Sentiment Analysis Using Unsupervised Methods for Economics

Completed for the Certificate in Scientific Computation and Data Sciences
Spring 2023

Nat Fernelius
Undergraduate
Economics
The University of Texas at Austin

[illegible]

Olivier Coibion
Malcolm Forsam Centennial Professor
Economics

Abstract

In this project I develop a complete, unsupervised learning method for generating a categorical sentiment measure for use within an econometric regression of the unemployment rate. The sentiment measure itself was developed and training data obtained from Federal Open Market Committee (FOMC) minutes reports that are made publicly available on the Federal Reserve website. The project process itself can be easily divided into several distinct phases. First, the textual data is procured from the Federal Reserve's website. I use basic web scraping techniques and Python's Beautiful Soup library to accomplish this task. Then, the text is processed into term frequency–inverse document frequency (TF-IDF) vectors for use with the unsupervised learning model. Through trial and error, k-means clustering is determined to be the optimal model type for unsupervised categorization. This model is then optimized through the elbow plot method, and the TF-IDF vectors are fed into the model to create a categorical variable. From there, the generated categorical variable is utilized within an existing macroeconomic regression of the unemployment rate, along with a traditional, pretrained dictionary sentiment analysis measure from the Natural Language Toolkit (NLTK). The results of this regression are then analyzed for statistical and economic significance to assess its true usefulness within predictive macroeconomic research.

Introduction

The problem my project seeks to address is one of information. Within macroeconomics the accurate prediction of important macroeconomic indicators like the unemployment rate is absolutely critical for businesses, governments, and individuals to make sound and informed economic decisions. Macroeconomic indicators are not mere numbers, but rather highly relevant facts about the reality of the economic world. The unemployment rate, for example, tangibly represents proportionally how many real people do not have a job and are actively seeking one. This is something real, everyday people care about, and it should affect how they plan for their future. The more accurate macroeconomic forecasting of these indicators can be, the better prepared everyone can be for the future. Predictions like this can be made through many means; however, they all require one thing: information. After all, a prediction based upon nothing is fortune telling. A key problem within macroeconomics is making sure the maximum amount of valuable and predictively significant information is being used to make these predictions.

For many years, quantitative values derived from major macroeconomic indices have made up the bulk of this information along with untold amounts of qualitative assessments of textual information. However, qualitative assessments often lack the requisite rigor of predictive science and do not mesh well with quantitative data within established predictive models like regressions.

Recently, economists have begun to realize textual information itself is actually a valuable source of quantitative data. One of the easiest to understand pieces of textual data is that of a sentiment score. Sentiment, within a natural language processing context, refers to the relative level of negative or positive emotion in a body of text. In other words, it can approximate the general meaning of a text in terms of its negativity or positivity. Sentiment scoring has existed in some capacity since at least the 1950s but has picked up steam in recent years as one of the most important quantitative insights that someone can derive from a given piece of economically relevant text.[1]

Traditionally, sentiment scores have been generated either through hardcoded sentiment dictionaries, that assign sentiment scores to words and produce weighted averages over a piece of text by adding up those word-scores by some method, or use supervised machine learning methods that train upon textual data that has been prelabled by experts as having a certain sentiment. Both of these methods are quite labor intensive, and limit the amount of sentiment data available to potential macroeconomic forecasters.

Thus, I set out to create an easy to replicate sentiment categorizer that utilizes unsupervised learning methods. Unsupervised learning methods have the benefit of requiring no hardcoding of sentiment dictionaries or laborious labeling of training data. It presupposes that the relationships in the words within a piece of text is more than enough to establish distinct sentimental groupings.

By converting each piece of text into a vector based on word frequency, specifically a TF-IDF, I generate a distinct quantitative footprint relating to each FOMC minutes report that can be interpreted in relation to all other generated vectors. From there I just need to create a learning method to group these vectors into categories.

I predicted that a k-means clustering algorithm would be effective in producing economically and statistically significant categorical measures of a text's sentiment from these vectors.

Through analyzing a regression that included a categorical variable produced by a k-means clustering algorithm implemented upon the minutes report texts, I determined that unsupervised sentiment analysis does hold significant predictive power toward the unemployment rate and potentially many other macroeconomic variables.

Materials and Methods

This project relied entirely on Python 3, its accompanying libraries, and the Jupyter Notebook environment.

For the first major phase of this project, I focused on textual data collection and web scraping. Web scraping simply refers to computational methods through which data can be pulled from websites in an automated manner. Developing a scraping pipeline was necessary for this project, as I needed textual data in sufficient quantity and with sufficient levels of organization and precision such that manual methods were infeasible. I utilized the premier Python web scraping library known as Beautiful Soup for this task.[2] It is called Beautiful Soup because the library produces a "soup" of HTML code that is then parsed through systematically to obtain relevant data.

To start this process I needed a base URL to feed the library for it to create its baseline parsing soup. The Federal Reserve's website did contain all necessary data for a soup, as the minutes text embedded within specific HTML files was, through enough mouse clicks, accessible from the website's homepage. To simplify things for myself I learned through experimentation that the URL "https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm" contained direct

[1]Scott Sims, "Sentiment Analysis 101," KDnuggets (KD Nuggets), accessed January 30, 2023, https://www.kdnuggets.com/2015/12/sentiment-analysis-101.html.

[2]"Beautiful Soup Documentation," Beautiful Soup Documentation - Beautiful Soup 4.4.0 documentation (Beautiful Soup), accessed January 31, 2023, https://beautiful-soup-4.readthedocs.io/en/latest/.

links to all relevant minutes report data within its corresponding HTML code.[3] This served as the base URL for the scraping procedure, and formed the base of the primary soup that needed parsing. First, I used the requests.get() method of the library to obtain a textual representation of the aforementioned HTML file. Then, I converted this textual variable into a parsable Beautiful Soup object using the namesake BeautifulSoup() function. From there, I was able to parse out all URLs from the soup for further investigation and encode them into a Python list. Using list comprehension and base Python methods I was able to narrow this list to only contain URLs with FOMC minutes data within their HTML code. Then I used simple regular expression (regex) expressions to split up portions of these URLs into a variable representing date. I then filtered the URLs by date based on a list of relevant dates I created. The resulting cleaned list of URLs was then fed into a function that downloaded the minutes text within the corresponding HTML files. These were saved within text files in a generated directory. Each text file was titled by which date the minutes came from. This directory contained all textual data necessary for this project.

The next step was to obtain enough relevant macroeconomic data to create the assessment regression. This data was found within the FRED-MD database constructed by the St. Louis Federal Reserve bank.[4] There is an easily downloadable CSV file with all relevant macroeconomic data organized by date. This is the tangible file I downloaded named "current.csv." Having a CSV file was of great importance, because CSV data easily interfaces with Python's Pandas library which I used as the base library for the organization of the rest of the project. All the rest of this project's work was applied to a Pandas dataframe.

The next step was to create this dataframe. I used Pandas' read_csv() method to easily upload the FRED-MD dataset to the Jupyter Notebook Python environment while preserving the structure of the dataset as it was found in the CSV. Then, it was trimmed down by dropping irrelevant macroeconomic measures.

I then needed to integrate the textual data. By creating a dictionary of dates and the corresponding textual data of each file, I then constructed a dataframe of all text data by date. I then grabbed the list of all relevant dates and filtered the main dataframe to only contain dates that have a corresponding statement. All that was left to do was make sure the date formats were standardized and to perform a merge of the main and text datasets. The resulting new main data frame contained all relevant macroeconomic indices along with the minutes text corresponding to each particular date.

To assess the categorical sentiment measure I plan to later create's effectiveness, I needed a baseline sentiment measure within the regression. Using the Natural Language Toolkit's (NLTK) sentiment analyzer known as VADER, an automated dictionary sentiment analysis method, I was able to create a new column representing the negative sentiment of each minutes report.[5] Using Python's shift method I was also able to create lags in relevant macroeconomic variables in concordance with effective economic regression practices. After dropping the text column, the dataframe only contained

[3]"Meeting Calendars and Information," The Fed - Meeting calendars and information (The Federal Reserve), accessed January 31, 2023, https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm.
[4]"Federal Reserve Economic Data: Fred: St. Louis Fed," FRED (Federal Reserve Bank of St. Louis), accessed January 31, 2023, https://fred.stlouisfed.org/.
[5]"Python: Sentiment Analysis Using Vader," GeeksforGeeks (GeeksforGeeks, October 7, 2021), https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/.

quantitative variables. This is quite convenient as once I create the unsupervised method's sentiment measure, all I will have to do is append that measure to the main dataframe and separate out the unemployment rate variable as the regression target. Then, the entire frame can be fed into the regression as its features, and the effectiveness of my sentiment measure can be easily assessed.

From there, I was finally able to move on to the meat of the project, namely developing a sentiment analysis method using an unsupervised learning method. The first task within this segment was choosing which unsupervised learning method I wanted to use. I needed a method that could easily and systematically group different minutes documents into discrete categories reflecting their sentimental contents. When I think of grouping, k-means clustering is the first algorithm that comes to mind and thus I went about utilizing k-means clustering to solve my problem. If it failed, I would have no problem trying a different method, even though that did not end up being necessary.

K-means clustering works by plotting points within a certain dimensional space and then creating k number of clusters surrounding k different means. It does this through a simple algorithm.[6] First, k different critical points, known as centroids, are plotted randomly within the space. The algorithm then assigns all data points to the closest centroid to it. By dividing the sum of the data points assigned to each centroid by the number of datapoints each centroid is then recalculated. This process repeats and the centroids move until after recalculation there is no change in each centroid. After this, the algorithm remembers which centroid each point optimally belongs to and assigns them categories based on their centroid membership. This final categorical assignment is the sentiment measure I am seeking. If I can obtain a quantitative vector that sentimentally represents each piece of text, this algorithm should work flawlessly.

Therefore, I needed to vectorize the pieces of text into usable mathematical vectors. I turned towards a term frequency–inverse document frequency (TF-IDF) vector to accomplish this task.[7] This type of vectorization is excellent at distinguishing pieces of text from one another by mathematically representing their difference from one another relative to the corpus. This is done through relatively simple frequency calculations modified by two general principles. Namely, the vectorization procedure gives more weight to infrequently used terms and more importance to terms used frequently in each document. FOMC minutes themselves use many of the same words and phrases throughout the entire corpus; however, texts with different sentiments often use distinct words from the broader corpus and use them frequently. It was my hope that these distinctions would be enough for a cluster of TF-IDF vectors to be neatly grouped into sentiment categories by the clustering algorithm.

Thus, I utilized Python libraries to assign each minutes report to a distinct TF-IDF vector. After that, I needed to begin using the k-means clustering algorithm on these vectors. But first, I needed to determine what value of k made sense to use. This is easily done with an elbow plot.[8] This plots k on the x axis and distortion, a measure of the squared distances from the cluster centers of the clusters, on the y axis. Distortion

[6]Natasha Sharma, "K-Means Clustering Explained," neptune.ai, January 25, 2023, https://neptune.ai/blog/k-means-clustering.
[7]Anirudha Simha, "Understanding TF-IDF for Machine Learning," Capital One (Capital One, October 6, 2021), https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.
[8]"Elbow Method for Optimal Value of K in Kmeans," GeeksforGeeks (GeeksforGeeks, January 11, 2023), https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/.

generally decreases with k. When distortion begins to drop at a lower rate indicating diminishing returns, an elbow should form. This elbow represents the optimal value of k because it minimizes distortion and preserves the discrete categorical nature of the clusters. If k is too high the information afforded by each category diminishes, and if it is too low the data remains insufficiently distinguished. Thus, I created the elbow plot Figure 1 below.

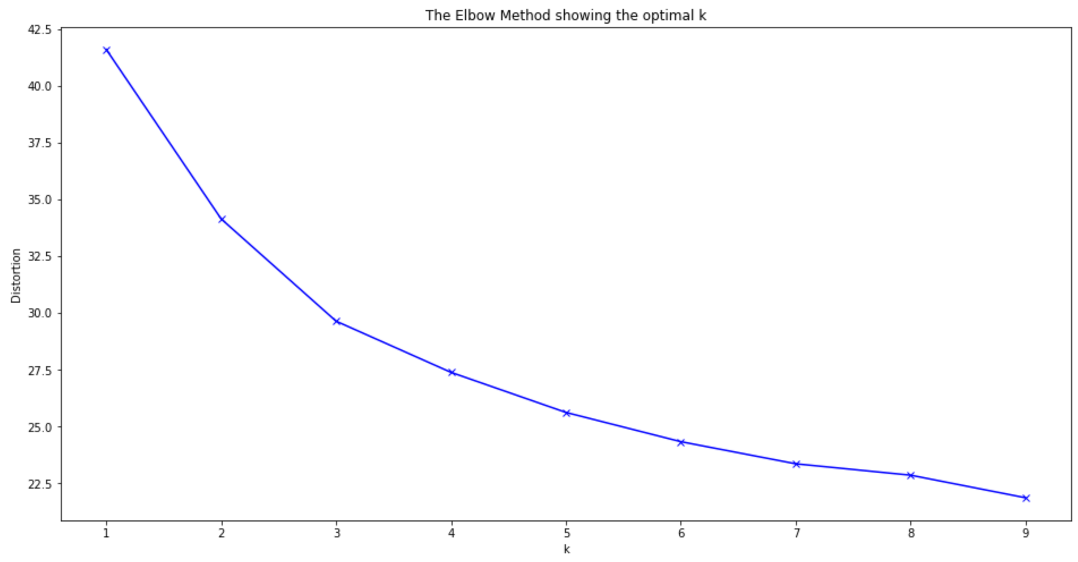


Figure 1: Elbow Plot of Distortion with Respect to k.

The elbow was relatively subtle, but there is a clear shift in the lowering of distortion at k equals 3. Thus, I used the value 3 for k in the final model.

Now, I just needed to run the vectors through the finalized model and to add their categorical labels to the regression dataframe. This was easily completed with the scikit-learn library's k-means clustering methods.[9] The dataframe column that contained the vectors was fed into the algorithm and the clustering algorithm iteratively assigned categories to each of them in a new column.

After that, I dropped the now useless vector column and prepared to feed the data into a diagnostic regression of the unemployment rate. The y-value of the regression was the unemployment rate column, and the remaining columns served as features. From there, I generated a regression table of important statistics and interpreted the results. This concludes the discussion of this project's technical methods.

Results

In order to evaluate the significance of my unsupervised sentiment measure for macroeconomic prediction of unemployment, a multilinear diagnostic regression was generated. The model was constructed from nationwide macroeconomic indicator data along with the dictionary based sentiment measure and the unsupervised sentiment measure I created with k-means clustering.

[9]"Sklearn.cluster.kmeans," scikit learn (scikit learn), accessed January 31, 2023, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

$$unemp_{+10} = \beta_0 + \beta_1 base\ sentiment + \beta_2 sasdate + \beta_3 CPI + \beta_4 INDPRO + \beta_5 S\&P + \beta_6 FEDFUNDS + \beta_7 unemp$$
$$\beta_8 categorical\ sentiment + lags + u$$

A variable without any sub notation denotes the current year of analysis. Positive or negative numbers in a variable's sub field denote how many federal reserve committee reports before or after the current date the variable refers to. Positive numbers indicate future report dates, while negative numbers indicate past report dates.

Unemp refers to the nationwide unemployment rate of the United States as a percentage. *Sasdate* refers to the base date of each row of observations. *CPI* refers to the Consumer Price Index for All Urban Consumers in the base year. *INDPRO* is an index of industrial production with an index base year of 2017. *S&P* refers to the S&P 500 stock index of 500 large companies' performance on the New York stock exchange. *FEDFUNDS* refers to the Effective Federal Funds Rate. *base sentiment* refers to the baseline dictionary sentiment measure generated by VADER. *categorical sentiment* refers to the sentiment measure created by k-means clustering.

The lag term for all models included a -1,-3, -6, and -10 reading for the unemployment rate, and the federal funds rate.

$$lags = \beta_8 unemp_{-1} + \beta_9 unemp_{-3} + \beta_{10} unemp_{-6} + \beta_{11} unemp_{-10} + \beta_{12} FEDFUNDS_{-1} + \beta_{13} FEDFUNDS_{-3} +$$
$$\beta_{14} FEDFUNDS_{-6} + \beta_{15} FEDFUND_{-10}$$

The error term is represented by *u*.

The results of the 10 minutes release prediction horizon's diagnostic regression can be found in Table 1 below.

	Table 1 Regression Summary	
	Coefficient	P-value
sasdate	-6.939e-09	1.723e-01
CPIAUCSL	6.900e-02	2.654e-02
INDPRO	1.118e-01	2.194e-01
S and P 500	-8.444e-04	4.309e-07
FEDFUNDS	-4.512e-01	7.775e-05
unemployment rate	7.645e-01	1.555e-26
base sentiment	2.103e+00	9.138e-08
UNRATE-1	1.863e-01	1.960e-22
UNRATE-3	-2.480e-02	9.584e-16
UNRATE-6	1.714e-01	9.207e-09
FEDFUNDS-1	1.551e-01	7.722e-04
FEDFUNDS-3	3.116e-01	2.750e-02
FEDFUNDS-6	-1.204e-02	5.947e-01
FEDFUNDS-10	5.007e-01	1.825e-01
past rate	-3.253e-01	1.668e-03
categorical sentiment	-1.745e-02	9.202e-03
r2	0.9091	
MSE	0.2858	
RMSE	0.5346	

Table 1: Diagnostic Regression Results for 10 Minutes Release Horizon

Both the base sentiment measure and the categorical sentiment measure had significant p-values within the regression ($p < .001$).

To increase my confidence in these results, I also performed a regression with a prediction horizon of 5 minutes releases. The results of this second diagnostic regression can be found in Table 2 below.

	Table 2 Short-Term Regression Summary	
	Coefficient	P-value
sasdate	-1.040e-08	6.410e-02
CPIAUCSL	9.596e-02	1.012e-02
INDPRO	-5.419e-03	8.441e-03
S and P 500	-1.006e-03	1.669e-08
FEDFUNDS	-4.741e-01	8.541e-13
unemployment rate	8.547e-01	1.266e-64
base sentiment	-2.167e-02	6.089e-07
UNRATE-1	9.158e-02	3.746e-54
UNRATE-3	-3.521e-01	2.261e-38
UNRATE-6	2.568e-01	2.783e-23
FEDFUNDS-1	1.628e-01	3.746e-11
FEDFUNDS-3	2.918e-01	4.284e-08
FEDFUNDS-6	-1.193e-01	1.915e-04
FEDFUNDS-10	2.281e-01	1.395e-01
past rate	-2.186e-01	4.258e-11
categorical sentiment	-8.100e-02	4.651e-04
r2	0.9625	
MSE	0.1163	
RMSE	0.3410	

Table 2: Diagnostic Regression Results for 5 Minutes Release Horizon

Again, in this shorter prediction horizon regression my categorical sentiment measure had a significant p-value ($p<0.001$).

This indicates that our unsupervised sentiment measure did have a significant impact on the prediction of the unemployment rate, as it in fact did significantly contribute to the weighting of the regression itself. This is a definitive result. Thus, I can say that my prediction that the unsupervised categorical sentiment measure would perform well in the regression was supported.

Discussion

The implications of these findings are wide ranging. Firstly, it is with confidence that I can say I helped to solve an information problem within macroeconomics by providing a method to successfully extract usable predictive information from text without requiring expert hardcoding of textual features or labeling large volumes of training data. By using this method, I was able to improve established prediction methods of the unemployment rate, a measure that millions of Americans care about.

Since my unsupervised categorical measure of sentiment was a statistically and economically significant predictor of the unemployment rate when used in conjunction with other macroeconomic variables it stands to reason that similar sentiment analysis methods when performed on different pieces of text or when used to predict different macroeconomic variables could be a fruitful path of exploration. More broadly, my results suggest that text itself has more quantitative data content than many people probably appreciate. With advances in machine learning technology that necessitate usable quantitative information to function, text may become an ever more important component of the data science environment.

Acknowledgements

I would like to thank my project supervisor Dr. Olivier Coibion for being an invaluable and patient resource throughout my writing process. I would also like to thank my brother Henry, my parents Chris and Jennifer Fernelius, and my grandparents for always supporting me in my academic pursuits. Additionally, it is with the utmost gratitude that I thank my friends and the peers I have met along the way in college for driving me to be my best self. Finally, I would like to thank the entire SDS department for their incredible support during this process.

Literature Review

In this section I want to provide an overview of the established literature surrounding sentiment analysis, uses of textual data for macroeconomic prediction, and unsupervised learning methods that inspired and guided my project.

First of all, it was important to establish that FOMC minutes indeed contained important information about the macroeconomy that could fill in information gaps that would exist without their publication. If this could not be ascertained, no data science method could create significant quantitative data from what would be insignificant pieces of text. However, Romer and Romer (2000) did in fact show that the Federal Reserve itself has asymmetric information to the private sector, specifically in terms of commercial inflation forecasts.[10] FOMC minutes are released to establish the current attitudes of the Federal Reserve towards the macroeconomy. Thus, it would follow that releasing FOMC meeting minutes closes some of these information asymmetries in a way beneficial to economic forecasters, and that the pieces of text themselves hold some significance toward macroeconomic prediction.

Second, I looked toward established supervised methods of sentiment analysis and their implications towards macroeconomic prediction. In terms of utilization of supervised or dictionary-based sentiment analysis methods for the analysis of FOMC minutes data there are several papers of note. Gürkaynak et al. (2005) found that FOMC sentiment data was useful for forecasting the federal funds rate.[11] Tadle (2022) utilized supervised sentiment analysis methods on both FOMC statement and minutes data and determined that the sentiments of policy documents contain information that can influence the predictability of a financial market.[12] It specifically focused on implications for the predictability/volatility of fed funds futures contracts, broad equity, real estate investment trust indices, and various exchange rate indices involving the U.S. dollar.

My paper not only generates an unsupervised categorization of sentiment using FOMC minutes but also evaluates such a measure's viability for predictive

[10] Christina D. Romer and David H. Romer, "Federal Reserve Information and the Behavior of Interest Rates" (The University of California Berkeley, 2000), https://eml.berkeley.edu/wp/c+dromer_aer2000.pdf.
[11]Refet S Gürkaynak, Brian Sack, and Eric Swanson, "Do Actions Speak Louder than Words? the Response of Asset Prices to Monetary Policy Actions and Statements - IJCB - May 2005," Premier issue (May 2005) of the International Journal of Central Banking (International Journal of Central Banking, 2005), https://www.ijcb.org/journal/ijcb05q2a2.htm.
[12]Mu-Yen Chen and Ting-Hsuan CHen, "Modeling Public Mood and Emotion: Blog and News Sentiment and Socio-Economic Phenomena," ScienceDirect (Future Generation Computer Systems, November 3, 2017), https://www.sciencedirect.com/science/article/abs/pii/S0167739X17323750?via%3Dihub.

macroeconomics through the use of macroeconomic regression methods giving it a viable point of departure from the established literature.

References

"Beautiful Soup Documentation¶." Beautiful Soup Documentation - Beautiful Soup 4.4.0 documentation. Beautiful Soup. Accessed January 31, 2023. https://beautiful-soup-4.readthedocs.io/en/latest/.

Chen, Mu-Yen, and Ting-Hsuan CHen. "Modeling Public Mood and Emotion: Blog and News Sentiment and Socio-Economic Phenomena." ScienceDirect. Future Generation Computer Systems, November 3, 2017. https://www.sciencedirect.com/science/article/abs/pii/S0167739X17323750?via%3Dihub.

"Elbow Method for Optimal Value of K in Kmeans." GeeksforGeeks. GeeksforGeeks, January 11, 2023. https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans

"Federal Reserve Economic Data: Fred: St. Louis Fed." FRED. Federal Reserve Bank of St. Louis. Accessed January 31, 2023. https://fred.stlouisfed.org/.

Gürkaynak, Refet S, Brian Sack, and Eric Swanson. "Do Actions Speak Louder than Words? the Response of Asset Prices to Monetary Policy Actions and Statements - IJCB - May 2005." Premier issue (May 2005) of the International Journal of Central Banking. International Journal of Central Banking, 2005. https://www.ijcb.org/journal/ijcb05q2a2.htm.

"Meeting Calendars and Information." The Fed - Meeting calendars and information. The Federal Reserve. Accessed January 31, 2023. https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm.

"Python: Sentiment Analysis Using Vader." GeeksforGeeks. GeeksforGeeks, October 7, 2021. https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/.

Romer, Christina D., and David H. Romer. "Federal Reserve Information and the Behavior of Interest Rates." The University of California Berkeley, 2000. https://eml.berkeley.edu/wp/c+dromer_aer2000.pdf.

Sharma, Natasha. "K-Means Clustering Explained." neptune.ai, January 25, 2023. https://neptune.ai/blog/k-means-clustering.

Simha, Anirudha. "Understanding TF-IDF for Machine Learning." Capital One. Capital One, October 6, 2021. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

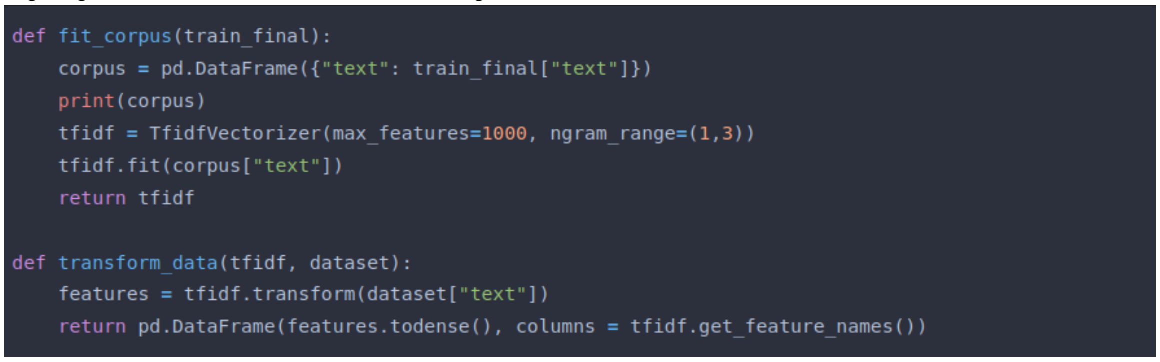
Sims, Scott. "Sentiment Analysis 101." KDnuggets. KD Nuggets. Accessed January 30, 2023. https://www.kdnuggets.com/2015/12/sentiment-analysis-101.html.

"Sklearn.cluster.kmeans." scikit learn. scikit learn. Accessed January 31, 2023. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

Computer Code

All code for this paper is included within my GitHub appendix available at the following link: https://github.com/ferneliusn/sentiment_unemployment

However, there were a few especially notable pieces of code I wanted to highlight. The first is the code dealing with textual data vectorization.

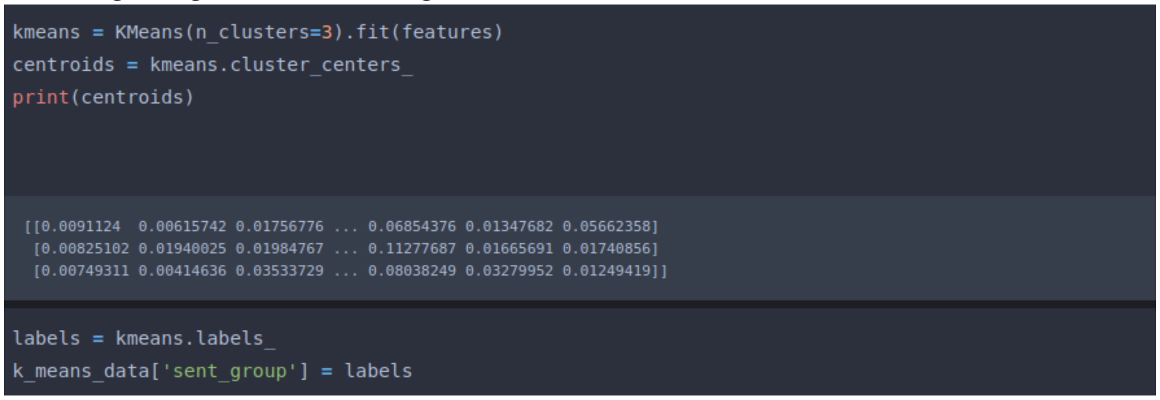


```
def fit_corpus(train_final):
    corpus = pd.DataFrame({"text": train_final["text"]})
    print(corpus)
    tfidf = TfidfVectorizer(max_features=1000, ngram_range=(1,3))
    tfidf.fit(corpus["text"])
    return tfidf

def transform_data(tfidf, dataset):
    features = tfidf.transform(dataset["text"])
    return pd.DataFrame(features.todense(), columns = tfidf.get_feature_names())
```

The first function fits the TFIDF vectorizer over the sum total of textual data utilized for the project. The second function is what we feed the dataframe into to actually complete the vectorization.

The next important piece of code was the code that actually performed k means clustering and generated the categorical measure.



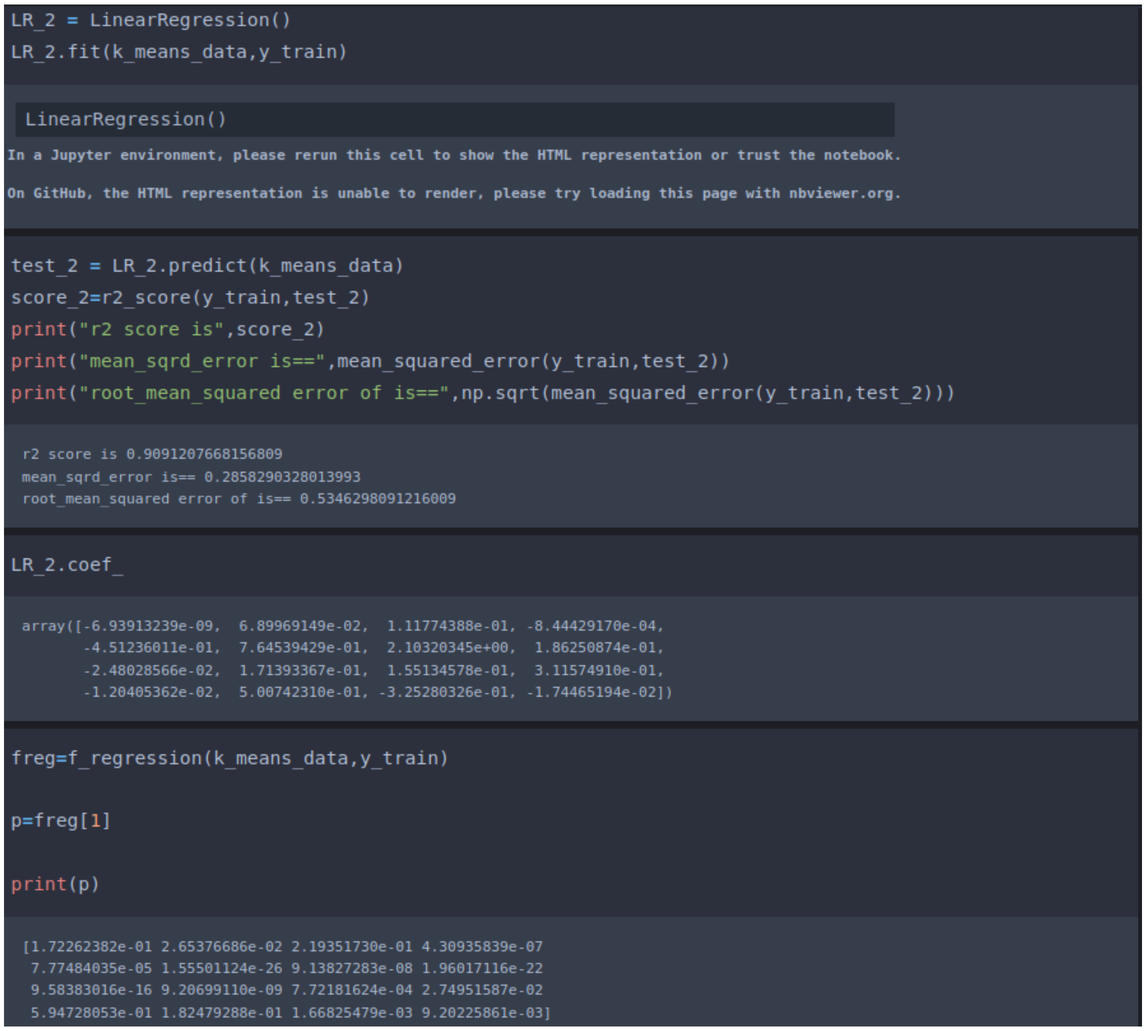
```
kmeans = KMeans(n_clusters=3).fit(features)
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[0.0091124  0.00615742 0.01756776 ... 0.06854376 0.01347682 0.05662358]
 [0.00825102 0.01940025 0.01984767 ... 0.11277687 0.01665691 0.01740856]
 [0.00749311 0.00414636 0.03533729 ... 0.08038249 0.03279952 0.01249419]]
```

```
labels = kmeans.labels_
k_means_data['sent_group'] = labels
```

The first cell instantiated and fit a k-means clustering algorithm over the TFIDF vectors. The second piece of code extracts the categorical labels from the kmeans object and assigns them to a new column of the dataframe.

The final important piece of code runs the diagnostic regression and generates summary statistics for the model.



```
LR_2 = LinearRegression()
LR_2.fit(k_means_data,y_train)
```

```
LinearRegression()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

```
test_2 = LR_2.predict(k_means_data)
score_2=r2_score(y_train,test_2)
print("r2 score is",score_2)
print("mean_sqrd_error is==",mean_squared_error(y_train,test_2))
print("root_mean_squared error of is==",np.sqrt(mean_squared_error(y_train,test_2)))
```

```
r2 score is 0.9091207668156809
mean_sqrd_error is== 0.2858290328013993
root_mean_squared error of is== 0.5346298091216009
```

```
LR_2.coef_
```

```
array([-6.93913239e-09,  6.89969149e-02,  1.11774388e-01, -8.44429170e-04,
       -4.51236011e-01,  7.64539429e-01,  2.10320345e+00,  1.86250874e-01,
       -2.48028566e-02,  1.71393367e-01,  1.55134578e-01,  3.11574910e-01,
       -1.20405362e-02,  5.00742310e-01, -3.25280326e-01, -1.74465194e-02])
```

```
freg=f_regression(k_means_data,y_train)

p=freg[1]

print(p)
```

```
[1.72262382e-01 2.65376686e-02 2.19351730e-01 4.30935839e-07
 7.77484035e-05 1.55501124e-26 9.13827283e-08 1.96017116e-22
 9.58383016e-16 9.20699110e-09 7.72181624e-04 2.74951587e-02
 5.94728053e-01 1.82479288e-01 1.66825479e-03 9.20225861e-03]
```

After this code instantiates a linear regression object denoted by LR_2 and fits it to the data, obtaining summary statistics of the regression is easily achieved by accessing various linear regression object methods.

Reflection

This project taught me many things about myself, the research project process, and natural language processing/sentiment analysis.

First of all, completing this project helped me to see how data found on the internet can be targeted, obtained, cleaned and made materially useful toward solving real world problems with the right computational methods and cleaning pipeline. In my case, I took FOMC minutes statements that existed on the Federal Reserve's website and was able to convert them into usable quantitative categorical data to solve a problem within my personal field of economics. Data itself is useless unless it can tell us something about the world, and I witnessed this first hand as I saw my data come to life once properly processed.

In my new job I start next fall, feature extraction from textual data is my primary job focus, so I also feel that this project has been instrumental in preparing me for life outside of academia. It is often easy to get swept up into the magic of learning and forget that the things you are actually learning constitute valuable skills and knowledge that are tangibly useful in the real world. One of the best skills I honed was simply my raw coding ability. It is one thing to learn coding syntax and do simple one off practice problems or assignments. It is a very different thing to wholly rely on your coding ability to create a unique solution to a real world problem. No amount of rote memorization could ever be as valuable as my learning by doing during the research project process. For this, I am incredibly grateful, as will my future self be when I encounter tough problems in the workforce.

Finally, the project forced me to develop many equally important soft skills relating to empirical idea communication and professional presentation by not only requiring me to develop code but also to communicate the value of this code to the wider scientific community. Your ideas are ultimately only as valuable as the words with which you can convey them, and this has never felt more true for me than while doing this project.

Thus, for the above reasons this research project has been uniquely valuable to my personal and professional education.