Heterogeneous Edge Prediction Between Users and Stock Mentions on Twitter

Cody Brown
Virginia Tech
800 Washington St. SW
Blacksburg, VA 24061
1-(707) 228-7420
c0dy@vt.edu

ABSTRACT

Link prediction in heterogeneous networks that map Twitter users to stock symbol mentions is a powerful and complex analysis task. The results of such analysis can be used to aid in stock trend analysis by calculating the probability that one stock will be mentioned in conjunction with another on Twitter. This data can allow an analyst to more accurately determine which stocks are often bought and sold together, resulting in a better view of ongoing buy/sell trends in the market. To accomplish this, users must be mapped with their stock mentions in a one-to-one fashion and paths mapping stock symbol to user to stock symbol must be established using random path traversals. In this paper, we use a series of path-generation, regression, classification, and clustering algorithms to associate stocks with other stocks and predict user's interests in a given stock. We seek to solve two problems: 1. We would like to see how effective a given stock can be in predicting the mention of another stock, and 2. We would like to predict specific stock mentions based on mentions of other stocks. Our experiments on stock-mention correlation and heterogeneous link prediction demonstrate the effectiveness of our approach.

CCS Concepts

• CCS → Computing methodologies → Machine learning → Machine learning approaches → Logical and relational learning → Statistical relational learning

Keywords

Machine Learning; Twitter; Market Research; Stock Trends;

INTRODUCTION

Network link prediction using machine learning has become a valuable tool in many large analysis arenas throughout the world. This concept has been applied to many large datasets and for many purposes such as friend recommendation on Facebook and product recommendation on Amazon. Most of the well-researched algorithms used for this analysis only support homogeneous networks, or networks that only contain one "kind" of node and edge. An example of this would be a connection between two users on a social media website. A far less well-established area of study is that of link prediction between unlike nodes. One example of where this analysis can be applied is predicting a stock symbol that a user might be interested in based on their previous stock symbol mentions. This is a complex operation that requires a heterogeneous graph which maps users to their stock mentions. This paper will focus on this issue.

Both correlating stock mentions and predicting a user's stock preference are valuable problems to solve. Correlating stock mentions means that an analyst has a better perspective on macro trends in the market. For example, if we know that five stocks are usually mentioned together, then we might be able to determine the outcome of one stock based on the outcome of others. Predicting a user's stock preference or mentions can not only help fill holes in the stock correlation data, but it can act as a standard recommender for users on stock trading websites.

To tackle these problems, we used a structured, pipelined approach. The first step is to collect and preprocess the data with the proper feature set. In this case, we collected a mass of tweets using stock symbols as search terms, then save the username, tweet text, and symbols from each tweet. Next, each user was mapped to all of their stock symbol mentions in a one-to-one format. This formed the basis of our heterogeneous graph.

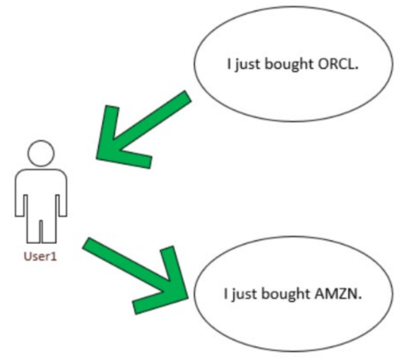


Fig. 1. Using a user as a "hop" between stocks

The second step involved path mapping between each stock symbol using users as a "hop". For example, if "user1" mentions "MSFT" and "user1" also mentions "AAPL", then we can map "MSFT" to "AAPL" using "user1" as a hop. This would result in a possible path between these two stocks (Figure 1). By counting the number of these paths, we can increase the probability that they are related to each other and might be mentioned together. To perform this path mapping operation, we adapted the Hetio library [1] to work with our dataset. The Hetio library easily found all paths from "symbol to user to symbol" in the dataset using random walks. We calculated all paths for all symbols to all other symbols in our dataset. Using this data, it is very easy to calculate the relative probability of two stocks being related within the dataset. This solves our first problem.

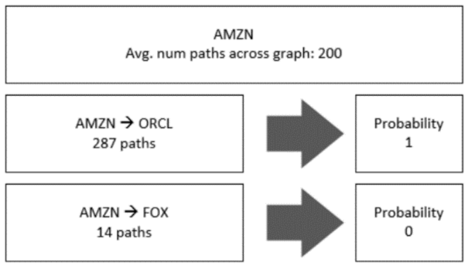


Fig. 2. Probability mapping based on path count

The third step was to process the mappings between stock symbols to work on standard machine learning models for link prediction. To do this, we calculated the average number of paths between each symbol within the graph, then created a probability threshold at that average. Using this threshold, if the number of paths between two given stock symbols was higher than the average for the source, we would use this as a positive indicator (1). If the number of paths between the two stocks was lower than the average for the source, we would use this as a negative indicator (0) (Figure 2). This helps us weight our graph if we would like to use stock mentions as features and also helps solve our first problem.

The fourth step involved using our original user to tweet heterogeneous graph and mapping it on to our probability graph. This was accomplished by taking all user-to-stock mappings and creating a matrix with an array for each user. For each stock symbol mentioned by the user, we mapped a positive indicator (1) and for each symbol not mentioned by the user, we mapped a negative indicator (0). This resulted in a matrix that can be used for link prediction using classifiers and/or clustering.

Finally, we used a series of classifiers and clustering models on the user-to-stock mention matrix. This let us determine how well mentions of stocks could be used to predict mentions of other stocks, solving our second problem.

BACKGROUND/RELATED WORK

Analysis of heterogeneous graphs is a fairly unstudied area. For this project, we found that the Hetio project [1] was performing similar analysis to what we wanted to perform. However, they were analyzing biomedical data, so the problem set was different. Fortunately, they provided an extensible set of tools for path calculation, and we were able to adapt our dataset to work with their library.

APPROACH

To start, we downloaded a list of the S&P top 500 stock symbols

from Sure Dividend [2] and removed several from the list that would have caused issues in processing. This resulting in a list of 442 popular stock symbols. We used this list as search terms for downloading top tweets using Twitter's search API. We collected more than 600,000 tweets and stored them as plain-text representations of the Tweepy python library's Status object. This representation maintained all data and metadata from each tweet for later processing.

Next, we normalized the data into a standard JSON model that maintained the tweet text, the tweet's associated symbols, the username of the author, and the tweet's creation time. This allowed us to be flexible when deciding how to use the data. Upon deduplication of tweet text, we had approximately 300,000 data points. Our data model is as follows:

{ "ScreenName", "TweetText", "Symbols", "Date" }

We then created our heterogeneous network that mapped users to tweets. First, we created a list of unique users from our dataset. For each user, we checked to see if they had mentioned any of our top 442 stock symbols. If they had, we would add an entry to our graph using the following data model:

{ "ScreenName", "Symbol" }

This resulted in a heterogenous network containing 42,512 user nodes, 442 stock symbol nodes, and 658,428 edges.

Once our first preprocessing operation was complete, we downloaded and installed the Hetio Python library [1] to perform path calculations. This library is used to perform heterogeneous network path calculations on biomedical datasets. For our purposes, we needed to make some minor modifications to the library and then switch our data model to support the Hetio framework. Further details of this data model can be found within Hetio documentation.

Using the Hetio library, we calculated the number of paths between each symbol and all other 441 stock symbols. This approach uses a user as a "hop" between the stock symbols, as seen in Figure 2 above. When applied on a large scale, this resulted in a large number of paths being found between each stock in the graph (Figure 3).

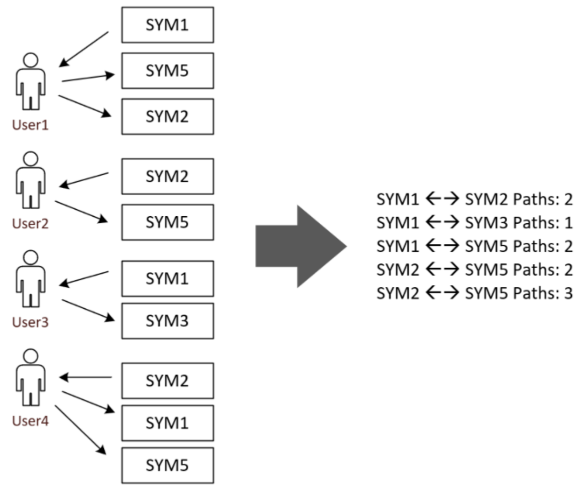


Fig. 3. Large number of paths between stock symbols

We mapped this data using the following data model for further processing:

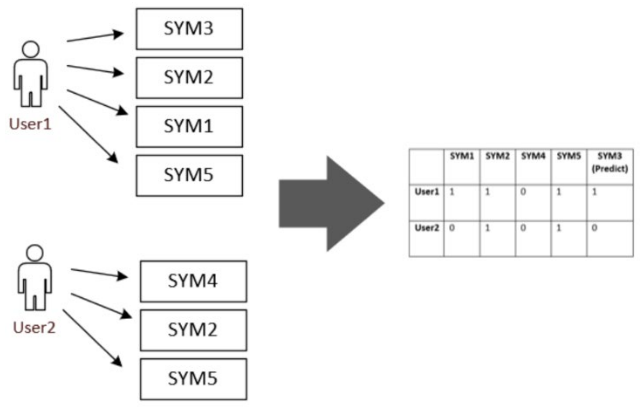
{ "source", "target", "numPaths" }

The number of paths between two given stock symbols helps indicate how good one stock mention is at predicting the other. The Hetio library also allowed us to perform degree calculations on each path, which effectively calculates the prevalence of a specific type of path between two nodes [1]. Using this data, we easily found the best predictors of each stock using a simple python script. We stored this information in the following data model:

{ "symbol", "bestPredictor", "numPaths" }

These predictors can potentially be aggregated in the future and analyzed for use in stock market trend analysis. A larger sample set should be used to assess the efficacy of these as valid predictors.

To aid in weighting a secondary graph if necessary, we also created a dataset that rated the efficacy of a path by whether it was higher or lower than the average number of paths for the source node overall. This method is illustrated in Figure 2. If the number of paths between two nodes was higher than the average for the source node in the graph, we gave it a positive indicator (1), otherwise it would get a negative indicator (0). We did not end up using this dataset as planned, but it could be useful in running the k-nearest neighbors model on the data in the future. This dataset could also be used to create a homogenous network between nodes that have a positive indicator association.



	SYM1	SYM2	SYM4	SYM5	SYM3 (Predict)
User1	1	1	0	1	1
User2	0	1	0	1	0

Fig. 4. Mapping binary features for classification and clustering

Next, we needed to prepare our data for processing by clustering models and classifiers. These are necessary for true link prediction on a heterogeneous graph like this one, due to the fact that traditional link-prediction models only work on homogeneous graphs. To support this, we needed to make a matrix of binary features, with the stock to predict being the result. We accomplished this by creating a matrix with an array/row per-user. For each stock symbol, we added an entry to each row. If the user had mentioned the stock symbol, we added a positive indicator (1) in that spot. Otherwise, we added a 0 in that spot. If the user had mentioned the stock we were looking to predict, we added a positive indicator to the result column (Figure 4).

Finally, we created predictor datasets as described in Figure 4 for 7 stock symbols for testing: AAPL, AMZN, CSCO, IBM, NFLX, ORCL, and PYPL. For each of these datasets, we used linear regression, decision tree classifier, Naive Bayes classifier, and K-Means Clustering models to determine accuracy. We then recorded and analyzed the results.

EXPERIMENT

For the first portion of our experimentation, we used our Hetio-preprocessed heterogeneous network to calculate the number of paths between each stock symbol. We took our stock symbols list and created a static mapping between every unique pair possible in the set. Using this unique set of tuples, we used Hetio's path_between function to calculate the path between a given symbol and every other symbol in the set. From there, we extracted the result with the highest number of paths and stored this as the best predictor for the symbol. Some of our top predictors were as follows:

Table 1. Top predictors based on path number

Source Symbol	Best Predictor	Paths Between
AMZN	AAPL	1275
AMD	AAPL	847
QCOM	AAPL	675
TWTR	AMZN	515
ATVI	AAPL	402
MMM	AMZN	389
NVDA	AAPL	297
COST	ESS	259
ORCL	NKE	223
JPM	AAPL	161

This data can be aggregated and used in mass to help analyze macro market trends or add weight to graphs created from this dataset to aid in processing. Figure 5 shows the degree distribution of the paths in this heterogeneous network.

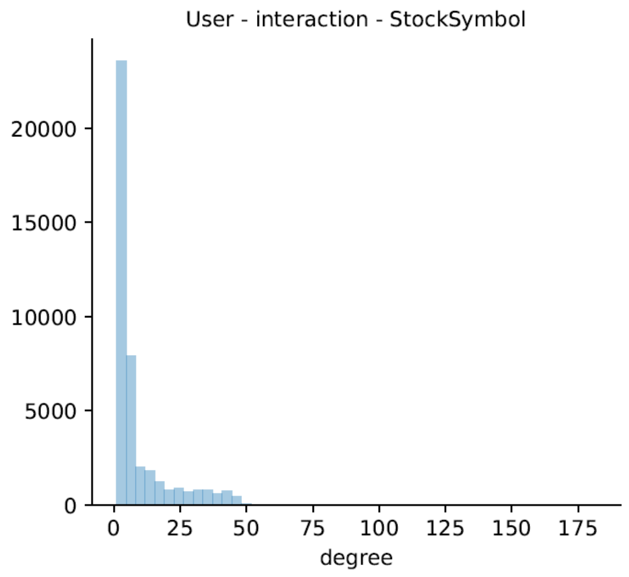


Fig. 5. Path degree distribution across the heterogeneous network

Next, we created a binary graph that uses user stock mentions as features as illustrated in Figure 4. We generated a graph for 7 stock symbols from our list (randomly selected): AAPL, AMZN, CSCO, IBM, NFLX, ORCL, and PYPL. For each of these datasets, we used linear regression, decision tree classifier, Naive Bayes classifier, and K-Means Clustering models to determine accuracy. Our results were as follows:

1.1 AAPL

Table 2. Linear Regression on AAPL dataset

Residual Sum of Squares	Variance Score
0.02	0.98

We can see here that using linear regression was a very accurate means of predicting AAPL mentions with this dataset. This is a trend we will see across the board as the features appear to be good predictors.

Table 3. Decision Tree Classifier on AAPL Dataset

	Precision	Recall	F1
Mean	0.94426	0.93405	0.93910
Std Dev	0.00418	0.00875	0.004256

The decision tree classifier was the most precise method of prediction tested on this dataset. This is likely because the features were binary, and the data contains all possible outcomes which represents a perfect environment for a decision tree.

Table 4. Naïve Bayes Classifier on AAPL Dataset

	Precision	Recall	F1
Mean	0.94079	0.93926	0.94000
Std Dev	0.00789	0.00723	0.005678

Again, for this dataset, classifiers appeared to be very accurate. In this case, roughly 6% of instances were improperly predicted using Naïve Bayes.

Table 5. K-Means Clustering on AAPL dataset

	Class 0	Class 1
Clusters	6482	36030
Ground Truth	35166	7346

Here, we can see that the K-Means clustering model did not perform so well. This likely means that the dataset is not very divisive in nature.

1.2 AMZN

Table 6. Linear Regression on AMZN dataset

Residual Sum of Squares	Variance Score
0.05	0.95

For this dataset, Linear Regression was by far the most accurate way of predicting AMZN mentions. It seems that this dataset was most suited to linear approximation.

Table 7. Decision Tree Classifier on AMZN Dataset

	Precision	Recall	F1
Mean	0.52508	0.51164	0.51812
Std Dev	0.00650	0.014813	0.007416

When compared to the AAPL dataset, the decision tree classifier performed very poorly on the AMZN dataset. Roughly 50% of predictions were inaccurate.

Table 8. Naïve Bayes Classifier on AMZN Dataset

	Precision	Recall	F1
Mean	0.23359	0.47309	0.31252
Std Dev	0.018235	0.02681	0.02095

For this dataset, classifiers appeared to perform very poorly. In this case, roughly 60% of instances were improperly predicted using Naïve Bayes.

Table 9. K-Means Clustering on AMZN dataset

	Class 0	Class 1
Clusters	7017	35495
Ground Truth	39725	2787

The K-Means clustering model performed about as poorly on this dataset as with the other. This likely means that the dataset's features are not very divisive in nature.

1.3 CSCO

Table 10. Linear Regression on CSCO dataset

Residual Sum of Squares	Variance Score
0.01	0.99

The CSCO dataset was nearly perfect when using regression. This is likely because it was smaller than some of the others by comparison and had better features by chance.

Table 11. Decision Tree Classifier on CSCO Dataset

	Precision	Recall	F1
Mean	0.100417	0.08216	0.09023
Std Dev	0.029788	0.021804	0.02500

The decision tree classifier, like the others in the CSCO dataset, performed very poorly. This is likely because of the smaller number of positive results compared to the others.

Table 12. Naïve Bayes Classifier on CSCO Dataset

	Precision	Recall	F1
Mean	0.070069	0.79550	0.12871
Std Dev	0.005791	0.03151	0.00987

For this dataset, classifiers appeared to perform terribly. In this case, roughly 90% of instances were improperly predicted using

Naïve Bayes. Again, this is likely due to the number of positive results.

Table 13. K-Means Clustering on CSCO dataset

	Class 0	Class 1
Clusters	35496	7016
Ground Truth	42022	490

Interestingly, the smaller CSCO dataset actually performed the best when it came to K-Means clustering. Unfortunately, this is likely due to chance rather than the data.

1.4 IBM

Table 14. Linear Regression on IBM dataset

Residual Sum of Squares	Variance Score
0.07	0.93

The IBM dataset performed pretty well with Linear Regression. It seems that this data is well-suited to Linear Regression overall.

Table 15. Decision Tree Classifier on IBM Dataset

	Precision	Recall	F1
Mean	0.36032	0.34802	0.35398
Std Dev	0.02022	0.01398	0.01632

The decision tree classifier performed poorly on the IBM dataset. However, it performed better than the CSCO dataset. It seems that the larger the number of positives, the better the results.

Table 16. Naïve Bayes Classifier on IBM Dataset

	Precision	Recall	F1
Mean	0.11327	0.07757	0.09143
Std Dev	0.016820	0.00969	0.01062

For this dataset, the Naïve Bayes classifier performed terribly. In this case, roughly 90% of instances were incorrectly predicted. An interesting note is that this is the first significant divergence we have seen between decision tree and Naïve Bayes. This demonstrates the value of using multiple models.

Table 17. K-Means Clustering on IBM dataset

	Class 0	Class 1
Clusters	7016	35496
Ground Truth	39611	2901

As we now expect, the K-Means clustering operation performed very poorly on this data.

1.5 NFLX

Table 18. Linear Regression on NFLX dataset

Residual Sum of Squares	Variance Score
0.04	0.96

As expected, the Linear Regression model performs well on the data again.

Table 19. Decision Tree Classifier on NFLX Dataset

	Precision	Recall	F1
Mean	0.35169	0.31555	0.33223
Std Dev	0.01407	0.02429	0.01741

The decision tree classifier performed poorly on the NFLX dataset, but about the same as the IBM dataset. This is likely because both datasets had a similar number of positive results.

Table 20. Naïve Bayes Classifier on NFLX Dataset

	Precision	Recall	F1
Mean	0.12268	0.44332	0.19214
Std Dev	0.00764	0.03770	0.01276

The Naïve Bayes classifier did not work out again. The recall score was in the 40s, but this is likely due to ordering of the data, not the actual feature set.

Table 21. K-Means Clustering on NFLX dataset

	Class 0	Class 1
Clusters	36069	6443
Ground Truth	40633	1879

In this case, the K-Means clustering algorithm actually performed better than expected. This could be chance, but it could also have to due with the number of positives and the feature set being more divisive.

1.6 ORCL

Table 22. Linear Regression on ORCL dataset

Residual Sum of Squares	Variance Score
0.02	0.98

As expected, the Linear Regression model performs well on the data again. This time with a near-perfect score.

Table 23. Decision Tree Classifier on ORCL Dataset

	Precision	Recall	F1
Mean	0.29897	0.29423	0.29574
Std Dev	0.02415	0.02538	0.01854

The decision tree classifier performed poorly on the ORCL again, but this time there does not seem to be a relation between the number of positive results. In this case, there are only 638 positive results which is similar to the CSCO dataset. I would have expected worse results here.

Table 24. Naïve Bayes Classifier on ORCL Dataset

	Precision	Recall	F1
Mean	0.07774	0.70291	0.13989
Std Dev	0.00764	0.03770	0.01276

The Naïve Bayes classifier did poorly when compared to the Decision Tree. It seems that this dataset is more suited to the decision tree process.

Table 25. K-Means Clustering on ORCL dataset

	Class 0	Class 1
Clusters	35494	7018
Ground Truth	41874	638

The K-Means clustering algorithm performed fairly well. However, K-Means always seems to split the data the roughly the same way. It is hard to determine at this point if this is by chance.

1.7 PYPL

Table 26. Linear Regression on PYPL dataset

Residual Sum of Squares	Variance Score
0.01	0.99

As expected, the Linear Regression model performs well on the data. Again, with a near perfect score.

Table 27. Decision Tree Classifier on PYPL Dataset

	Precision	Recall	F1
Mean	0.14396	0.12390	0.13288
Std Dev	0.05064	0.03571	0.04235

The decision tree classifier performed very poorly on the PYPL dataset. In this case, there are 524 positive results which is like the CSCO dataset. As expected, we get similar results to those from the CSCO dataset.

Table 28. Naïve Bayes Classifier on PYPL Dataset

	Precision	Recall	F1
Mean	0.07095	0.76188	0.12977
Std Dev	0.00564	0.04384	0.00985

The Naïve Bayes classifier performed poorly on the PYPL dataset. This is likely due to the small number of positive results.

Table 29. K-Means Clustering on PYPL dataset

	Class 0	Class 1
Clusters	35496	7016
Ground Truth	41988	524

The K-Means clustering algorithm performed fairly well. However, K-Means always seems to split the data the roughly the same way. It is hard to determine at this point if this is by chance.

CONCLUSION

To conclude, there are a few things we can draw from this experiment. To start, we can see that it is possible to traverse a heterogeneous network of users and things they mention. It is also possible to map this to a homogenous network by finding the number of paths between each set of nodes then creating a mapping between nodes that have a high enough number of paths. We have also found that it is possible to determine the nodes that have the most influence/interaction over other nodes in a heterogeneous network by determining the paths between them.

We can also note that it is possible to use the path heterogeneous mappings in a network as features for clustering and classification. Here, we saw that for predicting stock mentions in binary datasets with larger numbers of positive results, classification was fairly accurate. In many cases, we had above 30% accuracy, even with lower numbers of positive results in the dataset. In every case, linear regression on the dataset was extremely accurate as well. However, clustering appeared to have the same results every time and was very poor.

In the future, we can likely apply these techniques to a much larger dataset for better results. For predicting larger stock symbols such as AMZN and AAPL, these experiments performed well and speak to the efficacy of our approach.

Finally, we were able to solve our original two problems using this approach to data processing:

1. We would like to see how effective a given stock can be in predicting the mention of another stock.

2. We would like to predict specific stock mentions based on mentions of other stocks.

ACKNOWLEDGMENTS

Our thanks to Sure Dividend at https://www.suredividend.com for providing us with a list of the top 500 S&P stock symbols.

REFERENCES

[1] Himmelstein D. S. and Baranzini S. E. 2015. Heterogeneous Network Edge Prediction: A Data Integration Approach to Prioritize Disease-Associated Genes. *PLOS Computational Biology*. DOI= https://doi.org/10.1371/journal.pcbi.1004259.

[2] Sure Dividend Top 500 S&P Stocks. *https://www.suredividend.com/sp-500-stocks/.*