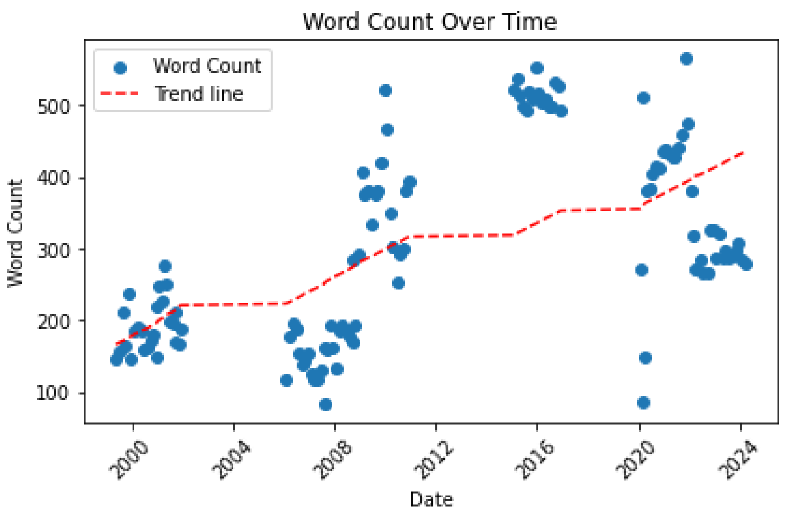
Why Watch the Fed? An Investigation Using Natural Language Processing and K-Means Clustering

Luke Heald, Jake Kerrigan, Sam Wood
Math 123: Mathematical Aspects of Data Analysis
Professor James Murphy

May 1, 2024

Introduction

The words spoken by the Chair of the Federal Reserve move markets, and for good reason. Changes in monetary policy in the United States not only ripple throughout domestic financial markets, as investors are looking to price Fed policy into their investment strategies, but also has large implications that are felt globally. Investors are increasingly leaning on the words of the Fed in the current era of high interest rates to curb inflation, as many are concerned with the future of their returns. This is due to the general economic malaise that often accompanies prolonged periods with higher interest rates. In addition, an increased commitment to transparency from the Fed beginning in the 1990s has led to ”as much of a focus on the Fed’s words as its actions as indications of future policy.”[1]. Dedications to increased transparency have also led to an increase in the length of FOMC statements, as shown in the figure below. It is thus reasonable to assume that more valuable information is embedded within these documents, in addition to it being a report containing current economic conditions and what the Fed sees as the economic outlook in the near term.



Recently, researchers looking to investigate the immediate effect of FOMC statements on financial markets has become more popular. Many papers have also examined the use of different natural language processors on how post-meeting sentiment can impact certain financial instruments. Additionally, other papers have attempted to quantify the relationship between the qualitative descriptors of the economy in FOMC statements with financial markets, such as seen in Doh, Kim, and Yang’s paper.[2]

The aim of this project was to take a wider approach, and see if clusters of FOMC statements could be learned more generally, and then see if certain components within

[1]Boukus, Ellyn, and Joshua V. Rosenberg. ”The Information Content of FOMC Minutes.” Federal Reserve Bank of New York, January 2005.

[2]Doh, Taeyoung, Sungil Kim, and Shu-Kuei Yang. ”How you say it matters: Text analysis of FOMC statements using natural language processing.” Economic Review-Federal Reserve Bank of Kansas City 106, no. 1 (2021): 25-40.

these statements were more powerfully associated with certain financial market conditions than others. We expected to see FOMC statements cluster across changes in interest rate policy–namely, when there was an interest rate cut, interest rate hold, or a raise in the interest rate. We believed this would be a good indicator of the clusters that were learned largely being associated with changes in economic policy, therefore showing the statements being good indicators of recessions, boom periods, or periods of low and stable inflation.

In order to see whether or not our clusters were correlated with financial market performance, we decided to examine the relationship of each cluster to the VIX index. The CBOE Volatility Index (VIX) is derived from the prices of SP 500 options with near term expiration dates, and then generates a 30-day forward projection of volatility in the market. The values of the VIX index are inversely related to the SP 500, where higher values imply higher volatility and lower prices of the SP 500 index. We would expect that if the clusters had statistical significance with regards to financial markets, VIX index values would be meaningfully different across clusters.

Methodology

The data set used for this project compromised of 119 FOMC statements ranging from the time periods 1999-2001, 2006-2010, 2015-2016, and 2020-2024. This subset of dates were chosen because it includes data across four separate Federal Reserve chairs and unique economic and financial crises beginning with the Dot-Com Bubble of 2000, and ending in the post-Covid period characterized by historically high inflation and monetary tightening. A subset of periods are also characterized for their robust growth and stability, notably 2006, 2010, 2015-2016, and the post-Covid economic rebound period in 2021 and 2022.

The FOMC statements were manually compiled from the Federal Reserve Board of Governors website. The FOMC statements were cleaned using R software to remove any unnecessary paragraph indentations or spacing, as well as special characters. The statements were also cleaned to remove the voting records of individual Federal Reserve Bank chairs so it would not interfere with clustering. The VIX index was downloaded from the FRED (Federal Reserve Economic Database) published by the Federal Reserve Bank of St. Louis, and the corresponding dates were mapped to the dates of the FOMC statements.

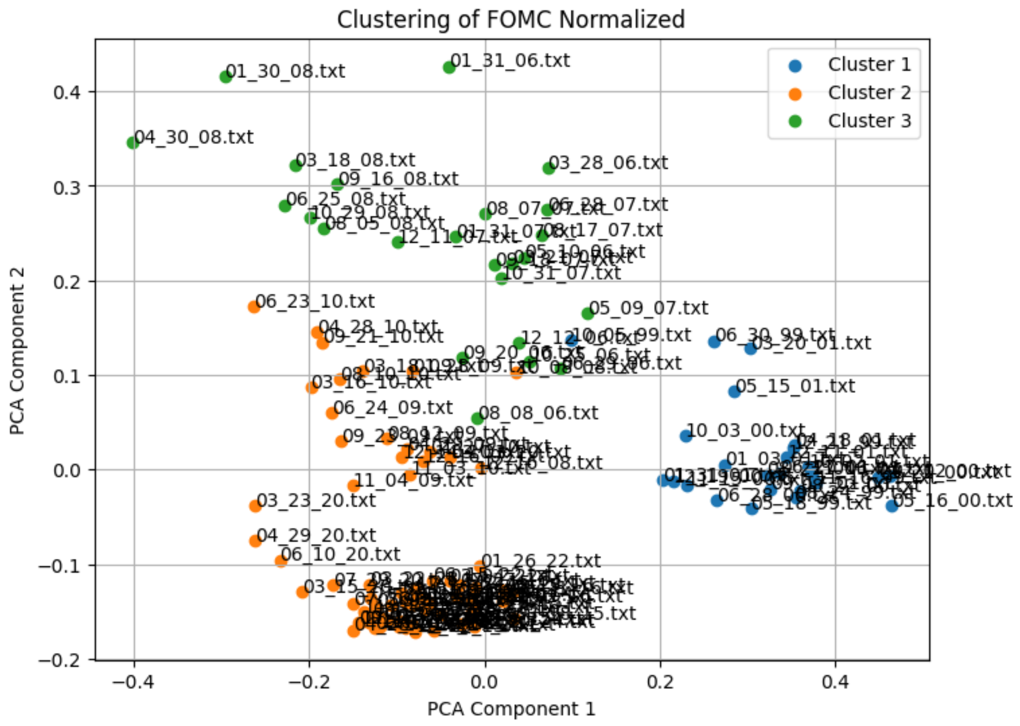
Once the FOMC statements were cleaned, they were then turned into vectors either by word count or by the Google Universal Sentence Encoder (USE). Two different types of vectors were utilized to account for semantic differences in sentences that would not be signalled in a vector that only analyzes word count. This is important in the context of FOMC statements, as the sentences ″inflation is still exceedingly high,″ and ″inflation has moderated″ both mention inflation, but have very different meanings.

The vectors were first normalized, and then clustered using K-Means with $K = 3$ clusters. While the elbow plots were produced for the NLP and word-count vectors, they were quite smooth so no obvious cutoff point existed for the optimal number of clusters. As we wanted to cluster by changes in interest rate policy across FOMC statements, 3 clusters were chosen. Principle Component Analysis (PCA) was performed after clustering. The first and second principle components were plotted against each other to visualize the clusters in 2-D space. Once each FOMC statement was labelled with a cluster, the averages of the corresponding VIX score was calculated, and a hypothesis test was conducted to calculate the 95% confidence interval to see if the clusters were statistically significant from each other in the context of financial market volatility.

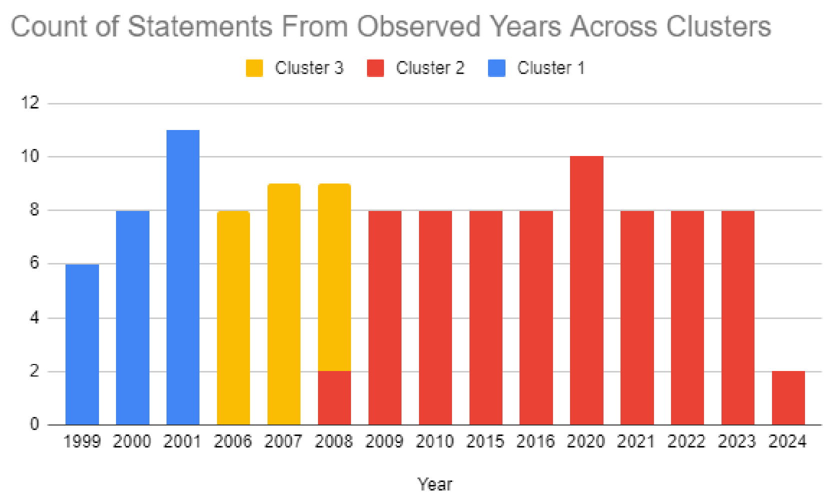
Results Discussion

Word Count Vectorization

Initially, the vectors were generated by making a vector of all the unique words across all FOMC statements. This yielded vectors of high dimensionality (1224).

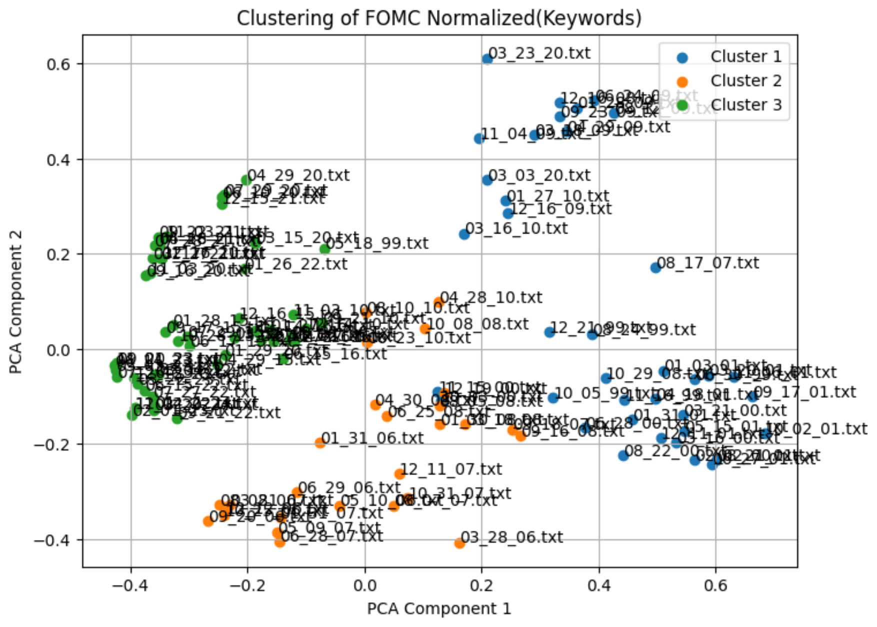


The FOMC statements do cluster to an extent. The best cluster is the blue cluster (cluster 1), which clusters around a period of economic expansion in 1999, 2000, and 2001 prior to the 2001 recession. The orange cluster (cluster 2) appears divided between the dense cluster at the bottom and the more sparse cluster toward the middle. Overall, the green cluster (cluster 3) was the weakest, with no clear cluster but acting more as the catch-all for FOMC statements that did not fit into either of the other clusters.

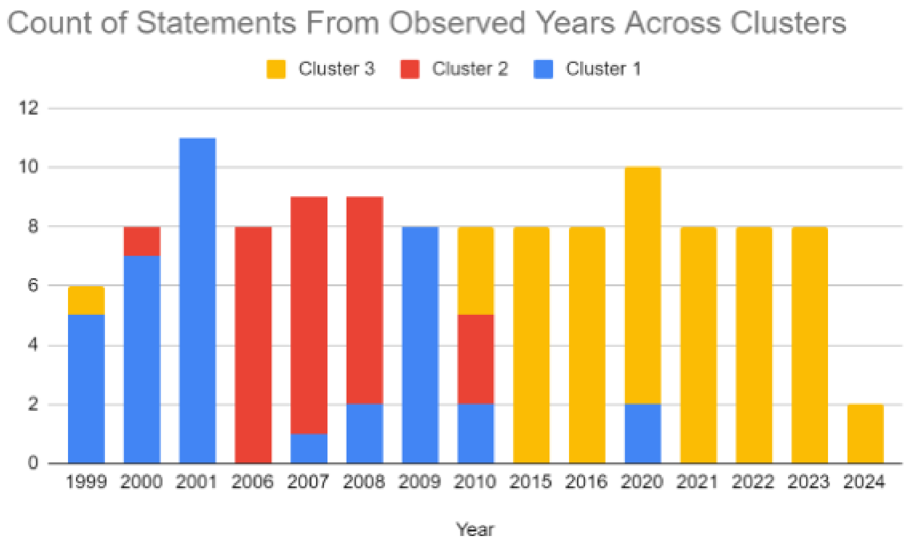


The bar graph further shows that clusters cluster more around the time period than economic status. Because all the unique words were used to make the vectors, the large, dense mass of cluster 2 is due to words related to the COVID-19 pandemic.

The conclusion was that allowing for all words leads to clustering by timer period because, during specific events, specific words would be more providently and then nonexistent in other clusters, leading to the cluster identifying time period instead of the economy's performance. This led to a further investigation, this time only looking for the frequency of economic words, ignoring all others.



Here, the clusters still cluster more around the year, even when excluding specific words and focusing on economic words. This ultimately illustrates that this model is differentiating more about the writer's style and word choice than any real meaning in terms of economic performance. The bar graph further illustrates that the clusters span more years but still resemble the first trial.



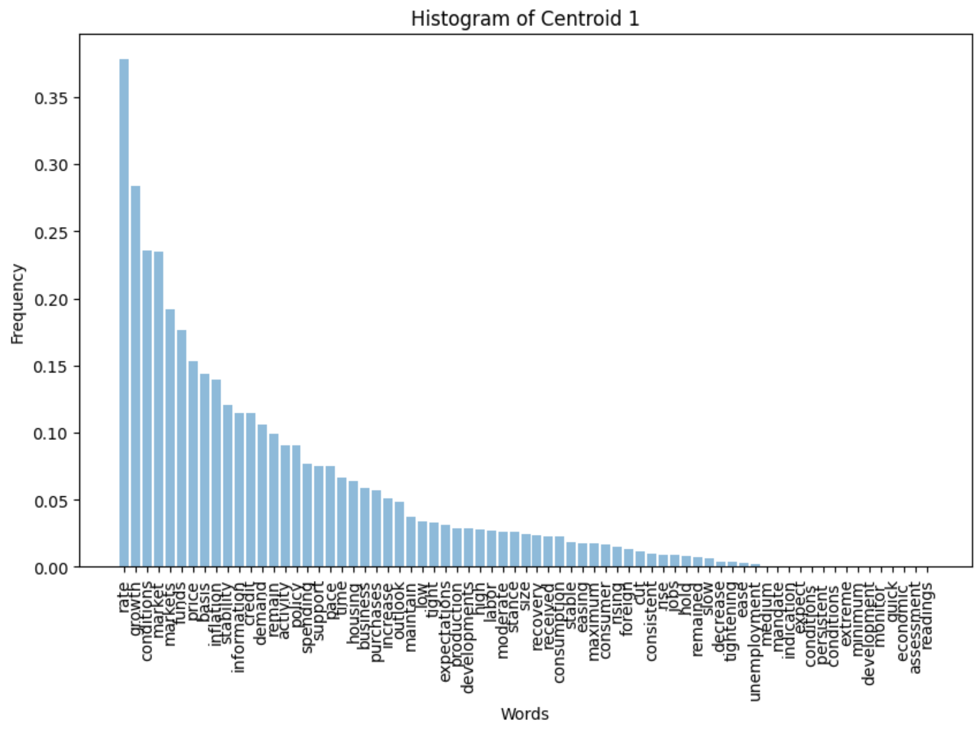
From this, bar graphs showing the frequency of the given words can be made. This further shows the similarity of the clusters and how the algorithm fails to separate with any degree of certainty. VIX confidence intervals were constructed to measure the difference in clusters quantitatively. and the results are shown below.

Table 1: 95% confidence intervals for Word Frequency

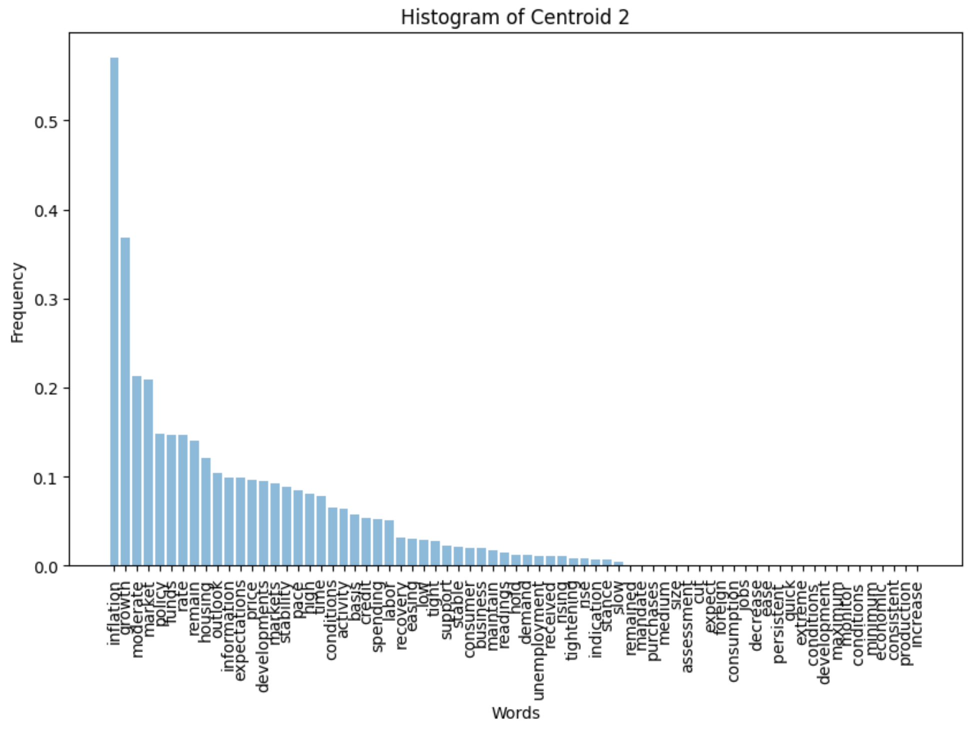
	Mean	St Dev	2.5%	97.5%
Cluster 1	24.83	4.83	19.16	35.41
Cluster 2	23.23	10.18	12.73	53.16
Cluster 3	20.01	12.46	10.55	47.87

Table 2: 95% confidence intervals for Key Words

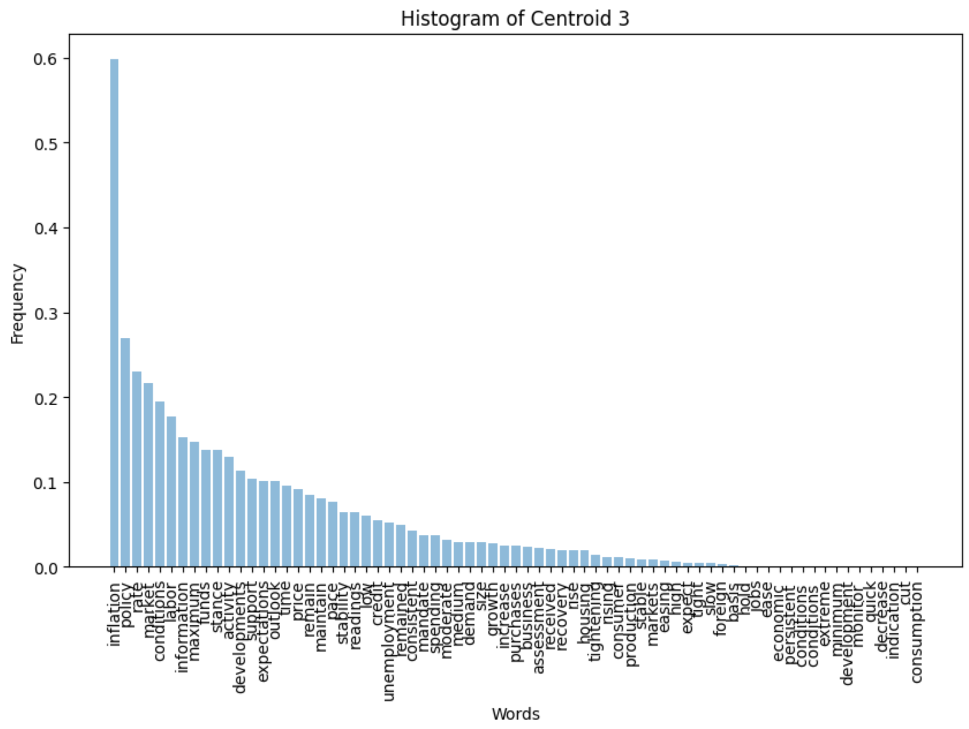
	Mean	St Dev	2.5%	97.5%
Cluster 1	28.91	11.45	17.67	62.22
Cluster 2	19.83	9.79	10.57	40.64
Cluster 3	20.19	6.39	12.60	34.48



Top 3 words of centroid 1: rate, growth, conditions.



Top 3 words of centroid 2: inflation, growth, moderate.

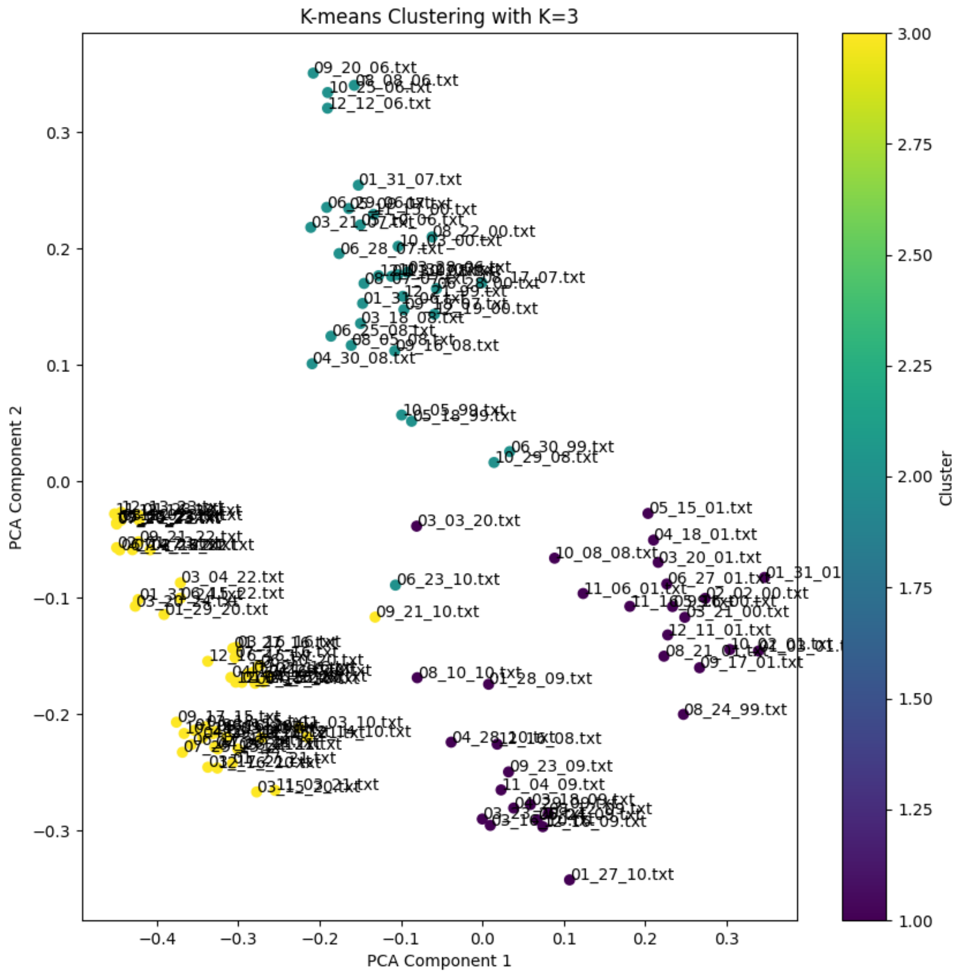


Top 3 words of centroid 3: inflation, policy, rate

NLP Vectorization

Using Google's universal sentence encoder, each statement was embedded in 512-dimensional space as a unit vector. In this embedded space, vectors of similar meaning or sentiment are relatively close to each other. The hope was that by using this more sophisticated method of vectorization we would be able to cluster the data in a more meaningful way.

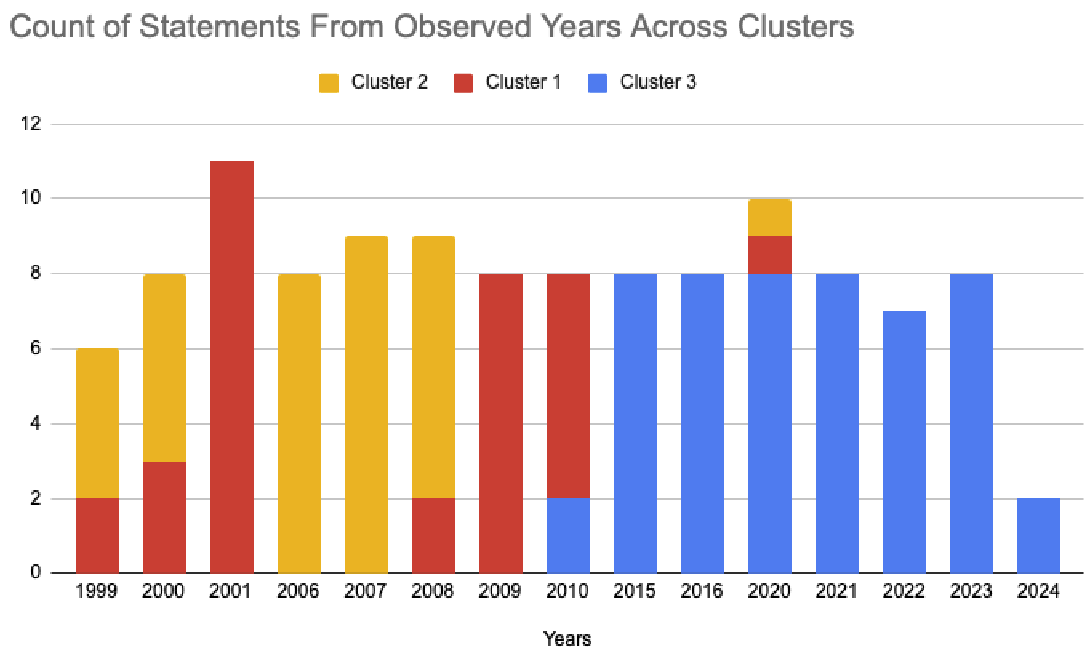
Below is the resulting clustering with the embedding of each cluster plotted along its first and second Principal components to visualize the clustering at a 2-dimensional level. Clusters are present, but cluster 1 and 2 (blue, green) blend.



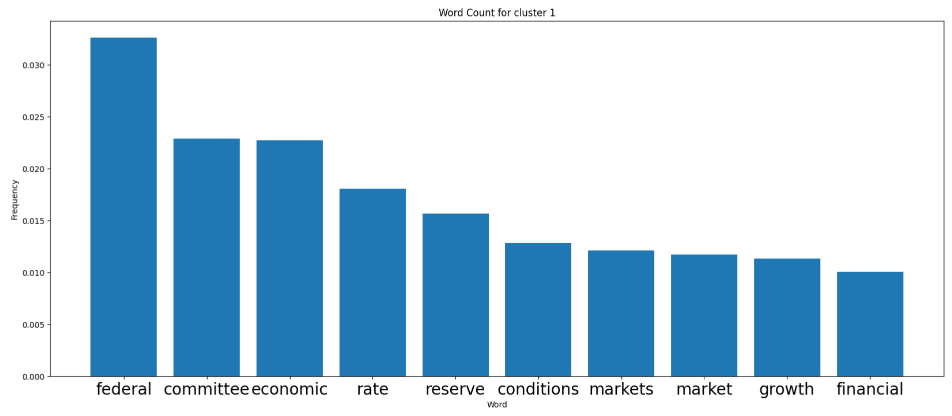
So, what patterns can we observe within the clusters? As the plot above is rather cluttered we first plotted a histogram of the year of publications (below) to see how the clusters differ chronologically.

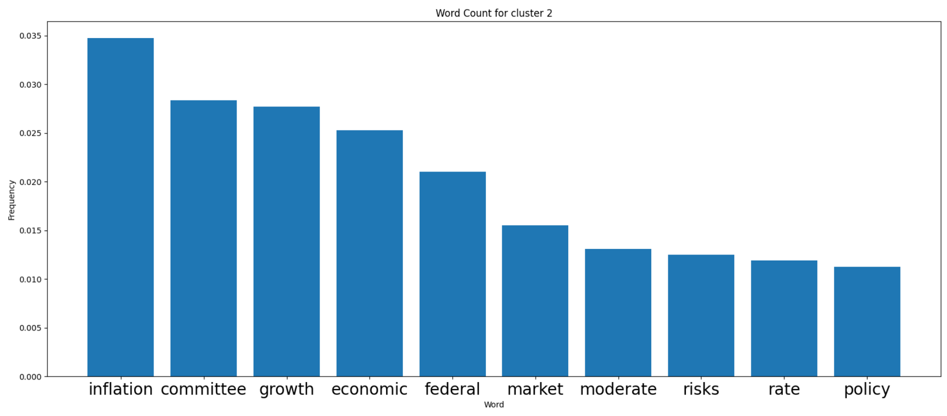
The year of publication profiles for each cluster below are different than the previous methods. Years are organized into clusters in either 1, 2 or even 3 groups, suggesting that the clusters contain more information than simply when the statements were released. Broadly we see a 2008 recession and dot.com bubble cluster for cluster 1, broad economic growth for cluster 2 and a COVID-19/present-day cluster for cluster 3.

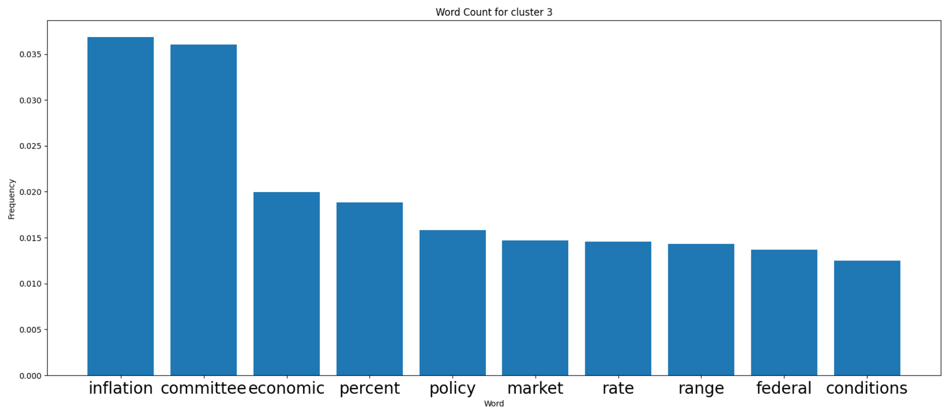
Using this data we can start to think about what economic trends were present during the years each cluster encompasses and try and make predictions based on how future FOMC statements cluster. However, many of the clusters span multiple years, and also the clusters overlap in year of FOMC statement publication, so we kept looking for patterns within the clusters.



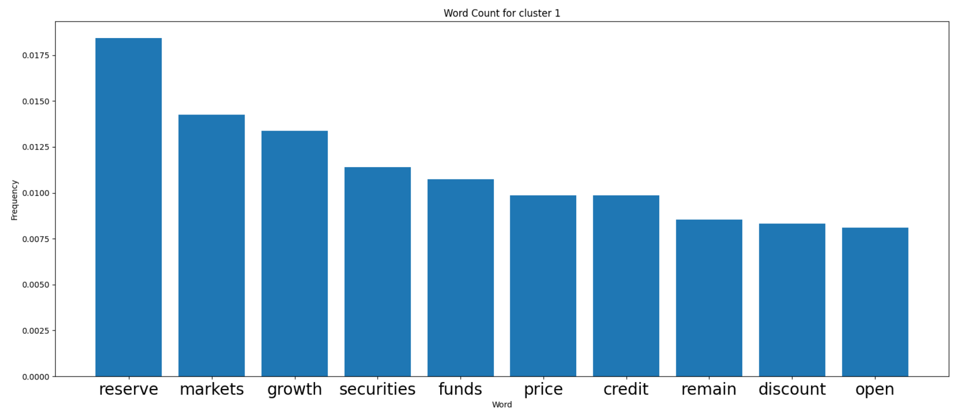
Chronologically the clusters show some similarity but what about semantically? To see how similar the statements between each cluster are we looked at the word frequency within each cluster. First, we looked at which words were most frequent that weren't stop words ("the", "a", etc.).

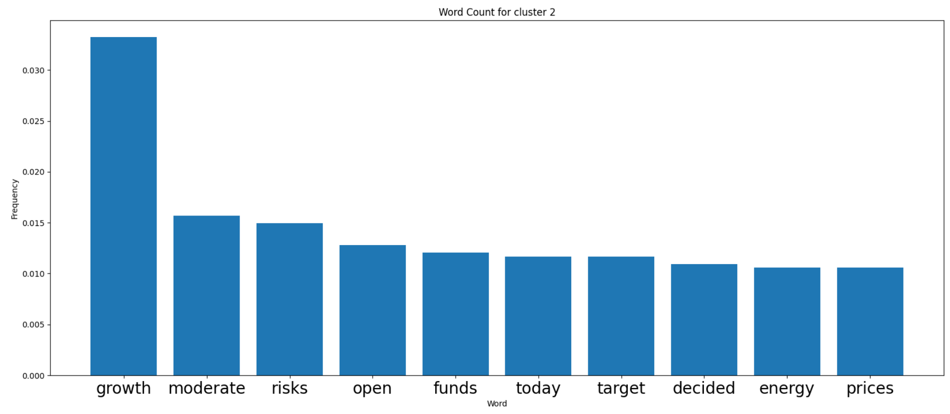


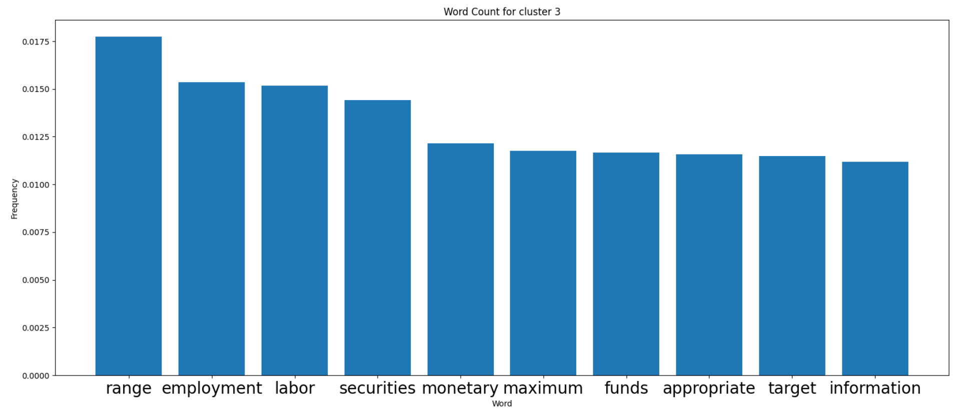




Upon closer examination, some interesting patterns emerge from the analysis. Despite inflation being the most common word in clusters 2 and 3, it doesn't even make the top 10 most frequent words in cluster 1. But, words like "committee" and "economic" are in the top 4 of all the clusters. To account for this, we then did an analysis of the most common words across all the clusters and we found the most frequent words (in order): "committee", "inflation", "economic", "federal", "rate", "market", "percent." Thus, to find which words distinguish each cluster, we examined the word count once more after removing the 10 most common words across clusters from each of the three clusters. The histograms for these word counts are plotted below.







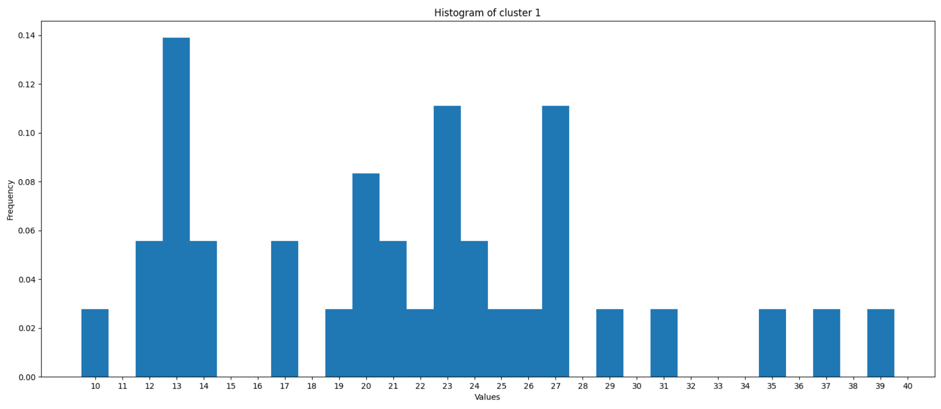
In cluster 1, the word ’reserve’ has a spike. In cluster 2 we see that ’growth’ is a distinctive term, which makes intuitive sense as cluster 2 consists of FOMC statements mainly from times of economic growth 1999-2008, excluding 2001. For cluster 3 we see words like range, employment, and labor are popular perhaps owing to that this cluster consists of statements released during economic recovery after the 2008 recession, as well as contains the COVID recession and recovery where the labor market remained surprisingly strong.

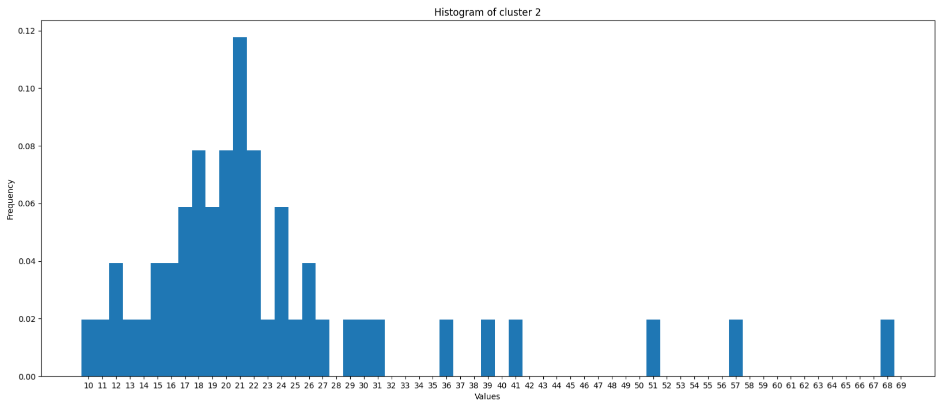
Also after removing the 10 most common words we found that the cluster's non-stop word count of (5369, 3283, 13058) decreased by 15%, 14.5%, 19.3% respectively which suggests that cluster 3 is the least distinct cluster. However, cluster 3 is also the largest and spans the most years.

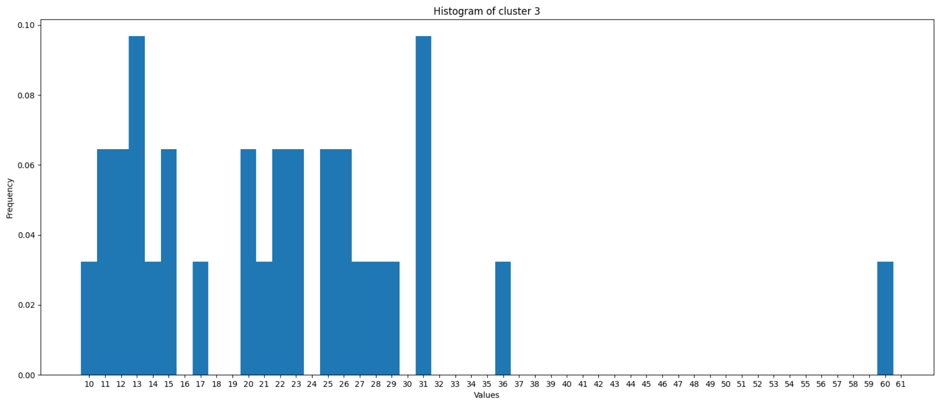
Finally, we also analyzed the VIX index profile of each cluster. The results showed the average VIX scores were close to each other for each cluster, showing each cluster contained relatively similar amounts of volatility. Cluster 2 did exhibit a slightly normal VIX score distribution but still had lots of outliers. Cluster 1 and 3's distributions were more or less uniform.

Table 3: Average VIX Index Scores Across NLP Clusters

	Mean
Cluster 1	21.95
Cluster 2	23.82
Cluster 3	22.54



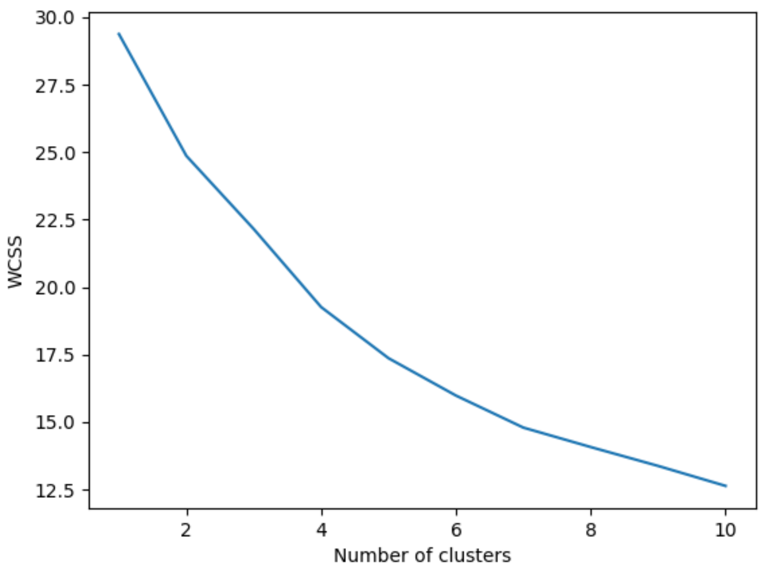




Potential Next Steps

As this area of research is currently rapidly developing, there are numerous areas of further research that could be undertaken. Firstly, we could adjust the macroeconomic indicators we are examining across clusters to see which ones are most dynamic when it comes to receiving new fed policy information from FOMC statements. Specifically, analyzing high frequency trading data in the time shortly after the meetings occurred would allow us to examine the more immediate effects of the policies announced.

Additionally, we arbitrarily chose 3 clusters for our analysis but interesting trends may lie at other cluster counts. Below is the elbow graph for USE clustering, which like our other elbow graphs didn't help in choosing the cluster count due to its smoothness. Using a different metric for choosing clusters could help us uncover previously unseen patterns in the data, and potentially lead to clusters that don't just fit the data chronologically.

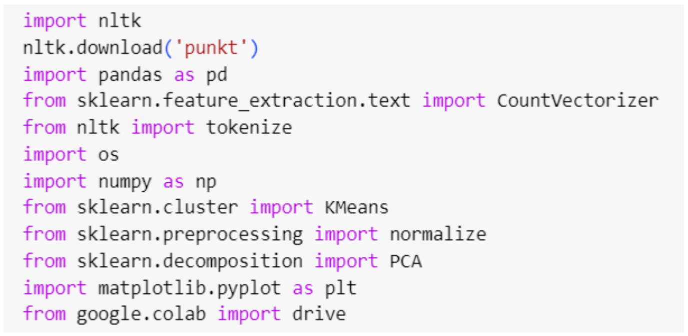


Supervised learning techniques and how they apply to the qualitative information embedded in FOMC statements may also be an avenue of further research. Supervised learning techniques and regression could be used to find a quantifiable relationship between the calculated sentiment scores we used to cluster our data and specific variables, such as bond prices, asset purchases, and even stock index prices. This would give a more insight on the direct impact of sentiment scores in financial markets.

Code Appendix

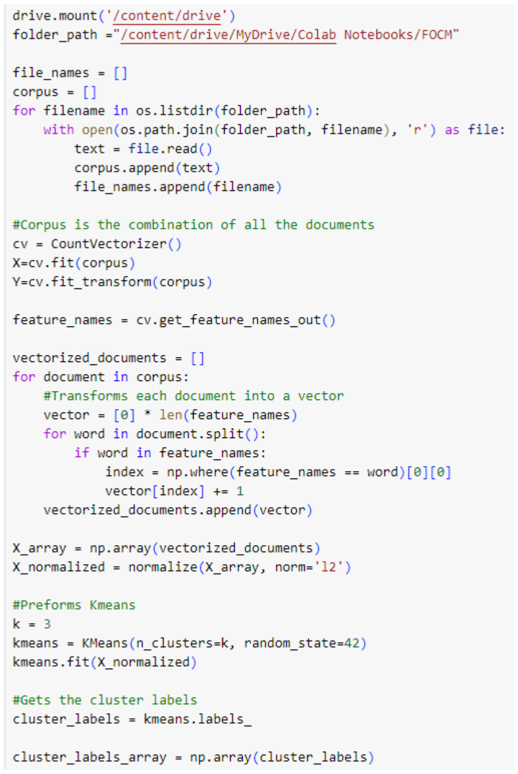
0.1 Word count Code

Packages used:



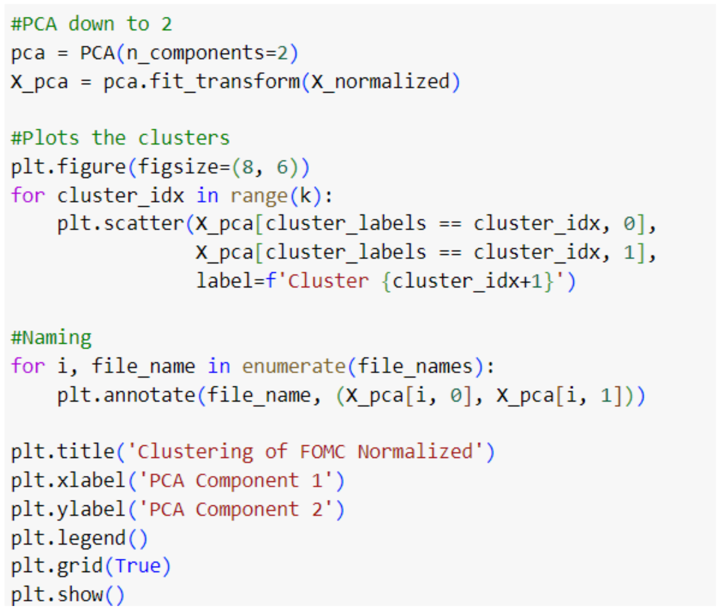
```
import nltk
nltk.download('punkt')
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from nltk import tokenize
import os
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from google.colab import drive
```

K-Means code and Vectorization (1224):



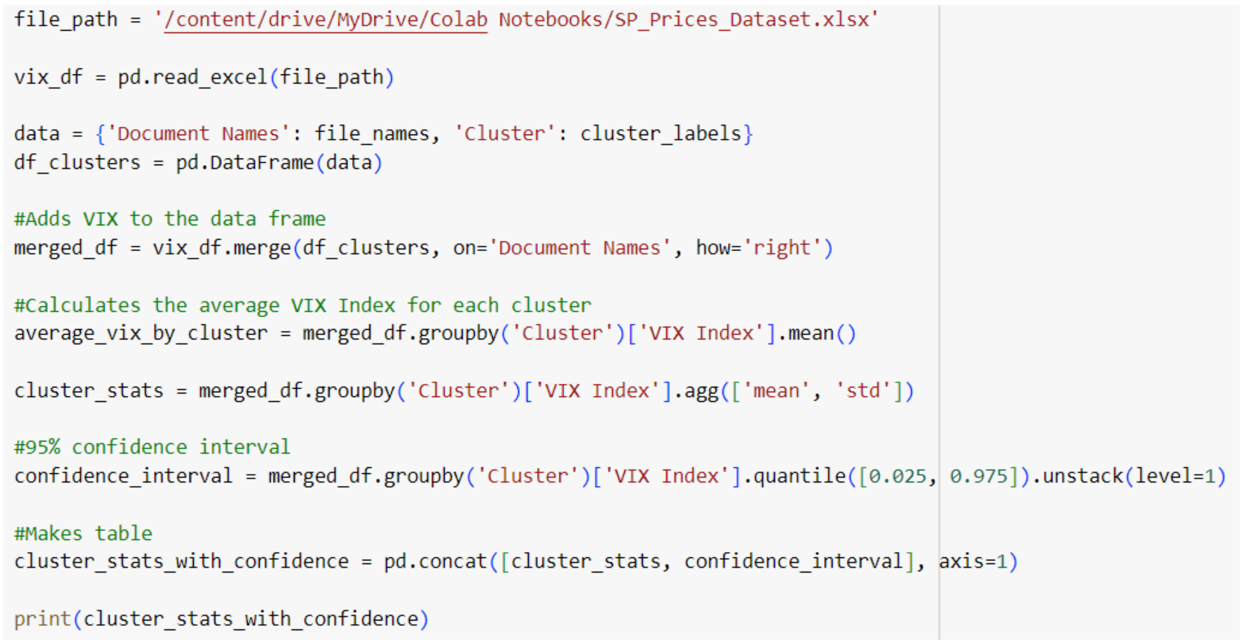
```
drive.mount('/content/drive')
folder_path ="/content/drive/MyDrive/Colab Notebooks/FOCM"

file_names = []
corpus = []
for filename in os.listdir(folder_path):
    with open(os.path.join(folder_path, filename), 'r') as file:
        text = file.read()
        corpus.append(text)
        file_names.append(filename)

#Corpus is the combination of all the documents
cv = CountVectorizer()
X=cv.fit(corpus)
Y=cv.fit_transform(corpus)

feature_names = cv.get_feature_names_out()

vectorized_documents = []
for document in corpus:
    #Transforms each document into a vector
    vector = [0] * len(feature_names)
    for word in document.split():
        if word in feature_names:
            index = np.where(feature_names == word)[0][0]
            vector[index] += 1
    vectorized_documents.append(vector)

X_array = np.array(vectorized_documents)
X_normalized = normalize(X_array, norm='l2')

#Preforms Kmeans
k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X_normalized)

#Gets the cluster labels
cluster_labels = kmeans.labels_

cluster_labels_array = np.array(cluster_labels)
```

Cluster graph (1224):



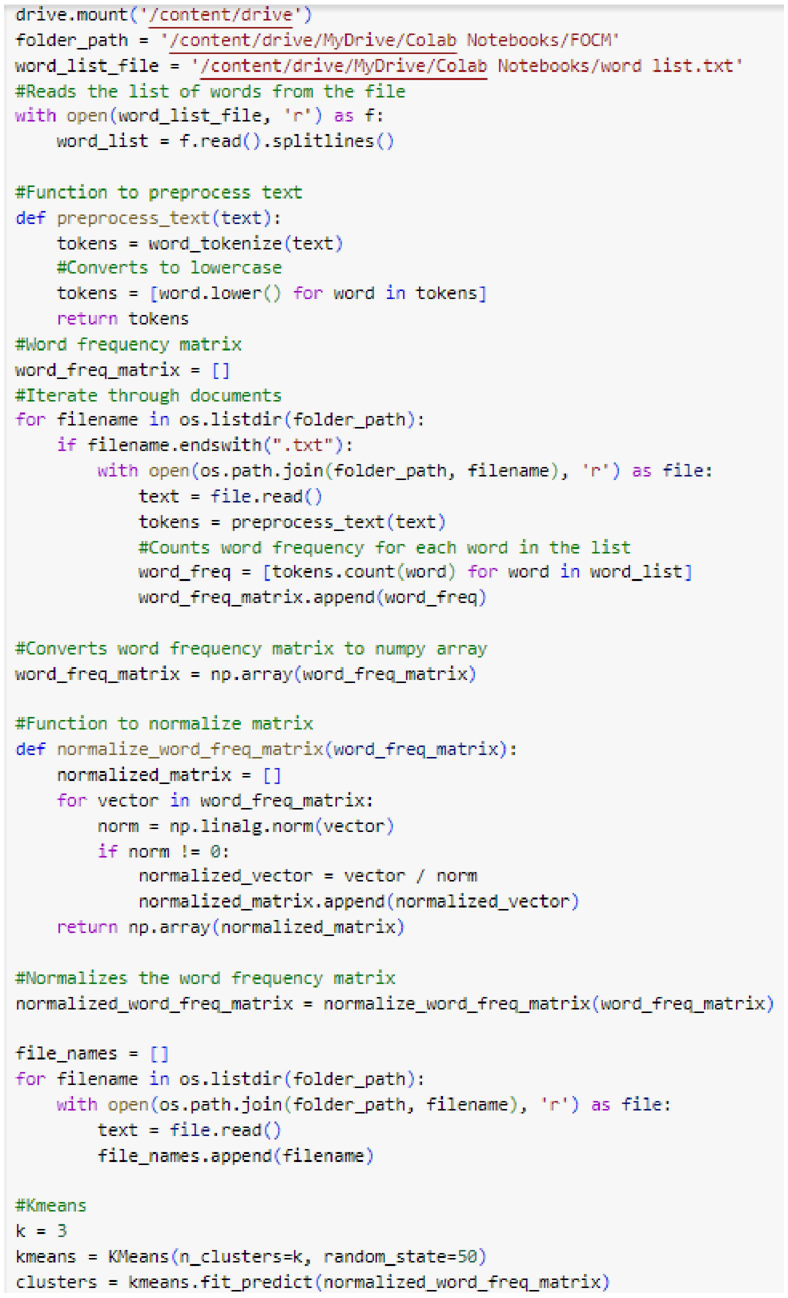
```python
#PCA down to 2
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_normalized)

#Plots the clusters
plt.figure(figsize=(8, 6))
for cluster_idx in range(k):
    plt.scatter(X_pca[cluster_labels == cluster_idx, 0],
                X_pca[cluster_labels == cluster_idx, 1],
                label=f'Cluster {cluster_idx+1}')

#Naming
for i, file_name in enumerate(file_names):
    plt.annotate(file_name, (X_pca[i, 0], X_pca[i, 1]))

plt.title('Clustering of FOMC Normalized')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.grid(True)
plt.show()
```

Confidence Interval for 95% (1224):



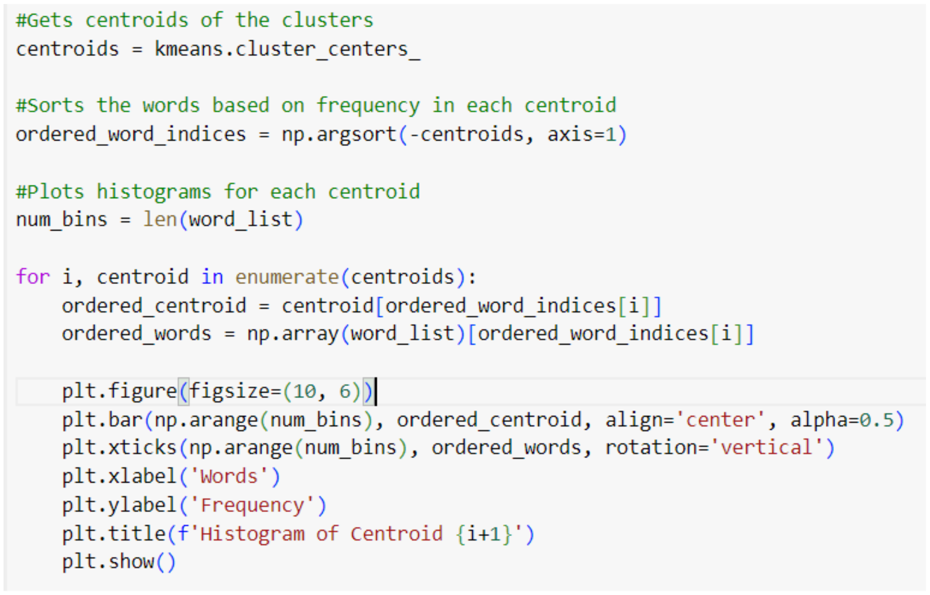
```python
file_path = '/content/drive/MyDrive/Colab Notebooks/SP_Prices_Dataset.xlsx'

vix_df = pd.read_excel(file_path)

data = {'Document Names': file_names, 'Cluster': cluster_labels}
df_clusters = pd.DataFrame(data)

#Adds VIX to the data frame
merged_df = vix_df.merge(df_clusters, on='Document Names', how='right')

#Calculates the average VIX Index for each cluster
average_vix_by_cluster = merged_df.groupby('Cluster')['VIX Index'].mean()

cluster_stats = merged_df.groupby('Cluster')['VIX Index'].agg(['mean', 'std'])

#95% confidence interval
confidence_interval = merged_df.groupby('Cluster')['VIX Index'].quantile([0.025, 0.975]).unstack(level=1)

#Makes table
cluster_stats_with_confidence = pd.concat([cluster_stats, confidence_interval], axis=1)

print(cluster_stats_with_confidence)
```

K-Means code and Vectorization (Key Words):



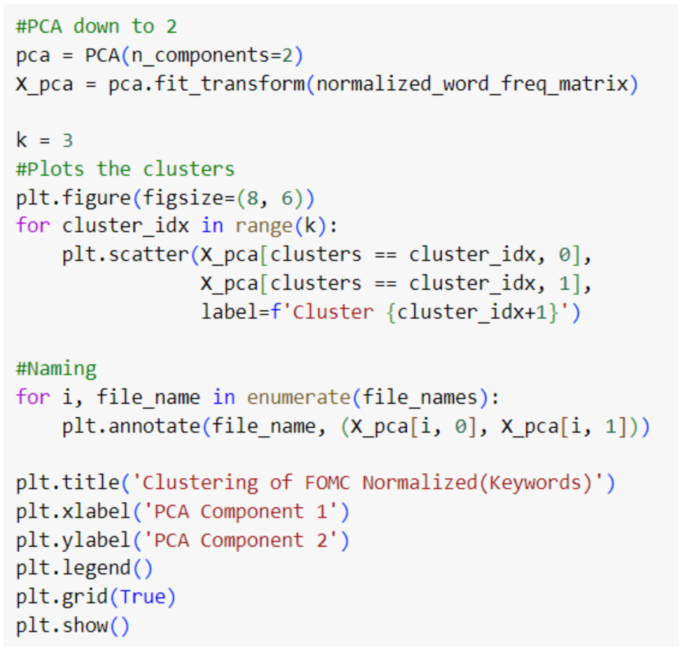
```python
drive.mount('/content/drive')
folder_path = '/content/drive/MyDrive/Colab Notebooks/FOCM'
word_list_file = '/content/drive/MyDrive/Colab Notebooks/word list.txt'
#Reads the list of words from the file
with open(word_list_file, 'r') as f:
    word_list = f.read().splitlines()

#Function to preprocess text
def preprocess_text(text):
    tokens = word_tokenize(text)
    #Converts to lowercase
    tokens = [word.lower() for word in tokens]
    return tokens
#Word frequency matrix
word_freq_matrix = []
#Iterate through documents
for filename in os.listdir(folder_path):
    if filename.endswith(".txt"):
        with open(os.path.join(folder_path, filename), 'r') as file:
            text = file.read()
            tokens = preprocess_text(text)
            #Counts word frequency for each word in the list
            word_freq = [tokens.count(word) for word in word_list]
            word_freq_matrix.append(word_freq)

#Converts word frequency matrix to numpy array
word_freq_matrix = np.array(word_freq_matrix)

#Function to normalize matrix
def normalize_word_freq_matrix(word_freq_matrix):
    normalized_matrix = []
    for vector in word_freq_matrix:
        norm = np.linalg.norm(vector)
        if norm != 0:
            normalized_vector = vector / norm
            normalized_matrix.append(normalized_vector)
    return np.array(normalized_matrix)

#Normalizes the word frequency matrix
normalized_word_freq_matrix = normalize_word_freq_matrix(word_freq_matrix)

file_names = []
for filename in os.listdir(folder_path):
    with open(os.path.join(folder_path, filename), 'r') as file:
        text = file.read()
        file_names.append(filename)

#Kmeans
k = 3
kmeans = KMeans(n_clusters=k, random_state=50)
clusters = kmeans.fit_predict(normalized_word_freq_matrix)
```

Word Histogram Code:



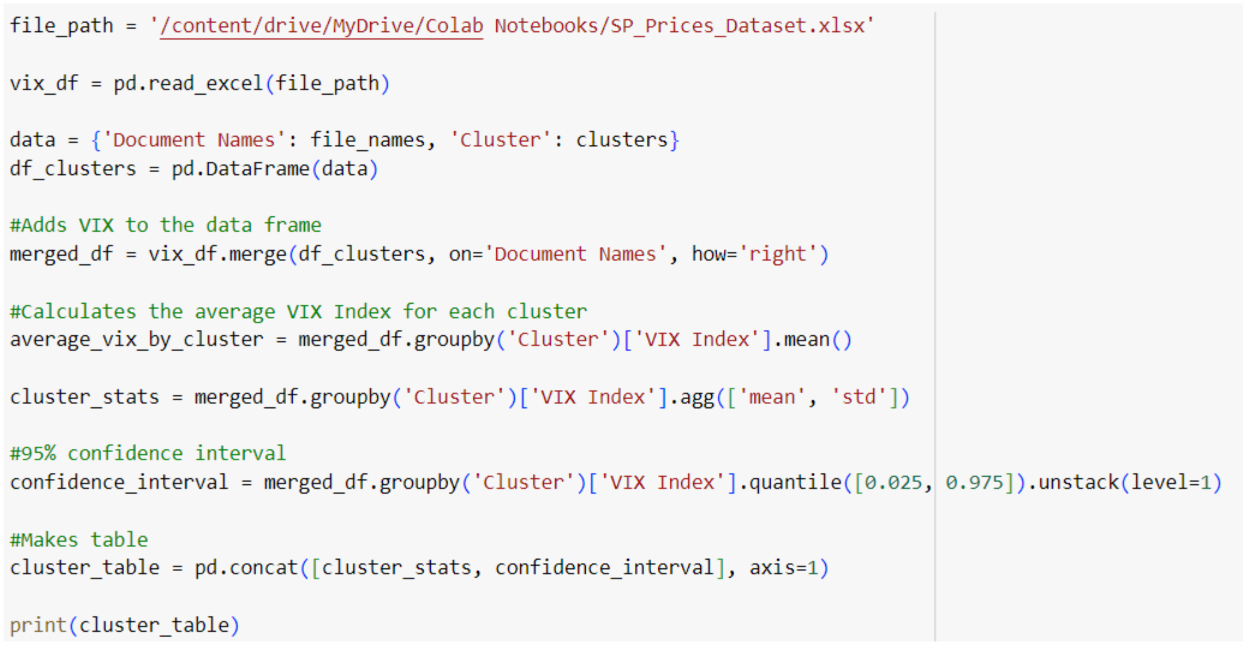
```
#Gets centroids of the clusters
centroids = kmeans.cluster_centers_

#Sorts the words based on frequency in each centroid
ordered_word_indices = np.argsort(-centroids, axis=1)

#Plots histograms for each centroid
num_bins = len(word_list)

for i, centroid in enumerate(centroids):
    ordered_centroid = centroid[ordered_word_indices[i]]
    ordered_words = np.array(word_list)[ordered_word_indices[i]]

    plt.figure(figsize=(10, 6))
    plt.bar(np.arange(num_bins), ordered_centroid, align='center', alpha=0.5)
    plt.xticks(np.arange(num_bins), ordered_words, rotation='vertical')
    plt.xlabel('Words')
    plt.ylabel('Frequency')
    plt.title(f'Histogram of Centroid {i+1}')
    plt.show()
```

Cluster graph (Key Words):



```
#PCA down to 2
pca = PCA(n_components=2)
X_pca = pca.fit_transform(normalized_word_freq_matrix)

k = 3
#Plots the clusters
plt.figure(figsize=(8, 6))
for cluster_idx in range(k):
    plt.scatter(X_pca[clusters == cluster_idx, 0],
                X_pca[clusters == cluster_idx, 1],
                label=f'Cluster {cluster_idx+1}')

#Naming
for i, file_name in enumerate(file_names):
    plt.annotate(file_name, (X_pca[i, 0], X_pca[i, 1]))

plt.title('Clustering of FOMC Normalized(Keywords)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.grid(True)
plt.show()
```

Confidence Interval for 95% (64):



```
file_path = '/content/drive/MyDrive/Colab Notebooks/SP_Prices_Dataset.xlsx'

vix_df = pd.read_excel(file_path)

data = {'Document Names': file_names, 'Cluster': clusters}
df_clusters = pd.DataFrame(data)

#Adds VIX to the data frame
merged_df = vix_df.merge(df_clusters, on='Document Names', how='right')

#Calculates the average VIX Index for each cluster
average_vix_by_cluster = merged_df.groupby('Cluster')['VIX Index'].mean()

cluster_stats = merged_df.groupby('Cluster')['VIX Index'].agg(['mean', 'std'])

#95% confidence interval
confidence_interval = merged_df.groupby('Cluster')['VIX Index'].quantile([0.025, 0.975]).unstack(level=1)

#Makes table
cluster_table = pd.concat([cluster_stats, confidence_interval], axis=1)

print(cluster_table)
```

Clustering for USE analysis

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Vectorize the statements
message_embeddings = embed(statements)

cluster_count = 3

# Run K-means with K=3
kmeans = KMeans(n_clusters= cluster_count)
kmeans.fit(message_embeddings)
clustering = kmeans.labels_

# perform PCA on the statements
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit_transform(message_embeddings)
firstComponent = pca.components_[0]
secondComponent = pca.components_[1]

# projects embeddings onto PCPs
X = []
Y = []


for embedding in message_embeddings:
    X.append(np.dot(embedding.numpy(), firstComponent))
    Y.append(np.dot(embedding.numpy(), secondComponent))

# Plot the data with labels

plt.figure(figsize=(10, 10))

for i in range(len(clustering)):
  clustering[i] += 1

plt.scatter(X, Y, c=clustering, cmap='viridis')
plt.title('K-means Clustering with K=3')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()


# Add labels
for i, fileName in enumerate(fileNames):
    plt.annotate(fileName, (X[i], Y[i]))
```

Word Count for USE analysis

```
import string
from string import digits
import re

from nltk.corpus import stopwords


#remove the commonCount most common words for the documents

stopWords = stopwords.words("english")

def wordCount(statement, bank, tot=0):
  statement = statement.lower()
  statement = statement.strip(".")
  remove_punc = str.maketrans('', '', string.punctuation)
  statement = statement.translate(remove_punc)

  remove_digits = str.maketrans('', '', digits)
  statement = statement.translate(remove_digits)

  for word in re.split("\s+", statement):
    if word not in stopWords:
      if word in bank:
        bank[word] += 1
      else:
        bank[word] = 1
      tot += 1
  return tot

#word count on all the statements to remove the most used words

totalBank = dict()

for statement in statements:
  wordCount(statement, totalBank)

#Sort totalBank

totalBank = dict(sorted(totalBank.items(), key = lambda a: a[1], reverse =
                                              True))


commonCount = 10
commonWords = (list(totalBank.keys())[0:commonCount])


import matplotlib.pyplot as plt

# scrape through clusters to find frequency

# cluster count 1 though 3
```

```
bank = dict()
idx = 3
tot = 0

for i in range(len(message_embeddings)):

    if (clustering[i] == idx):
      # fail safe because there's one statement with no VIX value
      tot += wordCount(statements[i],bank)

#Remove the common words
collisions = 0
totRemove = tot
for commonWord in commonWords:
  if commonWord in bank:
    totRemove -= bank[commonWord]
    bank.pop(commonWord)
    collisions += 1

#sort the bank

bank = dict(sorted(bank.items(), key = lambda a: a[1], reverse = True))


# Extract keys and values
labels = list(bank.keys())
values = list(bank.values())

# Normalize values
for i in range(len(values)):
  values[i] = values[i] / totRemove



# Plotting
plt.figure(figsize=(20, 8))
plt.bar(labels[:10], values[:10])
plt.xlabel('Word')
plt.ylabel('Frequency')
plt.title('Word Count for cluster ' + str(idx))
plt.xticks(fontsize=20)


from google.colab import files
plt.savefig("endWords3.png")
files.download("endWords3.png")

print("Total words before: " + str(tot))
print("Taking away the " + str(commonCount) + " most commons words from
                                     the clustsers")
print("Total words after taking out common words: " + str(totRemove))
print("Number of collisions (overlapping words): " + str(collisions))
```