chunk: {e}")
            continue

    # Combine chunk summaries by appending them
    if len(chunks) == 1:
        return chunk_summaries[0]
    else:
        # For multiple chunks, append the summaries together
        combined_summary = ' '.join(chunk_summaries)
        return combined_summary

counter = 0
total = len(textDict)
for txt in textDict:
    try:
        length = len(textDict[txt])
        summary = summarize_long_text(
            text=textDict[txt],
            summarizer=summarizer,
            max_length_div=2,   # divisor of chunk
            min_length_div=4,   # divisor of chunk
            max_chunk_size=256  # Adjust based on model's token limit
        )
        if not os.path.exists(SUMMARY_DIR):
            os.makedirs(SUMMARY_DIR)
        with open(os.path.join(SUMMARY_DIR, txt), "w+") as summary_file:
            summary_file.write(f"File: {txt}\nSummary: {summary}\n")
            counter += 1
            print(f"{counter}/{total} files summarized.")
    except Exception as e:
        print(f"Error processing {txt}: {e}")
print(f"Finished outputting all summaries to ./data/summaries!")
```

```
/home/u.al234966/.local/lib/python3.11/site-packages/huggingface_hub/file_download.py:797: FutureWarning:

`resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If
you want to force a new download, use `force_download=True`.

/opt/conda/lib/python3.11/site-packages/transformers/pipelines/base.py:1101: UserWarning:

You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
```

```
1/46 files summarized.
2/46 files summarized.
3/46 files summarized.
4/46 files summarized.
5/46 files summarized.
6/46 files summarized.
7/46 files summarized.
8/46 files summarized.
9/46 files summarized.
10/46 files summarized.
11/46 files summarized.
12/46 files summarized.
13/46 files summarized.
14/46 files summarized.
15/46 files summarized.
16/46 files summarized.
17/46 files summarized.
18/46 files summarized.
19/46 files summarized.
20/46 files summarized.
21/46 files summarized.
22/46 files summarized.
23/46 files summarized.
24/46 files summarized.

25/46 files summarized.
26/46 files summarized.
27/46 files summarized.
28/46 files summarized.
29/46 files summarized.
30/46 files summarized.
31/46 files summarized.
32/46 files summarized.
33/46 files summarized.
34/46 files summarized.
35/46 files summarized.
36/46 files summarized.
37/46 files summarized.
38/46 files summarized.
39/46 files summarized.
40/46 files summarized.
41/46 files summarized.
42/46 files summarized.
43/46 files summarized.
44/46 files summarized.
45/46 files summarized.
46/46 files summarized.
Finished outputting all summaries to ./data/summaries!
```

List compression rate of summaries

Written by Claude 3.5 Sonnet (New)

Modified by myself

In [12]:
```python
# Get list of original and summary files
original_files = [f for f in os.listdir(TEXT_DIR) if f.endswith('.txt')]
summary_files = [f for f in os.listdir(SUMMARY_DIR) if f.endswith('.txt')]

# Initialize a list to store compression results
compression_results = []

# Function to read file content with error handling
def read_file_content(file_path):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            return file.read()
    except UnicodeDecodeError:
        with open(file_path, 'r', encoding='ISO-8859-1') as file:  # Fallback encoding
            return file.read()

# Compare lengths and calculate compression percentage
for original_file in original_files:
        original_path = os.path.join(TEXT_DIR, original_file)
        summary_path = os.path.join(SUMMARY_DIR, original_file)

        # Check if summary file exists
        if original_file in summary_files:
                original_content = read_file_content(original_path)
                summary_content = read_file_content(summary_path)
```

```
                original_length = len(original_content)
                summary_length = len(summary_content)

                # Calculate compression percentage
                compression_percent = ((original_length - summary_length) / original_length) * 100
                compression_results.append({
                        "file": original_file,
                        "original_length": original_length,
                        "summary_length": summary_length,
                        "compression_percent": compression_percent
                })

# Display results
for result in compression_results:
    print(f"{result['file']}: {result['original_length']} -> {result['summary_length']} (characters) | Compress
```

```
FOMCpresconf20240501.txt: 47836 -> 2299 (characters) | Compression: 95.19%
FOMCpresconf20240612.txt: 52102 -> 2691 (characters) | Compression: 94.84%
FOMCpresconf20221102.txt: 43266 -> 1879 (characters) | Compression: 95.66%
FOMCpresconf20230503.txt: 48230 -> 2300 (characters) | Compression: 95.23%
FOMCpresconf20231213.txt: 43891 -> 2027 (characters) | Compression: 95.38%
FOMCpresconf20190501.txt: 37670 -> 1424 (characters) | Compression: 96.22%
FOMCpresconf20210428.txt: 54392 -> 3302 (characters) | Compression: 93.93%
FOMCpresconf20201105.txt: 46599 -> 2344 (characters) | Compression: 94.97%
FOMCpresconf20220504.txt: 45055 -> 1958 (characters) | Compression: 95.65%
FOMCpresconf20230920.txt: 53597 -> 2869 (characters) | Compression: 94.65%
FOMCpresconf20211103.txt: 53542 -> 2991 (characters) | Compression: 94.41%
FOMCpresconf20230726.txt: 51493 -> 2781 (characters) | Compression: 94.60%
FOMCpresconf20220615.txt: 52615 -> 2941 (characters) | Compression: 94.41%
FOMCpresconf20220727.txt: 53449 -> 2977 (characters) | Compression: 94.43%
FOMCpresconf20190320.txt: 42111 -> 1827 (characters) | Compression: 95.66%
FOMCpresconf20191030.txt: 44764 -> 2017 (characters) | Compression: 95.49%
FOMCpresconf20230614.txt: 48843 -> 2297 (characters) | Compression: 95.30%
FOMCpresconf20190731.txt: 42756 -> 1916 (characters) | Compression: 95.52%
fomcpresconf20240731.txt: 46832 -> 2190 (characters) | Compression: 95.32%
FOMCpresconf20210317.txt: 57186 -> 3492 (characters) | Compression: 93.89%
FOMCpresconf20200429.txt: 44014 -> 1696 (characters) | Compression: 96.15%
FOMCpresconf20200916.txt: 60597 -> 3729 (characters) | Compression: 93.85%
FOMCpresconf20240131.txt: 50889 -> 2500 (characters) | Compression: 95.09%
FOMCpresconf20240918.txt: 48017 -> 2195 (characters) | Compression: 95.43%
FOMCpresconf20220921.txt: 39946 -> 1716 (characters) | Compression: 95.70%
FOMCpresconf20220316.txt: 49974 -> 2664 (characters) | Compression: 94.67%
FOMCpresconf20240320.txt: 44982 -> 2135 (characters) | Compression: 95.25%
FOMCpresconf20190918.txt: 49051 -> 2637 (characters) | Compression: 94.62%
FOMCpresconf20200729.txt: 54908 -> 2999 (characters) | Compression: 94.54%
FOMCpresconf20210127.txt: 52132 -> 2589 (characters) | Compression: 95.03%
FOMCpresconf20231101.txt: 50673 -> 2527 (characters) | Compression: 95.01%
FOMCpresconf20230201.txt: 43410 -> 1799 (characters) | Compression: 95.86%
FOMCpresconf20190130.txt: 44387 -> 2096 (characters) | Compression: 95.28%
FOMCpresconf20241107.txt: 41431 -> 1461 (characters) | Compression: 96.47%
FOMCpresconf20190619.txt: 40790 -> 1693 (characters) | Compression: 95.85%
FOMCpresconf20221214.txt: 43363 -> 1807 (characters) | Compression: 95.83%
FOMCpresconf20220126.txt: 52753 -> 2786 (characters) | Compression: 94.72%
FOMCpresconf20201216.txt: 56927 -> 3237 (characters) | Compression: 94.31%
FOMCpresconf20200129.txt: 52787 -> 2828 (characters) | Compression: 94.64%
FOMCpresconf20200610.txt: 56502 -> 3267 (characters) | Compression: 94.22%
FOMCpresconf20211215.txt: 59105 -> 3625 (characters) | Compression: 93.87%
FOMCpresconf20191211.txt: 50162 -> 2634 (characters) | Compression: 94.75%
FOMCpresconf20230322.txt: 43017 -> 1791 (characters) | Compression: 95.84%
FOMCpresconf20210616.txt: 57872 -> 3422 (characters) | Compression: 94.09%
FOMCpresconf20210728.txt: 53011 -> 2967 (characters) | Compression: 94.40%
FOMCpresconf20210922.txt: 51111 -> 2794 (characters) | Compression: 94.53%
```

Question Answering

consciousAI's question answering used via pipeline:

Ask a question to see how the FOMC's answer changes over time

Please wait 8 - 12 minutes

In [13]:

```
questAns = pipeline(model="consciousAI/question-answering-roberta-base-s-v2", device=device)

#question="What is the current status of the economy? "                #57.97%
#question="What is the future of the economy going to be? "            #44.17%
question="What is the current rate of inflation? "                             #89.51%         #useful questio
#question="What is the status of the stock market? "                   #25.63%
#question="What have been the main economic concerns lately? "  #65.54%
```

```
#question="What are the key decisions being made today? "               #51.64%
#question="What is the current federal funds rate? "                    #73.75%
#question="How long until the quantitative easing ends? "               #48.38%
#question="How much debt is the government in? "                                #12.43%         #useful questio
#question="How many Americans are unemployed? "                                 #66.61%
#question="What is the best news from this meeting? "                   #55.04%
#question="What time of day is it? "                                                    #78.1%          #non-us
#question="What color is my underwear? "                                                #16.05%         #non-us
question="What is the current rate of inflation? Only show me the exact number " #91.02%        #useful questio

print(question)
print()
scoreArray = []
for file in textDict:
    answer      = questAns(question=question, context=textDict[file])
    date        = file[12:20]
    year        = date[0:4]
    month       = date[4:6]
    day         = date[6:8]
    print(f"{month}/{day}/{year}: {round(answer['score'] * 100, 2)}%:\t", end="")
    scoreArray.append(answer['score'])
    answer = re.sub(r'\n', ' ', answer['answer'])
    print(f"{answer}")

npScoreArray = np.array(scoreArray)
mean = np.mean(npScoreArray)
variance = np.var(npScoreArray)
std_dev = np.std(npScoreArray)
std_err = std_dev/np.sqrt(len(npScoreArray))

print()
print(f"Mean (confidence): {mean:.2%}")
print(f"Standard Deviation: {std_dev:.2%}")
print(f"Variance: {variance:.2%}")
print(f"Standard Error: {std_err:.2%}")
```

What is the current rate of inflation? Only show me the exact number

```
05/01/2024: 97.66%:     2 percent
06/12/2024: 98.61%:     2 percent
11/02/2022: 90.51%:     2 percent
05/03/2023: 98.6%:      2 percent
12/13/2023: 97.02%:     3.7 percent
05/01/2019: 98.16%:     below 2 percent
04/28/2021: 94.71%:     below 2 percent
11/05/2020: 95.44%:     2 percent
05/04/2022: 92.31%:     2 percent
09/20/2023: 93.32%:     2 percent
11/03/2021: 47.81%:     2 percent
07/26/2023: 95.46%:     2 percent
06/15/2022: 98.56%:     2 percent
07/27/2022: 99.53%:     3.6
03/20/2019: 73.24%:     close to target
10/30/2019: 97.6%:      2.9 percent
06/14/2023: 96.18%:     2 percent
07/31/2019: 95.56%:     2 percent
07/31/2024: 94.82%:     2.5 percent
03/17/2021: 92.81%:     2 percent
04/29/2020: 23.84%:     in the third quarter
09/16/2020: 96.99%:     2 percent
01/31/2024: 96.6%:      1.9 percent
09/18/2024: 95.83%:     4.2 percent
09/21/2022: 97.25%:     2 percent
03/16/2022: 99.4%:      4.3 percent
03/20/2024: 95.53%:     2 percent
09/18/2019: 89.23%:     2 percent
07/29/2020: 61.17%:     well below our symmetric 2 percent
01/27/2021: 94.32%:     less than 2 percent
11/01/2023: 95.68%:     faithfully implement the statute
02/01/2023: 97.46%:     2 percent
01/30/2019: 74.11%:     between 2.25 and 2½
11/07/2024: 94.49%:     2 percent
06/19/2019: 92.05%:     2 percent
12/14/2022: 94.88%:     2 percent
01/26/2022: 96.76%:     2 percent
12/16/2020: 95.02%:     2 percent
01/29/2020: 95.05%:     2 percent
06/10/2020: 98.26%:     near zero
12/15/2021: 88.95%:     4.2 percent
12/11/2019: 87.5%:      2 percent
03/22/2023: 97.65%:     2 percent
06/16/2021: 97.2%:      8.4 percent
07/28/2021: 88.83%:     above 2 percent
09/22/2021: 95.11%:     2 percent

Mean (confidence): 91.02%
Standard Deviation: 13.97%
Variance: 1.95%
Standard Error: 2.06%
```