Capstone Project Report-Drafty Draft

Ryan Harty

4/21/2021

OK for draft. For final paper please select a title that indicates main question(s) or result(s)

Community Detection in Corporate Environments

Each of the references in your reference section should be cited at least once, at the first relevant spot, in the main body of your paper.

Authors: Ryan Harty, Ernest Kufuor, Aline Niyonsaba, Eric Ngabonzima

Email Addresses: rjharty@andrew.cmu.edu (mailto:rjharty@andrew.cmu.edu), ekufor@andrew.cmu.edu (mailto:ekufor@andrew.cmu.edu), aniyonsa@andrew.cmu.edu (mailto:aniyonsa@andrew.cmu.edu), engabonz@andrew.cmu.edu (mailto:engabonz@andrew.cmu.edu)

Abstract:

You should say that companies are the nodes, and also give a brief indication of how you decide whether companies are connected in the graph.

This paper seeks to explore the development of graphical structures in order to build and analyze corporate relationships using a large amount of data. Namely, we are interested in community detection and information prediction ~~through these methods~~. We have collected both news article data and financial performance data on 30 companies and the financial index fund they all belong to (the Dow Jones Industrial Average) for the purposes of graph creation. We have built methods for graph creation, as well as named entity recognition, graph combination, and node attribute prediction, though some of these are still being tweaked as we progress on the project. We have succeeded in visualizing the graphs we have created, the communities we have detected, and the analysis of these graphs as we proceed to collect even better results. Finally, we have evidence that community detection is both possible and useful in this context, but we have additional steps to take in order to successfullly predict information about our companies of interest.

This is all process stuff and doesn't actually say what results the project produced, nor any conclusions. Remember an abstract typically devotes a sentence or so to highlight the main message of each section of the paper: intro, data, methods, results, discussion.

Introduction:

There is an abundance of data in the Financial Services industry- there may be quantitative data describing a firm's individual financial performance, qualitative data describing a company's relationships to other companies both similar and dissimilar in industry type, and other data sources on anything that could affect a company's financial well-being. ~~However,~~ different datasets with different levels of structure are often used independently in machine learning tasks such as modeling and prediction, ~~when~~ even though much more could be learned from building an ensemble model that can learn to account for interactions between different data sources. In this project, we are seeking to construct knowledge graphs that provide information on the relationships between publicly-traded companies. These different knowledge graphs will each provide us with key insights into business relationships between companies, and combining them however possible should allow us to perform more accurate, informative clusters of similar companies from which we can infer different features of each company in a cluster. The applications for this are very open-ended, but some of the main ways that financial companies could benefit from this analysis are through enhancement of investment strategy and improvement in anomaly detection of companies that do not have strong relationships to many others. After conducting significant research into this topic, we have set our research questions as follows:

(1) what is a knowledge graph; and (2) what data will you use to construct them?

Always better to show than to tell

Research Question 1: How well does community detection in corporate environments work as a basis for anticipating future community formation?

Research Question 2: How can we use graphical structures governing relationships between different companies to determine future information about those companies?

Data:

Move a general discussion of networks / knowledge graphs with companies as the nodes to the introduction, and briefly mention the two criteria for edges there (new article co-mentions and price or volume correlations) That puts the goals of the project up front, and also gives you more room in the data section to concentrate on data sources and data features, rather than being sidetracked by definitions, etc.

There are two main datasets that we are incorporating here. The first is news article co-mentions data, where the goal is to identify which companies are mentioned together frequently in news articles as a basis for drawing edges in a graph. A single news article co-mention for two specified companies is considered to be a news article that mentions both companies. This is based on graph theory, where a graph G is defined by a set of edges E and nodes (or vertices) V, where nodes are connected to other nodes by edges formed between them. In the knowledge graph we'd like to build out of news article co-mentions, our nodes are going to be different companies, and an edge between two nodes will represent a high frequency of news article co-mentions between them. Since co-mentions are nondirectional (both companies are treated equally in the co-mention process), our edge set will be non-directed in this case, which means that an edge reflects an equal connection for both nodes it joins. When building our news article dataset, we decided to use the GoogleNews python package to pull in news articles from GoogleNews, which itself pulls from several different news sources in compiling online news for those who are interested in reading it. One of the main reasons for this is that we wanted to bring in news articles from numerous different sources to avoid single-publication bias affecting the relationships we notice, and another reason is the convenience with which several thousand news articles can be pulled at a time. For this project, we decided to focus on the Dow Jones Industrial Average (DJIA), a stock market index that takes into account the stock performance of the largest 30 publicly-traded companies in the United States. We set this small cap on companies to build an interpretable graph with a visually-appealing number of individually-important nodes, and to simplify and expediate downstream data-processing tasks. Below is a table displaying the news sources that GoogleNews pulled from in creating our article database:

I don't know what you are trying to say here. Do you mean you only consider companies in the DJIA? if so, then say that.

new parag

Table 1 displays

	media	count
108	Fortune	37
287	The Motley Fool	33
304	TheStreet	32
224	Quartz	29
161	MarketWatch	27
218	Profit Confidential	22
277	The Guardian	21
48	Business Wire	20
107	Forbes	19
56	CNBC	18
233	Reuters	16
183	Nasdaq	15
330	Wall Street Journal	15
252	Smarter Analyst	13
313	USA Today	12
44	Business - Insider	12
199	PCMag	11
102	FierceBiotech	11
281	The Indian Express	11
180	NPR	11

Table 1: Article Sources for News Article Data

We

From there, we conducted a GoogleNews search on the stock ticker of each company in question, and returned the top 40 news articles mentioning that company in each year from 2011-2020, and after removing duplicate articles we wound up with about 12000 news articles available for further processing, each with information on the

"top" in what sense?

date of the article, the title, the link to the article, and the full text content of the article. From there, we conducted Named Entity Recognition on the content of each article, a topic that will be more fully described in our Methods section, in order to establish a list of co-mentions between companies (with identical co-mentions allowed as long as they came from different articles). We then summed the identical co-mentions to develop a list of co-mention frequencies- the number of times each pair of companies had been mentioned together in the same article. This list of co-mention frequencies was ultimately stored as a 31 x 31 matrix, with each of the 31 rows and columns corresponding to a single company node (each company was deemed to have 0 co-mentions with itself for ease of usage, so the diagonal of the matrix was 0). Finally, when any two companies had a co-mention frequency greater than 6 (chosen as the cutoff to allow a visually-interpretable graph), an edge was drawn between the nodes representing those two companies in the graph that was drawn. The resulting knowledge graph was treated as our first dataset.

This is all methods and should be re-written and moved to the methods section.

Our second main dataset was in terms of stock price correlations and transaction volume correlations between publicly-traded companies. The goal of this dataset was to build a knowledge graph using more quantitative financial relationship data rather than a graph constructed on news article co-mentions. To do this, we pulled stock price and transaction volume data from Yahoo Finance data, with our dataset also ranging from 2011-2020 and utilizing the same 30 companies used in the first dataset (as well as the DJIA index itself, which we treated as a company for research purposes). Both the stock price and transaction volume were calculated for each business day in the time period, adding up to about 264 days' worth of data per year per company. The recorded stock price and transaction volume were those measured at the close of business each day, giving us a time series dataset of each of these financial variables over time for each company. Overall, each of the companies had about 2600 data points on stock price and volume available, and with these data points we were able to calculate the change in price for each company on each day as well. Below is a graphic showing the scatterplot for the correlation between JP Morgan and Goldman Sachs that shows the process behind finding the correlation coefficient for each pair of companies:

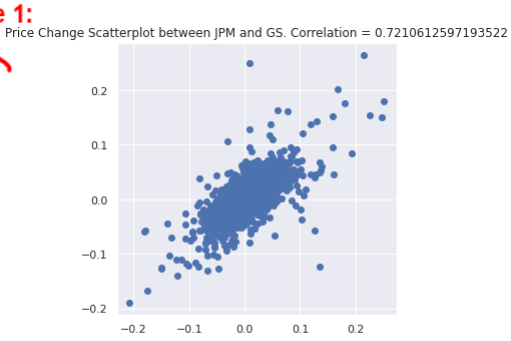
The graph is a result, not a data set.

Figure 1 shows

The graph does not show a process

(tell reader why it is important to see this graph)

Figure 1:



Correlation Between JP Morgan and Goldman Sachs

Methods:

(put process stuff from data section here)

One very important step in our process regarding our first dataset was to conduct Named Entity Recognition on the content of the articles we pulled in order to generate a list of co-mentions from our list of articles. The process here was to search the content of each article for entities, which in our case were companies represented by their stock ticker symbols. For each article, there was guaranteed to be at least one entity of interest since our news articles were pulled based on the companies we were interested in, but given that business articles usually discuss competition there were often mentions of competitors in a given company's articles. We used the flair package in python to perform Named Entity Recognition on the article content, since the flair package is a recently-developed solution that outperforms many older solutions for this process by utilizing a neural language model to assign tags to text data and learn which words in an article count as entities. After performing this Named Entity Recognition

move to spot on next page and explain how com-petitor men-tions affected graph con-struc-tion

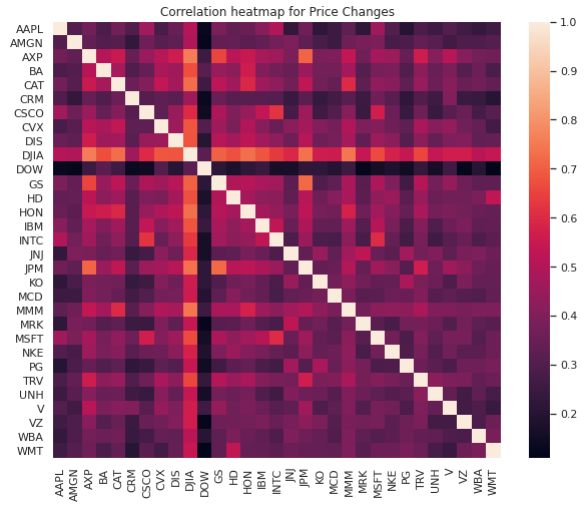
for each article, we removed corporate entities that were not listed in our 31 entities of interest (the 30 companies currently in the DJIA, plus the DJIA itself) so that our graphs would be visually interpretable. The next step was to take the entities found in each article and create a list of co-mentions for each pair of entities found in the article; so, if 6 entities were found in an article, there would be 15 different co-mentions returned in order to connect all the entities found in that article to each other. After standardizing the co-mentions so that there were not separate co-mentions for a company and its stock ticker, the large list of co-mentions (about 3000 co-mentions with the two companies, the date, and the link to the article attached) was passed along to the next overarching step- knowledge graph creation. Shown below is the database containing these co-mentions before any removal of duplicate and non-interest entities was applied, to give a sense of why these measures are important and also a feel for the data:

	from	to	title	date
0	AXP	American Express	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
1	AXP	American Express Company	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
2	AXP	Costco	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
3	AXP	Deutsche Bank	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
4	AXP	MarriottStarwood	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
...	...	...	...	...
73	USDA	Walmart	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016
74	USDA	WillardBishop	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016
75	WMT	Walmart	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016
76	WMT	WillardBishop	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016
77	Walmart	WillardBishop	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016

84760 rows × 4 columns

Table 2: Example Data for News Article Co-Mentions

Knowledge graph creation was a requirement for both datasets, as we were seeking to perform community detection and prediction using graphical data. After completing Named Entity Recognition on the first dataset, we had a list of about 3000 instances of co-mentions in news articles between our 31 entities of interest. From there, we summed identical co-mentions to develop a list of co-mention frequencies- the number of times each pair of companies had been mentioned together across our entire dataset of articles. This list of co-mention frequencies was ultimately stored as a 31 x 31 matrix, with each of the 31 rows and columns corresponding to a single company node (each company was deemed to have 0 co-mentions with itself for ease of usage, so the diagonal of the matrix was 0). Finally, when any two companies had a co-mention frequency greater than 6 (chosen as the cutoff to allow a visually-interpretable graph), an edge was drawn between the nodes representing those two companies in the graph that was drawn. The resulting knowledge graph was treated as our first dataset. To create the knowledge graphs for the financial dataset, we began with the stock price and transaction volume for each business entity of interest. To aggregate this data effectively, we calculated both the correlation in price change and the correlation in transaction volume for each pair of companies over the time period being measured, giving us two 31 x 31 matrices of correlation coefficients, one for price changes and one for transaction volume. These 31 x 31 correlation matrices are how we created our edges for this dataset- if you assign a number to each corporate entity we measured, then each entry in the matrix would reference one company by its row index and one company by its column index, so each of the calculated correlation coefficients became an entry in the corresponding graph. For each of the 31 company nodes in the graph, we drew an edge between the nodes if there was a moderate (0.40-0.60) correlation between the companies for both price changes and transaction volume, or a strong correlation (>0.60) for either of the metrics. The resulting knowledge graph was our second dataset, and the 31 x 31 matrix for price changes is shown below to aid in comprehension:



Why not show the volume correlation heat map also?

Figure 2: Correlation Matrix for Companies of Interest

Three more methods are currently being developed for this research project. The first is graph combination, where multiple graphs are fused through the use of edge regulation metrics or development of a multiplex graph. By using edge regulation metrics, two graphs with identical nodes can be combined into one graph by strictly delineating the requirements for an edge to be formed between two nodes. Since our knowledge graphs rely on news article co-mentions and financial correlations, we are currently testing different metrics for edge regulation to ensure the combined graph that we form will perform as well as possible in downstream community detection tasks. This part of our methods will be much better fleshed-out when we finish developing community detection methods and their evaluation tasks. Multiplex graphs are singular graphs which combine multiple knowledge graphs by joining common nodes in the two graphs, and the edges connecting the nodes in separate graphs are an additional part of the graphical structure. The multiplex graph construction is also currently in development, and we are weighing whether it will be better suited to a visual enhancer for our project or a viable solution for community detection and prediction. A sample multiplex graph we have constructed is shown below:

as shown in Fig 3 for example

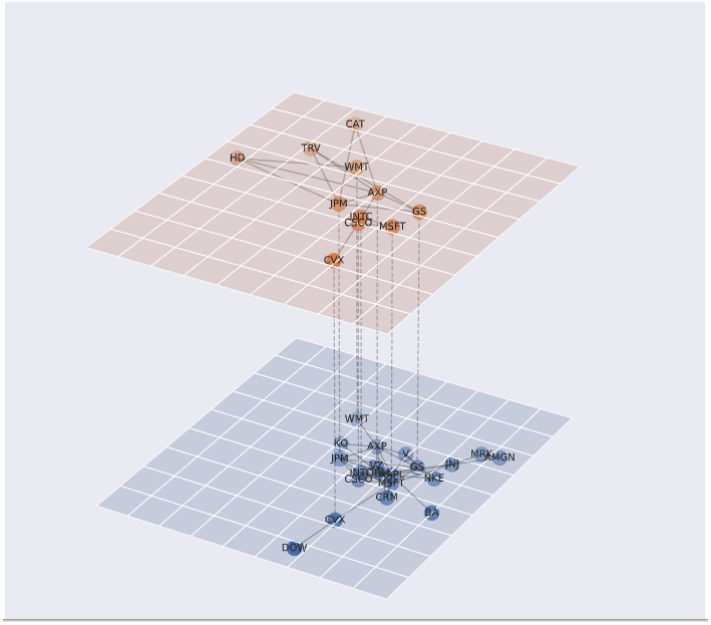


Figure 3: Sample Multiplex Graph

The second method mentioned above is community detection, where communities of nodes are formed from a knowledge graph to give us insight into which nodes are more connected than others. We detected communities by searching for groups of nodes within a graph which all had at least n edges among the nodes selected. After evaluation of which values of n would provide us with useful cliques for detecting communities, we then split the

didn't you also do eigen-vector cen-trality detec-tion?

Are you using any other community detection methods (such as random walk, etc.)?

graph into the k cliques that result from the choice n and then compare them to known corporate communities to evaluate whether we have established a new community on the basis of news mentions and financial correlation. These evaluations are very much under development and review as this is a temporary draft.

The final method of interest here is Node Attribute Prediction, where we take information about a node, combine it with information built into the graph we have built, and use it to prediction future information about that node. Currently, this is being developed on the financial data graph, using previous stock prices as the data we are aiming to predict- in particular, whether a certain stock will increase or decrease in price in the next available time period. There are two approaches we are pursuing here- one involves utilizing Graphical Neural Networks, with our node information as a node attribute, to train a black-box statistical model that tends to predict accurately (if not comprehensibly) whether the stock price will increase. Another option is to convert each node's graphical information into a vectorized format using a python package such as Node2Vec, which creates a representation of each node's place in the graph that can be easily combined with the price information for each node to form a feature set that can be modeled on. Both of these techniques will involve the labeling of price data and the development and training of a statistical model that can predict well even in the presence of only 30 different companies.

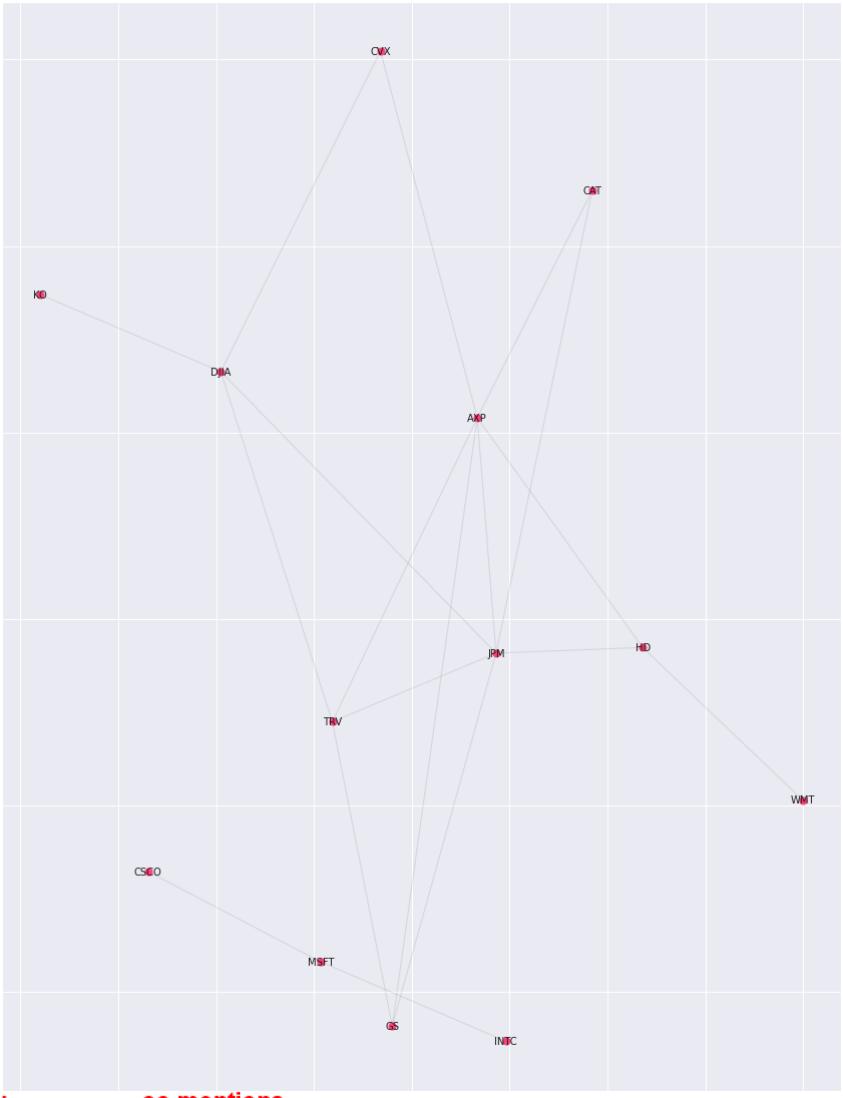
what information?

Results:

graph based on co-mentions in news articles

Our results are almost completely under construction at this point, as we gather intermediate results, use them to refine the methods, and rinse and repeat as our clients guide us. However, we have made several achievements in results gathering, and I will highlight them in succession. The first achievement is the construction of knowledge graphs from our data- we have been able to build both news article and financial data graphs, and visualize them as well. Shown below is the visualization of a news article graph created with a selection of the entities of interest, using stricter edge criteria than normal in order to show the graph more clearly in this stage. The letters on each dot correspond to a company's stock ticker, so MSFT corresponds to Microsoft, and CAT to Caterpillar, for example.

(1) why not show the actual graph you came up with? There is graph layout software to help with this
(2) also please show the graph based on financial data correlations



show graph based on financial data correlations also

please make the layout of the nodes be the same in both graphs, so we can clearly see differences in the edges in one graph vs the other

Figure 4: co-mentions.

News Articles Graph- First Pass

We also have several results on community detection. Using $n = 2$ in our community detection process, we were able to identify two distinct communities in a later version of our news article graph. Only the largest community is shown below, but we can see that there are both technology and financial companies included in this community, with larger bars on the right-hand side corresponding to more central nodes in the graph we are analyzing. So, we do have some progress in graph analysis, though we are looking to make even bigger strides.

co-mentions

in Figure 5

ok

how are you measuring this? I didn't see this in the methods section.

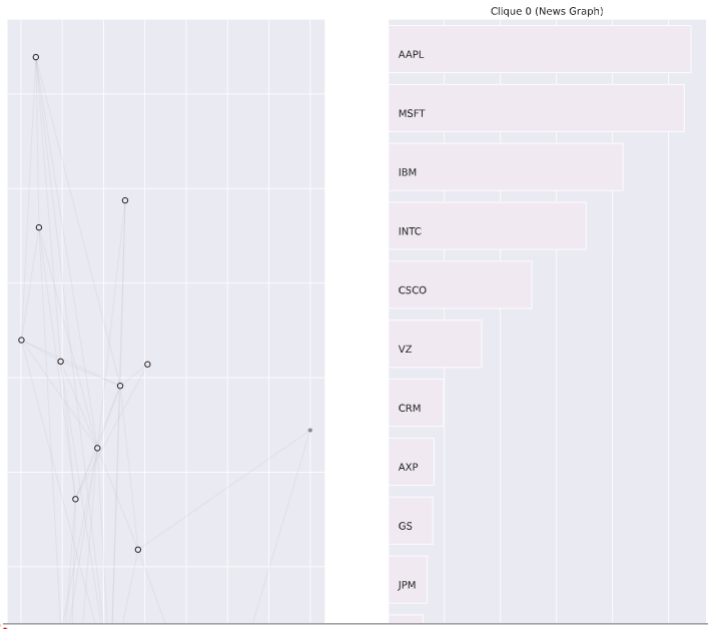


Figure 5: Community Detection in the News Article Graph

co-mentions

~~Shown below~~ Figure 6 shows is the largest community found in the financial data graph- and this one seems to show a much bigger breadth of industries included. Home goods, construction equipment, and insurance all appear here, highlighting the potential behind this approach- we may not have thought to look for connections between these industries if not for these graphical structures we created.

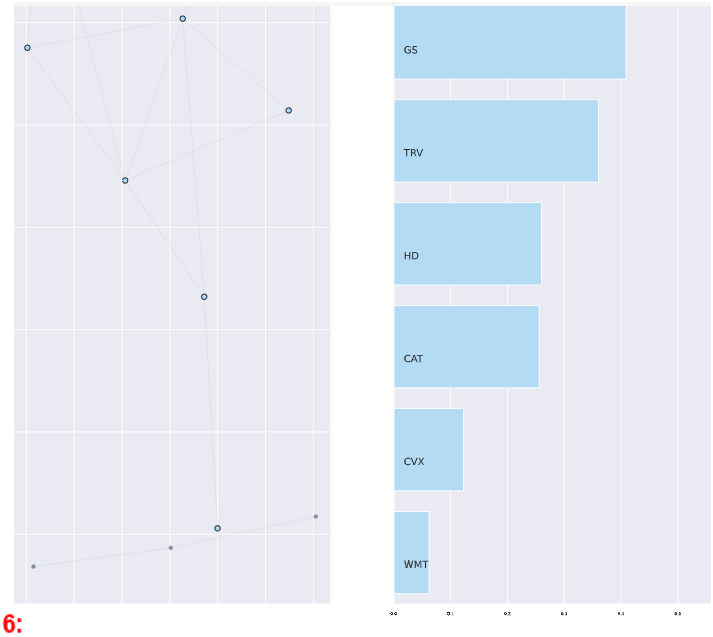


Figure 6: Community Detection in the Financial Data Graph

In Figure 7

We also have results on which entities are most central to the graphs we create. ~~Here,~~ we can see that in the news article data graph, we have tech companies dominating- Apple, Microsoft, IBM, Intel, Cisco, Verizon, and Salesforce are all tech companies before we arrive at American Express. In the graphical structure we have created, there is evidence of tech companies being more dominant in the news than financial companies such as American Express and Goldman Sachs, as the more central nodes have more connections to other nodes in the graph and thus are related to more companies that we have sampled.

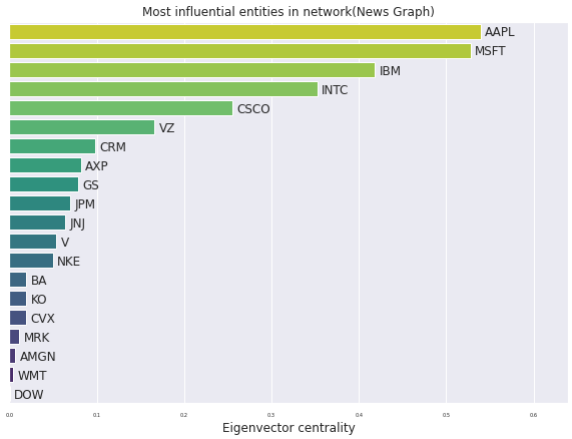


Figure 7 Eigenvector Centrality in the News Article Graph

In figure 8,

Here we notice a much different effect- in the financial data graph, financial companies are connected to many more nodes than other companies, which makes sense as they must invest in different industries. So, we have evidence that our two distinct knowledge graphs are going to provide us with two different perspectives on connections between companies of interest, and thus that our methods may be worth it for their potential applications to new data.

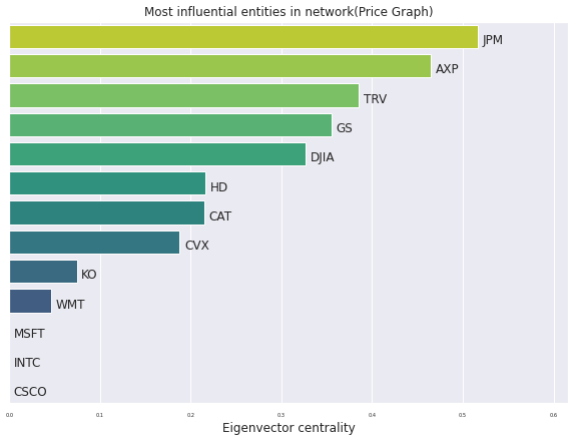


Figure 8 Eigenvector Centrality in the Financial Data Graph

Discussion:

good so far. Looking forward to final, complete, report.

When it comes to community detection, we have made significant progress. We are able to detect communities in the graphs we have created, and the only step that remains is to test these methods on combination graphs that we are creating. We are still in the process of evaluating these detected communities using some form of real-life data on which to compare, but the completion of this tasks seems to be closer in the pipeline rather than further. We may need to make some tweaks to our data pipeline as well as our graph creation methods in order to detect the communities that show up most often in other datasets, but there is significant progress here, as well as very good signs that our research in this subject will prove fruitful in detecting communities that are not easily detectable to a human being given the large amount of data we are pulling in.

Predicting new information is further off for us as a team. While we have made progress in Node Attribute Prediction, we are currently in the modeling stage for this data, so we are not yet able to say whether our graphs and communities are well-suited to predicting new information about their member nodes. This is our number one priority going forward and our main focus, but evaluation may be tricky- while we can likely develop strong accuracy levels given the right statistical model and evaluate them on a test set, we may have trouble discerning the graphical contribution to these results. We may need some creativity in order to measure this technique's performance against more standard machine learning methods.

References

Good start on references. Make sure formatting agrees with ASA style. (it does not now)

Anon. 2021. “PyTorch Geometric Documentation¶.” PyTorch Geometric Documentation - pytorch_geometric 1.7.0 Documentation. Retrieved April 23, 2021 (https://pytorch-geometric.readthedocs.io/en/latest/ (https://pytorch-geometric.readthedocs.io/en/latest/)).

Ferencz, Marcell. 2020. “Building a Social Network from the News Using Graph Theory by Marcell Ferencz.” Medium. Retrieved April 23, 2021 (https://towardsdatascience.com/building-a-social-network-from-the-news-using-graph-theory-by-marcell-ferencz-9155d314e77f (https://towardsdatascience.com/building-a-social-network-from-the-news-using-graph-theory-by-marcell-ferencz-9155d314e77f)).

Kung-Hsiang, Huang (Steeve). 2019. “Hands on Graph Neural Networks with PyTorch & PyTorch Geometric.” Medium. Retrieved April 23, 2021 (https://towardsdatascience.com/hands-on-graph-neural-networks-with-pytorch-pytorch-geometric-359487e221a8 (https://towardsdatascience.com/hands-on-graph-neural-networks-with-pytorch-pytorch-geometric-359487e221a8)).

Leskovec, Jure. 2016. “Scalable Feature Learning for Networks.” node2vec. Retrieved April 23, 2021 (https://snap.stanford.edu/node2vec/ (https://snap.stanford.edu/node2vec/)).

Add references for the various python packages you used.

Techical Appendix

Obtaining a list of equities tickers and the corresponding timeseries

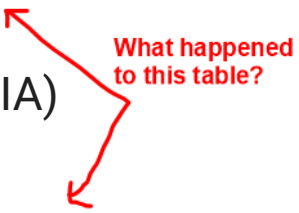
```
from google.colab import drive
drive.mount('/content/drive')
#newsframe.to_csv('/content/drive/My Drive/dow_raw_2011.csv')
import random
import numpy as np
import pandas as pd

import seaborn as sns
sns.set(rc={'figure.figsize':(10,8)})
import matplotlib.pyplot as plt

import pandas_datareader.data as web
from datetime import datetime
```

```
Mounted at /content/drive
```

+ Code + Text

```
# Configure filepaths for data storage
#data_dir = "data"
#if not os.path.exists(data_dir):
#    os.makedirs(data_dir)
```

	High	Low	Open	Close	Volume	Adj Close
Date						



Get list of companies in Dow-Jones Industrial Average (DJIA)

We will obtain and store the list of the constituents from Wikipedia.

2019-03-22	49.950001	48.160000	48.799999	48.599998	844700.0	43.245010

```
today = datetime.today().strftime('%Y-%m-%d')
print(today)

data    = pd.read_html('https://en.wikipedia.org/wiki/Dow_Jones_Industrial_Average#Components
table   = data[1]
symbols = list(table.Symbol.values) + list(['DJIA'])

df = pd.DataFrame(symbols, columns=['Symbol'])

print("{} symbols in total".format(len(df)))
```

```
2021-04-19
31 symbols in total
```

Download price timeseries

Now that we have a list of ticker symbols, we can download the corresponding timeseries from Yahoo Finance.

```
start_date = '2000-01-01'
end_date  = '2020-12-31'

#start_date = '2000-01-01'
#end_date  = '2020-12-31'

symbols = sorted(symbols)
print("Downloading {} files".format(len(symbols)))
for i, symbol in enumerate(symbols):
    try:
        df = web.DataReader(symbol,'yahoo', start_date, end_date)
        df = df[['Adj Close','Volume']]
        #df.to_csv(os.path.join(data_dir, "{}.csv".format(symbol)))
        df.to_csv('/content/drive/My Drive/JPM_financial_data/' + "{}.csv".format(symbol))
    except KeyError:
      print("Error for {}".format(symbol))
      pass
print("Stored {} files".format(i+1))
```

```
Downloading 31 files
Stored 31 files
```

```
# Printing a sample dataframe
idx = np.random.randint(len(symbols))
print("History for {}".format(symbols[idx]))
df = pd.read_csv('/content/drive/My Drive/JPM_financial_data/' + symbols[idx]+".csv").set_ind
df.head()
```

```
History for INTC
```

Date	Adj Close	Volume
2000-01-03	27.169121	57710200.0
2000-01-04	25.900448	51019600.0
2000-01-05	26.115145	52389000.0
2000-01-06	24.592737	55171200.0
2000-01-07	25.607666	41096400.0

Pre-process Financial Timeseries

```
index = pd.date_range(start=start_date, end=end_date, freq='D')     # initialize an empty Dat
df_price = pd.DataFrame(index=index, columns=symbols)               # initialize empty datafr
df_volume = pd.DataFrame(index=index, columns=symbols)
```

```
# Aggregate all symbols into a price dataframe and volume dataframe
for symbol in symbols:
    symbol_df = pd.read_csv('/content/drive/My Drive/JPM_financial_data/' + symbol+".csv").se
    symbol_df.index = pd.to_datetime(symbol_df.index)

    df_price[symbol] = symbol_df['Adj Close']
    df_volume[symbol] = symbol_df['Volume']
```

Removing NaNs

```
# Let's drop the dates where all the stocks are NaNs, ie., weekends/holidays where no trading
df_price.dropna(how='all', inplace=True)
df_volume.dropna(how='all', inplace=True)
assert((df_price.index == df_volume.index).all())
```

Let's see how many NaNs exist at this point, as some companies would've gone public later than others (so they won't have equally long price histories)

```
pd.isnull(df_price).sum()
```

```
AAPL       0
AMGN       0
AXP        0
BA         0
CAT        0
CRM     1122
CSCO       0
CVX        0
DIS        0
DJIA       0
DOW     4832
GS         0
HD         0
HON        0
IBM        0
INTC       0
JNJ        0
JPM        0
KO         0
MCD        0
MMM        0
MRK        0
MSFT       0
NKE        0
PG         0
TRV        0
UNH        0
V       2063
VZ         0
WBA        0
WMT        0
dtype: int64
```

```
# Backfill NaNs followed by forward fill
df_price = df_price.bfill(axis='rows')
df_price = df_price.ffill(axis='rows')
```

```
pd.isnull(df_price).sum()
```

```
AAPL    0
AMGN    0
AXP     0
BA      0
CAT     0
CRM     0
CSCO    0
CVX     0
```

```
DIS     0
DJIA    0
DOW     0
GS      0
HD      0
HON     0
IBM     0
INTC    0
JNJ     0
JPM     0
KO      0
MCD     0
MMM     0
MRK     0
MSFT    0
NKE     0
PG      0
TRV     0
UNH     0
V       0
VZ      0
WBA     0
WMT     0
dtype: int64
```

Storing the cleaned dataframes

```
df_price.to_csv('/content/drive/My Drive/JPM_financial_data/' + "prices.csv")
df_volume.to_csv('/content/drive/My Drive/JPM_financial_data/' + "volume.csv")
```

Obtain Percentage Change and Correlation

We need to convert prices to percent change in price as opposed to the actual $ price. This is because stocks with very similar prices can behave very differently and vice-versa. For e.g., if a stock moves from $100 to 110$, we want the price column to say 10% (indicating the change).

However, for volume, we will retain magnitude.

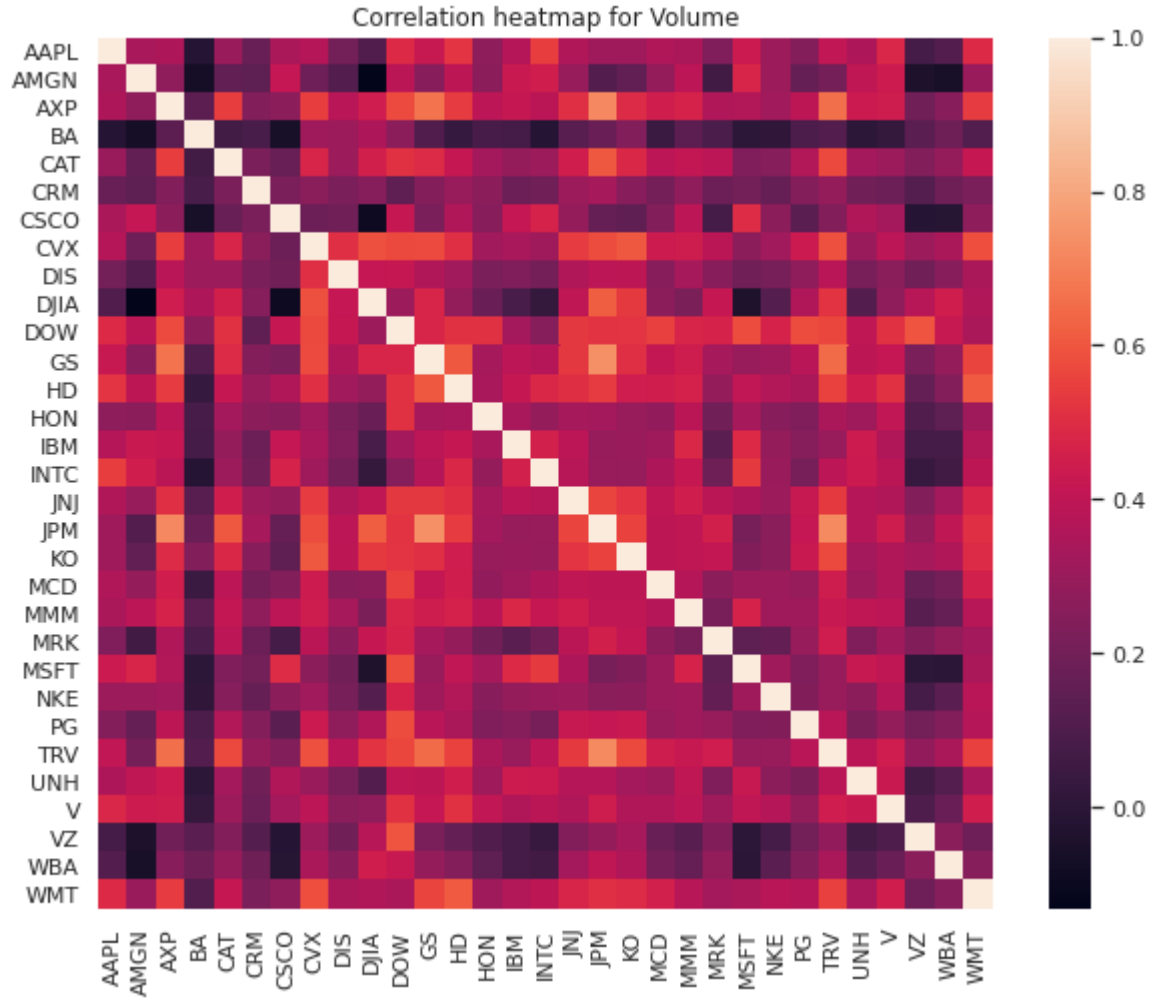
```
df_price_pct = df_price.pct_change().dropna(how='all')
df_volume_pct = df_volume.pct_change().dropna(how='all')
```

Calculate Correlations

```
price_corr = df_price_pct.corr()
volume_corr = df_volume.corr()
```

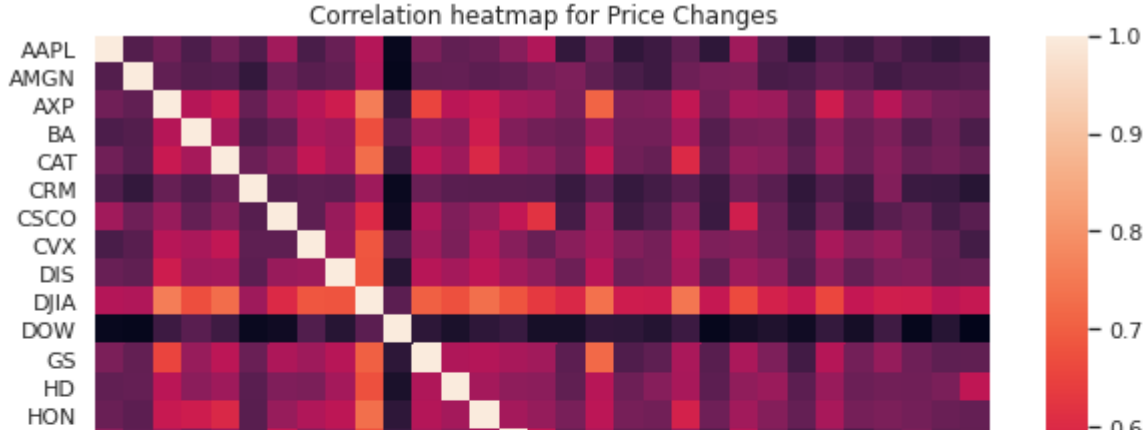
```
sns.heatmap(volume_corr).set_title("Correlation heatmap for Volume")
```

```
Text(0.5, 1.0, 'Correlation heatmap for Volume')
```




```
sns.heatmap(price_corr).set_title("Correlation heatmap for Price Changes")
```

```
Text(0.5, 1.0, 'Correlation heatmap for Price Changes')
```



what happened to this figure?

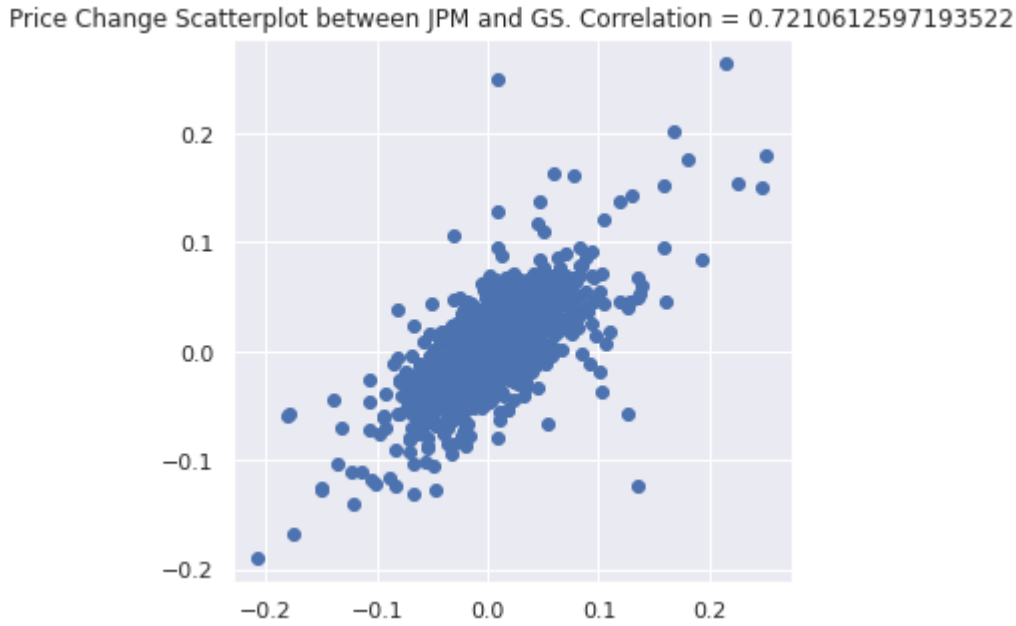
Examining correlations further

As we can see above, the price histories of JP Morgan (JPM) and Goldman Sachs (GS) are highly correlated. This makes sense as they are both financial sector stocks. That forms the motivation behind using correlation to form the graph.

This can be observed in the plots below.

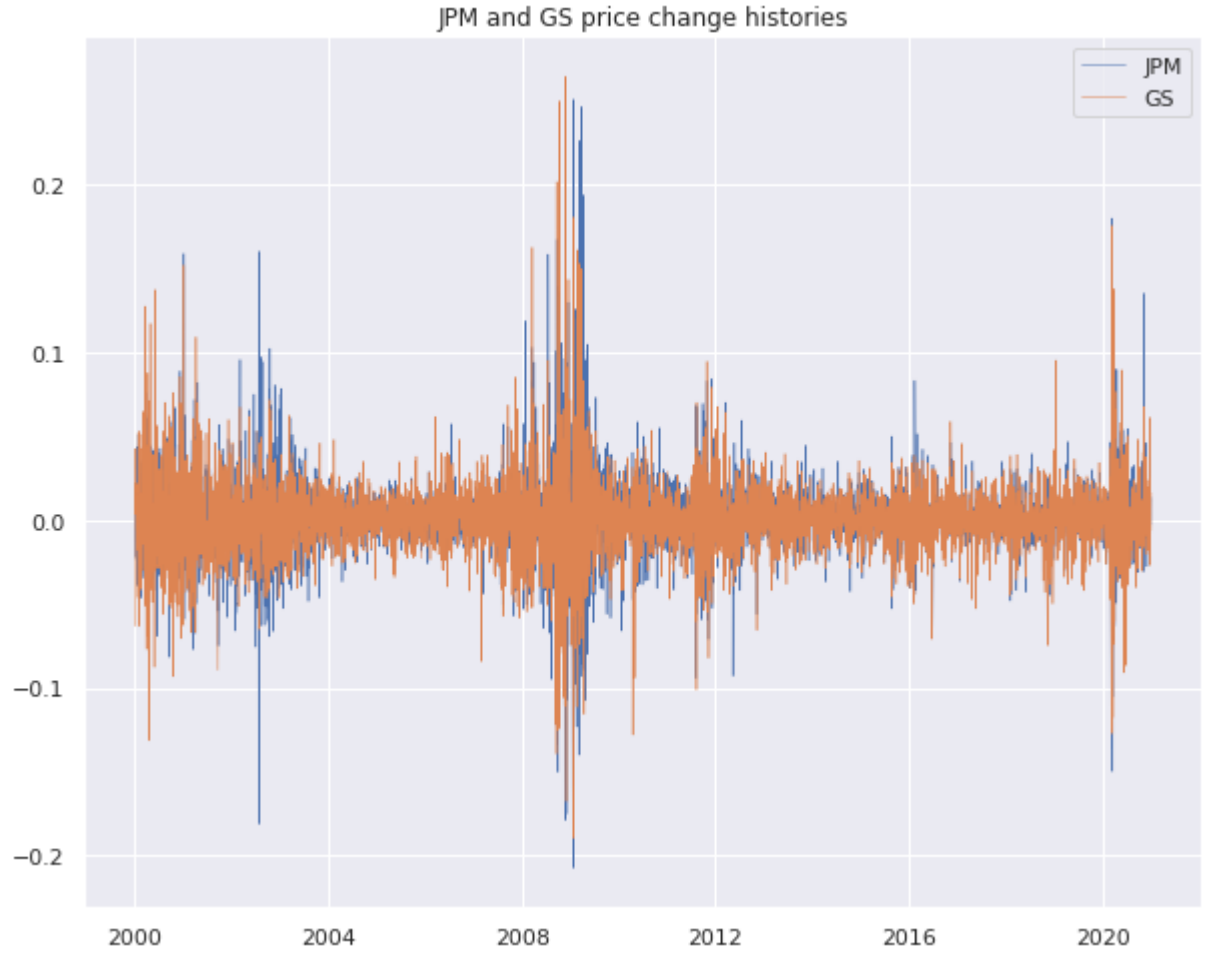

```
plt.figure(figsize=(5,5))
plt.scatter(df_price_pct['JPM'], df_price_pct['GS'])
plt.title('Price Change Scatterplot between JPM and GS. Correlation = {}'.format(price_corr['
```

```
Text(0.5, 1.0, 'Price Change Scatterplot between JPM and GS. Correlation = 0.72106
```




```
plt.title('JPM and GS price change histories')
plt.plot(df_price_pct['JPM'], label='JPM', linewidth=0.5)
plt.plot(df_price_pct['GS'], label='GS', linewidth=0.5)
plt.legend()
```

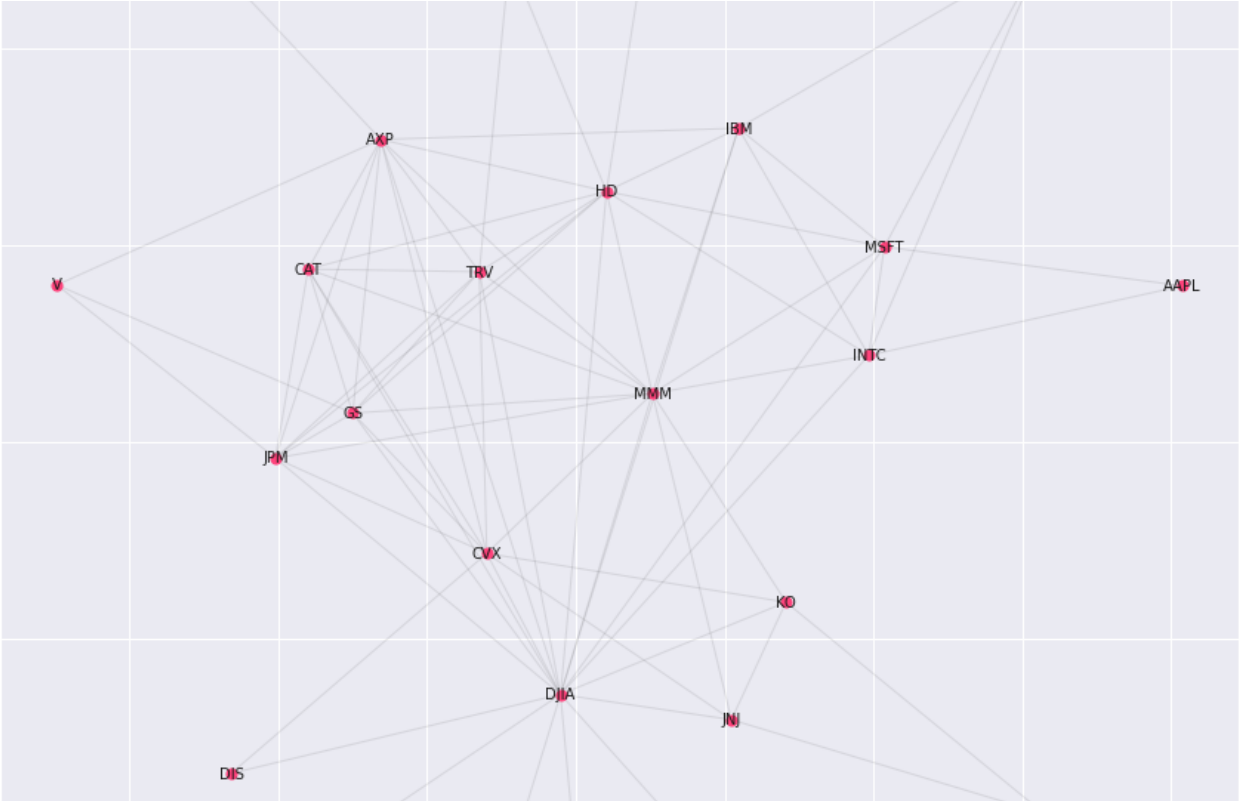
```
<matplotlib.legend.Legend at 0x7facdd7f17d0>
```



Saving correlation dataframes

```
df_price_pct.to_csv('/content/drive/My Drive/JPM_financial_data/' + 'price_pct.csv')
price_corr.to_csv('/content/drive/My Drive/JPM_financial_data/' + 'price_corr.csv')
volume_corr.to_csv('/content/drive/My Drive/JPM_financial_data/' + 'volume_corr.csv')
```

You can now use these correlations to create graphs.



Node Attribute Prediction

```
!pip install -q torch-scatter -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html
!pip install -q torch-sparse -f https://pytorch-geometric.com/whl/torch-1.8.0+cu101.html
!pip install -q torch-geometric

import torch
import torch_geometric.utils as tgu

nx.set_node_attributes(gr_price, df_change.loc[0:4527].to_dict('series'), name="x")
tgu.from_networkx(gr_price)
djia_31 = pd.Series(['AXP','AAPL','AMGN','BA','CAT','CRM','CSCO','CVX','DIS','DJIA','DOW',
                     'GS','HD','HON','IBM','INTC','JNJ','JPM','KO','MCD','MMM',
                     'MRK','MSFT','NKE','PG','TRV','UNH','V','VZ','WBA','WMT',])
#try ^DJI in yahoo finance-check whether DJIA is working
#fix adjacency matrix to be of size 31 x 31
#we want all nodes- some may have no edges
#try passing correlations as edge weights to see if it improves performance
djia_label= pd.Series([1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                       1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                       0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1])
pct_returns= pd.Series([0.25, 0.44, 0.24, 0.20, 0.01, 0.19, 0.16, 0.12, 0.20, 0.22, 0.15,
                        0.26, 0.38, 0.34, 0.33, 0.12, 0.02, 0.23, 0.18, 0.23, -0.12,
                        0.12, 0.38, 0.28, 0.38, 0.26, -0.12, 0.31, 0.11, -0.17, 0.29])
#could standardize to quantiles


label_dict = {djia_31[entry]: djia_label[entry] for entry in range(len(djia_label))}
```

```
label_dict = {ajia_si[ent]: ajia_label[ent] for ent in range(len(ajia_label))}

nx.set_node_attributes(gr_price, label_dict, name="y")
dataset=tgu.from_networkx(gr_price)



#df_change.loc[4520:4530]
dataset.x.size()

     torch.Size([26, 4528])


from torch_geometric.nn import GCNConv
from torch.nn import Linear
import torch.nn.functional as F

%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

def visualize(h, color):
    z = TSNE(n_components=2).fit_transform(out.detach().cpu().numpy())

    plt.figure(figsize=(10,10))
    plt.xticks([])
    plt.yticks([])

    plt.scatter(z[:, 0], z[:, 1], s=70, c=color, cmap="Set2")
    plt.show()

class GCN(torch.nn.Module):
    def __init__(self, hidden_channels):
        super(GCN, self).__init__()
        torch.manual_seed(12345)
        self.conv1 = GCNConv(4528, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, 2)

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = x.relu()
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.conv2(x, edge_index)
        return x

model = GCN(hidden_channels=16)
print(model)

model = GCN(hidden_channels=16).double()
model.eval()

out = model(dataset.x.double(), dataset.edge_index)
visualize(out, color=dataset.y)
```

```
dataset.y
```

```
GCN(
  (conv1): GCNConv(4528, 16)
  (conv2): GCNConv(16, 2)
)
```

what is this?

```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
        1, 1])
```

```
from IPython.display import Javascript  # Restrict height of output cell.
from sklearn.model_selection import ShuffleSplit
display(Javascript('''google.colab.output.setIframeHeight(0, true, {maxHeight: 300})'''))

model = GCN(hidden_channels=16).double()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)
criterion = torch.nn.CrossEntropyLoss()

def train():
      model.train()
      optimizer.zero_grad()  # Clear gradients.
      out = model(dataset.x.double(), dataset.edge_index)  # Perform a single forward pass.
      loss = criterion(out, dataset.y)  # Compute the loss solely based on the training nodes
```

```
Epoch: 116, Loss: 0.2976
Epoch: 117, Loss: 0.3057
Epoch: 118, Loss: 0.2962
Epoch: 119, Loss: 0.3070
Epoch: 120, Loss: 0.2677
Epoch: 121, Loss: 0.2841
Epoch: 122, Loss: 0.2858
Epoch: 123, Loss: 0.2873
Epoch: 124, Loss: 0.2873
Epoch: 125, Loss: 0.2854
Epoch: 126, Loss: 0.2783
Epoch: 127, Loss: 0.2890
Epoch: 128, Loss: 0.2868
Epoch: 129, Loss: 0.2959
Epoch: 130, Loss: 0.2890
Epoch: 131, Loss: 0.2833
Epoch: 132, Loss: 0.2802
Epoch: 133, Loss: 0.2915
Epoch: 134, Loss: 0.2847
Epoch: 135, Loss: 0.2788
Epoch: 136, Loss: 0.2660
Epoch: 137, Loss: 0.2722
Epoch: 138, Loss: 0.2787
Epoch: 139, Loss: 0.2677
Epoch: 140, Loss: 0.2696
Epoch: 141, Loss: 0.2825
Epoch: 142, Loss: 0.2763
Epoch: 143, Loss: 0.2595
Epoch: 144, Loss: 0.2971
Epoch: 145, Loss: 0.2922
Epoch: 146, Loss: 0.2665
Epoch: 147, Loss: 0.2868
Epoch: 148, Loss: 0.2915
Epoch: 149, Loss: 0.2776
Epoch: 150, Loss: 0.2824
Epoch: 151, Loss: 0.2632
Epoch: 152, Loss: 0.2624
Epoch: 153, Loss: 0.2710
Epoch: 154, Loss: 0.2841
Epoch: 155, Loss: 0.2856
Epoch: 156, Loss: 0.2747
Epoch: 157, Loss: 0.2686
Epoch: 158, Loss: 0.3089
Epoch: 159, Loss: 0.3057
Epoch: 160, Loss: 0.2829
Epoch: 161, Loss: 0.2746
```

```
test_acc = test()
print(f'Test Accuracy: {test_acc:.4f}')

model.eval()

out = model(dataset.x, dataset.edge_index)
visualize(out, color=dataset.y)
```

```
Test Accuracy: 0.8846
```

what is this?

```
!pip install node2vec
from node2vec import Node2Vec
n2v = Node2Vec(gr_price)
n2v_model = n2v.fit(window=10, min_count=1, batch_words=4)
```

```
Requirement already satisfied: node2vec in /usr/local/lib/python3.7/dist-packages (0.4.3
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (from no
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from nod
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from node
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: decorator<5,>=4.3 in /usr/local/lib/python3.7/dist-packag
Computing transition probabilities: 100%  26/26 [00:00<00:00, 36.60it/s]
Generating walks (CPU: 1):  20%|██        | 2/10 [00:00<00:00, 15.40it/s]
Generating walks (CPU: 1): 100%|██████████| 10/10 [00:01<00:00,  9.54it/s]
```

```
n2v_model.wv.get_vector('AAPL')
```

```
df_newie = pd.DataFrame(columns=('Node', 'X', 'Y'))
```

1. Import Packages

```
# install flair
!pip install flair

# load basic packages
import pandas as pd
from itertools import combinations, product
import string
import re
import matplotlib.pyplot as plt
import seaborn as sns

from tqdm import tqdm

# load Flair and NLTK
import torch
from flair.data import Sentence
from flair.models import SequenceTagger
from nltk import tokenize

import nltk
nltk.download('punkt')

# is cuda available?
torch.cuda.is_available()
```

```
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.7/dist-pack
Building wheels for collected packages: gdown
  Building wheel for gdown (PEP 517) ... done
  Created wheel for gdown: filename=gdown-3.12.2-cp37-none-any.whl size=9693 sha256=3
  Stored in directory: /root/.cache/pip/wheels/81/d0/d7/d9983facc6f2775411803e0e2d30e
Successfully built gdown
Building wheels for collected packages: ftfy, sqlitedict, langdetect, segtok, mpld3,
  Building wheel for ftfy (setup.py) ... done
  Created wheel for ftfy: filename=ftfy-6.0-cp37-none-any.whl size=41622 sha256=5ba08
  Stored in directory: /root/.cache/pip/wheels/22/8b/08/7d1c17849e10371206a262304973b
  Building wheel for sqlitedict (setup.py) ... done
  Created wheel for sqlitedict: filename=sqlitedict-1.7.0-cp37-none-any.whl size=1437
```

```
  Created wheel for sqlitedict: filename=sqlitedict-1.7.0-cp37-none-any.whl size=1437
  Stored in directory: /root/.cache/pip/wheels/cf/c6/4f/2c64a43f041415eb8b8740bd80e15
  Building wheel for langdetect (setup.py) ... done
  Created wheel for langdetect: filename=langdetect-1.0.8-cp37-none-any.whl size=9931
  Stored in directory: /root/.cache/pip/wheels/8d/b3/aa/6d99de9f3841d7d3d40a60ea06e6d
  Building wheel for segtok (setup.py) ... done
  Created wheel for segtok: filename=segtok-1.5.10-cp37-none-any.whl size=25019 sha25
  Stored in directory: /root/.cache/pip/wheels/b4/39/f6/9ca1c5cabde964d728023b5751c3a
  Building wheel for mpld3 (setup.py) ... done
  Created wheel for mpld3: filename=mpld3-0.3-cp37-none-any.whl size=116679 sha256=0b
  Stored in directory: /root/.cache/pip/wheels/c0/47/fb/8a64f89aecfe0059830479308ad42
  Building wheel for overrides (setup.py) ... done
  Created wheel for overrides: filename=overrides-3.1.0-cp37-none-any.whl size=10174
  Stored in directory: /root/.cache/pip/wheels/5c/24/13/6ef8600e6f147c95e595f1289a86a
  Building wheel for sacremoses (setup.py) ... done
  Created wheel for sacremoses: filename=sacremoses-0.0.44-cp37-none-any.whl size=886
  Stored in directory: /root/.cache/pip/wheels/3e/fb/c0/13ab4d63d537658f4483667446543
Successfully built ftfy sqlitedict langdetect segtok mpld3 overrides sacremoses
ERROR: torchvision 0.9.1+cu101 has requirement torch==1.8.1, but you'll have torch 1.
ERROR: torchtext 0.9.1 has requirement torch==1.8.1, but you'll have torch 1.7.1 whic
ERROR: google-colab 1.0.0 has requirement requests~=2.23.0, but you'll have requests
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3
Installing collected packages: ftfy, requests, overrides, konoha, sqlitedict, sentenc
  Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
  Found existing installation: gdown 3.6.4
    Uninstalling gdown-3.6.4:
      Successfully uninstalled gdown-3.6.4
  Found existing installation: torch 1.8.1+cu101
    Uninstalling torch-1.8.1+cu101:
      Successfully uninstalled torch-1.8.1+cu101
Successfully installed bpemb-0.3.3 deprecated-1.2.12 flair-0.8.0.post1 ftfy-6.0 gdown
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

2. Load Flair NER model

```
#Load NER Model
tagger = SequenceTagger.load('ner')
```

```
2021-04-13 06:09:23,271 ----------------------------------------------------------
2021-04-13 06:09:23,278 The model key 'ner' now maps to 'https://huggingface.co/fl
2021-04-13 06:09:23,281  - The most current version of the model is automatically
2021-04-13 06:09:23,283  - (you can alternatively manually download the original m
2021-04-13 06:09:23,284 ----------------------------------------------------------
Downloading: 100%                    432M/432M [00:13<00:00, 32.5MB/s]

2021-04-13 06:09:36,316 loading file /root/.flair/models/ner-english/4f4cdab26f24c
```

3. Load Data

```
from google.colab import drive

import functools
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/My Drive/eval_content_2016.csv')

#Above two lines may have to be changed to import your data depending on where it is
#and what the file name is
```

```
Mounted at /content/drive
```

4. Remove pronouns

```
pronouns = ['I', 'You', 'It', 'He', 'She', 'We', 'They']
suffixes = ["", "’m", "’re", "’s", "’ve", "’d", "'m", "'re", "'s", "'ve", "'d", "m", "re", "s

contraptions = [(p, s) for p in pronouns for s in suffixes]

df_contraptions = pd.DataFrame(contraptions, columns=['pronoun', 'suffix'])

df_contraptions['contraption'] = df_contraptions.apply(lambda x: x['pronoun'] + x['suffix'],

contraptions = df_contraptions.contraption.values
```

4. Define NER function

```
# define function

def get_ner_data(df_row):
    '''
    - function to extract named entities from a paragraph
    - returns two data frames:
        - the first is a dataframe of all unique entities (persons and orgs)
        - the second is the links between the entities
    '''
    paragraph=df_row.content
```

```
        #changed above row
        # remove newlines and odd characters
        paragraph = re.sub('\r', '', paragraph)
        paragraph = re.sub('\n', ' ', paragraph)
        paragraph = re.sub("’s", '', paragraph)
        paragraph = re.sub("“", '', paragraph)
        paragraph = re.sub("”", '', paragraph)


        # tokenise sentences
        sentences = tokenize.sent_tokenize(paragraph)
        sentences = [Sentence(sent) for sent in sentences]

        # predict named entities
        for sent in sentences:
            tagger.predict(sent)

        # collect sentence NER's to list of dictionaries
        sent_dicts = [sentence.to_dict(tag_type='ner') for sentence in sentences]

        # collect entities and types
        entities = []
        types = []
        for sent_dict in sent_dicts:
            entities.extend([entity['text'] for entity in sent_dict['entities']])
            types.extend([str(entity['labels'])[1:4] for entity in sent_dict['entities']])
       #The above line is what I changed from the default notebook to get things working

        # create dataframe of entities (nodes)
        df_ner = pd.DataFrame(data={'entity': entities, 'type': types})
        df_ner = df_ner[df_ner['type'].isin(['ORG'])]
        df_ner = df_ner[df_ner['entity'].map(lambda x: isinstance(x, str))]
        df_ner = df_ner[~df_ner['entity'].isin(df_contraptions['contraption'].values)]
        df_ner['entity'] = df_ner['entity'].map(lambda x: x.translate(str.maketrans('', '', strir
        #df_ner['entity'] = df_ner.apply(lambda x: x['entity'].split(' ')[len(x['entity'].split('
        df_ner = df_ner.drop_duplicates().sort_values('entity')

        # get entity combinations
        combs = list(combinations(df_ner['entity'], 2))

        # create dataframe of relationships (edges)
        df_links = pd.DataFrame(data=combs, columns=['from', 'to'])

        #Adding information to links for data tracking and visualization- use one section OR the

        df_links['title']=df_row.title
        df_links['date']=df_row.date
        #Use these two for Kaggle datasets- they do not have URLs since they are old now, so we u

        #df_links['url']=df_row.link
        #df_links['date']=(df_row.datetime)[0:10]
        #df_links['time']=(df_row.datetime)[10:25]
```

```
    #Use these for any of our own datasets pulled using Goose (they have URLs and datetimes)

    return df_ner, df_links
```

5. Apply function

```
df_domain = df.groupby('media').agg({'content': 'count'}).reset_index()
df_domain.columns = ['media', 'count']
df_domain = df_domain.sort_values('count', ascending=False)
dfd_small=df_domain.iloc[1:21,:]

dfd_small

# g2 = sns.barplot(data=dfd_small,
#              x='count',
#              y='Domain',
#              dodge=False,
#              orient='h',
#              hue='count',
#              palette='viridis')

# g2.set_yticks([])
# g2.set_title('Number of articles from each provider')
# g2.set_xlabel('Count')
# g2.set_ylabel('')
# g2.set_xlim(0, max(dfd_small['count'])+150)
# g2.legend_.remove()
# g2.tick_params(labelsize=5)

# for i in dfd_small.index:
#             g2.text(df_domain.iloc[i]['count']+5, i+0.25, df_domain.iloc[i]['Domain'], font

# sns.despine()
# g2.get_figure().savefig('domain_plot.png', dpi=1000)
```

	media	count
108	Fortune	37
287	The Motley Fool	33
304	TheStreet	32
224	Quartz	29
161	MarketWatch	27
218	Profit Confidential	22
277	The Guardian	21
48	Business Wire	20
107	Forbes	19
56	CNBC	18
233	Reuters	16
183	Nasdaq	15

```
df_ner = pd.DataFrame()
df_links = pd.DataFrame()
```

```
for row in tqdm(df.itertuples(index=False)):
#changed above row
  try:
    df_ner_temp, df_links_temp = get_ner_data(row)

    df_ner = df_ner.append(df_ner_temp)
    df_links = df_links.append(df_links_temp)
  except:
    continue
```

```
 1228it [1:08:11,  3.33s/it]
```

```
# praph=df['content'].iloc[1]
# praph = re.sub('\r', '', praph)
# praph = re.sub('\n', ' ', praph)
# praph = re.sub("’s", '', praph)
# praph = re.sub("“", '', praph)
# praph = re.sub("”", '', praph)
```

```
# # tokenise sentences
# sentences = tokenize.sent_tokenize(praph)
```

```
# sentences = [Sentence(sent) for sent in sentences]

# ## predict named entities
# for sent in sentences:
#     tagger.predict(sent)

# # # collect sentence NER's to list of dictionaries
# sent_dicts = [sentence.to_dict(tag_type='ner') for sentence in sentences]



# # # collect entities and types
# entities = []
# types = []
# for sent_dict in sent_dicts:
#     entities.extend([entity['text'] for entity in sent_dict['entities']])
#     types.extend([str(entity['labels'])[1:4] for entity in sent_dict['entities']])

# types
df_links
```

	from	to	title	date
0	AXP	American Express	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
1	AXP	American Express Company	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
2	AXP	Costco	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
3	AXP	Deutsche Bank	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
4	AXP	MarriottStarwood	Analysts Remain Positive on Starbucks Corporat...	Jan 22, 2016
...	...	...	...	...
73	USDA	Walmart	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016
74	USDA	WillardBishop	How Aldi is beating Walmart in the grocery aisle	Mar 29, 2016

6. Remove plurals and possessives

```
def remove_s(entity, entity_series):
  if (entity[-1] == 's') & (entity[:-1] in entity_series):
    return entity[:-1]
```

```
    else:
      return entity
```

```
df_links['to'] = df_links['to'].map(lambda x: remove_s(x, df_ner['entity'].values))
df_links['from'] = df_links['from'].map(lambda x: remove_s(x, df_ner['entity'].values))
df_ner['entity_cl'] = df_ner['entity'].map(lambda x: remove_s(x, df_ner['entity'].values))
```

```
df_links[df_links['to'].str.contains('They')]
```

	from	to	title	date

7. Export Data

```
djia_31_tix = pd.Series(['AXP','AAPL','AMGN','BA','CAT','CRM','CSCO','CVX','DIS','DOW','DJIA'
                         'GS','HD','HON','IBM','INTC','JNJ','JPM','KO','MCD','MMM',
                         'MRK','MSFT','NKE','PG','TRV','UNH','V','VZ','WBA','WMT',])

djia_31_names= pd.Series(['American Express','Apple','Amgen','Boeing', 'Caterpillar',
                          'Salesforce', 'Cisco', 'Chevron', 'Disney', 'Dow Chemical', 'Dow Jc
                          'Goldman Sachs', 'Home Depot', 'Honeywell', 'International Business
                          'Intel', 'Johnson  Johnson', 'JPMorgan', 'Coca-Cola',
                          'McDonalds', '3M', 'Merck', 'Microsoft', 'Nike', 'Proctor  Gamble',
                          'Travelers', 'UnitedHealth', 'Visa', 'Verizon', 'Walgreens', 'Walma

djia_31_comb=djia_31_tix.append(djia_31_names)

df_ner_dj=df_ner[df_ner.entity_cl.isin(djia_31_comb)]
df_links_dj=df_links[df_links['from'].isin(djia_31_comb)]
df_links_dj=df_links_dj[df_links_dj['to'].isin(djia_31_comb)]

conv_dict = {djia_31_names[company]: djia_31_tix[company] for company in range(len(djia_31_na
df_ner_dj=(df_ner_dj.replace({'entity_cl':conv_dict})).drop_duplicates(subset=['entity_cl'])
df_links_dj=df_links_dj.replace({'from':conv_dict})
df_links_dj=(df_links_dj.replace({'to':conv_dict})).drop_duplicates()
df_links_dj=df_links_dj[df_links_dj['from'] != df_links_dj['to']]
df_ner_dj
```

	entity	type	entity_cl
5	AXP	ORG	AXP
15	Apple	ORG	AAPL
34	Disney	ORG	DIS
4	Goldman Sachs	ORG	GS
1	Microsoft	ORG	MSFT
84	KO	ORG	KO
86	NKE	ORG	NKE
37	Visa	ORG	V
38	JPM	ORG	JPM
15	IBM	ORG	IBM
16	WMT	ORG	WMT
4	BA	ORG	BA
65	JNJ	ORG	JNJ
130	MMM	ORG	MMM
25	Cisco	ORG	CSCO
0	Amgen	ORG	AMGN
39	MRK	ORG	MRK
14	CRM	ORG	CRM
18	Verizon	ORG	VZ
73	UNH	ORG	UNH
18	Dow Jones Industrial Average	ORG	DJIA
31	INTC	ORG	INTC
3	Chevron	ORG	CVX
11	Caterpillar	ORG	CAT
7	DOW	ORG	DOW
11	Home Depot	ORG	HD
3	HON	ORG	HON
21	PG	ORG	PG
3	MCD	ORG	MCD

```
df_ner_dj.to_csv('/content/drive/My Drive/df_ner2_eval_2016.csv', index=False)
df_links_dj.to_csv('/content/drive/My Drive/df_links2_eval_2016.csv', index=False)
```