Predicting Investments in Startups using Network Features and Supervised Random Walks

Arushi Raghuvanshi
arushi@stanford.edu

Tara Balakrishnan
taragb@stanford.edu

Maya Balakrishnan
mayanb@stanford.edu

***Abstract*—Using data from Mattermark and Crunchbase, we represented investor-company relationships in a bipartite graph with edges representing an investor investing in a company. This feature-rich graph includes information on the companies such as industry, location and business model, and each edge is labeled with the investment amount and funding round. Using this network, we developed a link prediction model to predict investments, given company histories and investors' prior portfolios.**

***Index Terms*—social networks, investors, companies, Silicon Valley, link prediciton, machine learning, supervised random walks**

I. Introduction

Data analysis of the start-up ecosystem has been a trending topic in the past few years, because it is incredibly valuable to both investors and companies. In particular, predicting investments is valuable to companies, because it informs them about which investors to approach when they are looking to raise a round. And predicting investments is valuable to investors, because it allows them to identify which start-ups will have successful valuations and be good targets to invest in. However, very little of this analysis has been done through network analytics. There is rich information embedded in the graph of investment relationships. For example, let's say that two investors A and B both invested in company X. If investor A also invested in company Y, then investor B may be more likely to invest in company Y as well. An investment network contains this information as well as many other relationships between investors in companies that cannot be modeled by company and investor features alone. Therefore, we thought it would be interesting to approach investment prediction by understanding the relationships between investors and companies in a network.

II. Prior Work

A. Investment networks

In [1], Yuxian and Sor-Tsy predict whether an investor will invest in a company given that someone at that investment organization has previously worked at that company (called a 'social relationship'). The authors considered an undirected graph of companies and investors, where edges exist between an investor and company if they have a social relationship or an investment has been made, and used a supervised learning approach to predict links. They used network features commonly used in link prediction including shortest path, Adamic/Adar similarity, and Jaccard coefficient. The authors demonstrated success at predicting investments by achieving an AUC score of 0.84 using SVMs. One limitation of this paper, is that all edges are the same weight regardless of the connection. For example if someone worked briefly at a company the edge is probably less significant than if they were the founder of a company. Similarly, if they invested a small amount in a company the edge may be different than if they invested a large amount in a company. We hope to improve on the edge model described in this paper, by adding edge weights to the network based on the amount invested. We will also expand on the investment prediction objective by predicting not only whether an investment will occur, but also at what round and how much money will be raised in that round.

B. Link prediction

Different link prediction features are better suited for different networks. The network that we will analyze is a bipartite graph and relatively sparse. One promising approach to link prediction in bipartite graphs is using a weighted projection of the graph and what are called internal links [5]. This model scores edges based on the intuition that two nodes with a common neighbor are more likely to acquire more neighbors in the future. The authors used this model for link prediction in recommendation systems and achieve a precision of about 50% and recall of about 20% which greatly out preforms collaborative filtering on this dataset.

C. Supervised random walks

In their paper "Supervised Random Walks: Predicting and Recommending Links in Social Networks," Backstrom and Leskovec address the problem of predicting links in a graph with rich node and edge data [3]. Unsupervised random walks and PageRank methods take the structure of the graph into account, but not the relevant information of the node attributes. Supervised machine learning algorithms take node attributes into account, but have to use feature extraction techniques to extract node features based on domain knowledge and trial-and-error that do not take the full network into account.Testing against both synthetic and real world data, they found that compared to unsupervised methods, SRW gave a boost of 25 percent. It also performed better than supervised machine learning algorithms.

III. Data Collection and Network Model

A. Datasets

Our network is a directed bipartite graph from investor nodes to company nodes where an edge (A, B) represents an investment by investor A in company B. We used two sources to compile our data: Crunchbase and Mattermark.

Mattermark has rich feature information on companies including their business model(s) (B2B, B2C, etc.), industry (IoT, Advertising, etc.), and keywords. Their data is incredibly useful for feature vectors, because the values stored in each category (industry, business model, etc.) are standardized across all companies, and stored in lists instead of text blobs. This allows us to use complex features such as industry and business model without sophisticated natural language processing algorithms. The data also includes company information such as current stage, current number of employees, social media metrics, and compiled growth scores. The limitation of Mattermark is that all information is based on the current stage of the company, so it only includes the total amount of funding raised to date. In order to predict investments, we want to model the amount invested at each round, so that we can predict what new links will be made from time t_i to time t_i'.

The Crunchbase dataset provided us with this time based information. The Crunchbase dataset included the series (A, B, C, etc.) during which a portfolio company received money from one of their investors and the total amount of money raised during that round.

B. Collection

Investor-Portfolio Company networks are naturally sparse, because there are many investors and companies and comparatively few investments. To minimize sparseness, we downloaded all portfolio companies of the 20 investors with the largest portfolios at each stage of investment (Pre series-A, series A, series B, series C, and late). We were able to download Excel spreadsheets from Mattermark for each investor's portfolio listing all the companies they had invested in along with statistics on each company. We then parsed these Excel files and created a dictionary that contained all the unique companies and investors as keys and mapped each one to the relevant features that were recorded.

We collected data from Crunchbase by leveraging the Crunchbase API. We wrote a sequence of scripts to get all pages of the investment data for each of our selected investor nodes, parse the json files, and print the features to a csv file.

Combining and cleaning the Mattermark and Crunchbase datasets into feature vectors and their corresponding labels required a significant amount of processing. While there was significant overlap in nodes between the two datasets (about 80%), we had to join the datasets based on company name, which often had variations in capitalization and special characters. Additionally, the Mattermark dataset included seemingly arbitrary line breaks in the middle of the data. Since line breaks were used as delimiters in Mattermark, this required manual cleaning.

C. Network description

Each of the 100 investor portfolios that we downloaded contained about 100-800 portfolio companies. These investors and companies then formed the nodes of our graph. There was a significant amount of overlap resulting in about 55 investor nodes and 8,000 company nodes.

We parsed through both datasets and created edges from investors to companies resulting in almost 13,500 edges in our network. For each edge we also stored edge weight information, which is the round(s) during which funding was received and the total amount received in those rounds. Because a investor-company node combination could have multiple edges (ie. the investor chose to invest in the company during multiple rounds), we choose to create a Multi-Graph (TNEANet in Snap.py) to represent our network. This way, each edge would be labeled with a unique id, allowing for multiple edges from the same investor-node combination.

IV. Feature and Network Statistics

A. Company Feature Statistics

In order to understand how the features of a company impact the amount of money the company receives per round, we created some summary graphs with our main features. A selection can be found here, while more graphs are in the Appendix. Each graph shows the average amount a company raised per round, for series A, series B, and series C - given some descriptor that is applied to the company. The descriptors that we looked at were: business models, investors who also invested in the company, location, keywords, and industry.

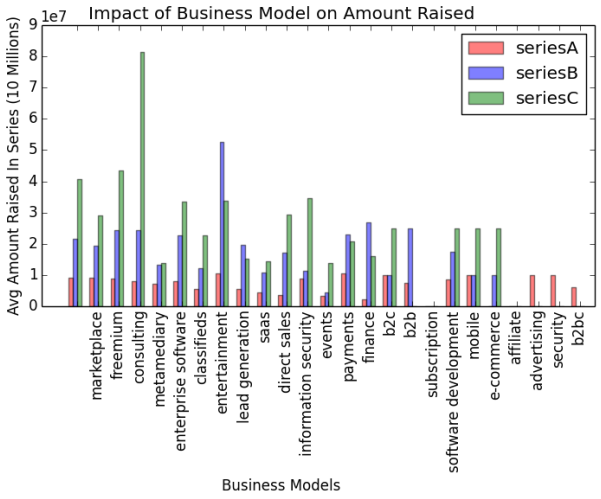


Fig. 1.

In Fig. 1 we looked at the different business models that companies fall under. While it appears that companies who use certain business models receive no investments, based on Fig. 1, those models are only used to describe companies who raised late stage funding (series C-G), which is not pictured here.

A general trend across features, is the increase in amount raised per round as later rounds are reached. This corresponds to something we initially assumed about investors; investors are inherently risk averse and won't invest as much money in an unproven company as in a proven one. This indicates that the series during which a company raises money in, is a key feature in predicting the amount an investor will invest in a company.

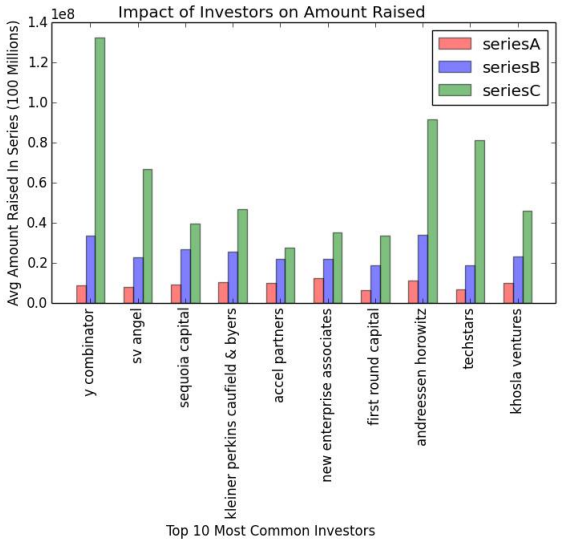


Fig. 2.

Fig. 2 gives us a general investing profile for each of the top 10 investors. For example, we can see that while Y Combinator has on average lower investments in series A stage companies than Kleiner Perkins Caufield & Byers, their average investment increases exponentially along with the series. It is possible that as we reach later stages, Y Combinator and Andreesen Horowitz, for example, have an algorithm for more accurately discovering companies that make a lot of money, so in turn due to lower risk, they are able to invest a lot more money in them. Fig. 2 also indicates how much money a company is likely to raise if a given investor has invested in them at any gives point in time. Since there is a great deal of variation in amounts that different investors choose to invest in companies, it is clear that these will also be crucial features to look at when predicting investment amounts. While Fig. 2 could be an important investor feature, we decided to include it in the company node feature vector. Based on this analysis we combined all of these potential features into a single very large, very sparse, feature vector based on the attributes of each company. Our company node feature vector now contains a series of binary features for each option of business model, investor, location, keyword, and industry. Although most companies ascribe to multiple business models, keywords, locations, etc. given the total number of possibilities, the number of features which appear for any given company are comparatively few. Out of approximately 6,740 possible company node features less than 100 will be applied to each company.

B. Network statistics

We input our network into snap and gephi as a sanity check, and the graph appeared reasonable. We also plotted the degree distributions on both a standard scale and log log plot (see Fig. 3). As expected, there were many nodes with low degrees (company nodes), and few nodes with high degrees (investor nodes).

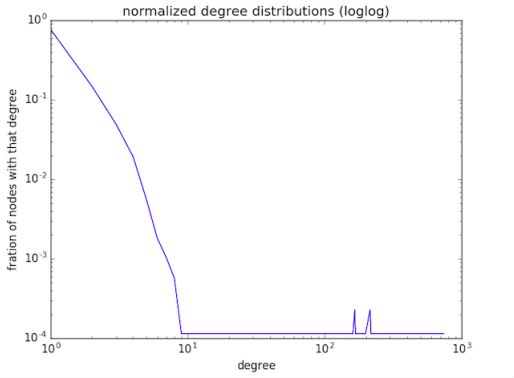


Fig. 3.

V. Algorithms and Methods

We can approach our objective of predicting edges and their respective weights (the amount invested) between investors and companies using two different methods

1) Supervised Learning Algorithms: use a supervised learning algorithm on extracted network features and node features to predict whether a link will be formed and how much money that link represents

2) Supervised Random Walks: predict links with supervised random walks, and predict edge weights with a multi-class supervised learning algorithm.

These methods and associated algorithms are described in more detail below.

A. Extracting network features

For the supervised learning approach, the first step is to extract network features. The network features used are scores commonly used in link prediction such as preferential attachment. Note we were limited in our ability to use many commonly used network extraction features, because they do not generalize well to bipartite graphs. One sophisticated network feature we used was an induced link score described here.

A bipartite graph can be transformed into a classical graph called a projection. The nodes in the projection are the nodes of one side of the bipartite graph. The edges in the graph of the projection are drawn between all nodes (u, v) where u and v share at least one neighbor in the original bipartite graph.

Now considering the projection of a bipartite graph G to be G', a link (u, v) is an internal link if adding it to G does not change G'. In other words, a link is internal if the projection of G and the projection of $G \cup (u, v)$ are the same.

Finally, induced links are the links that are added to the projection of G when some given link is added to the network G. The internal link cost function is the number of induced links that are not internal links when some edge e is added to the graph. We expect this feature to preform well, because if two investors have invested in the same company in the past, it is more likely that they will invest in the same company in the future, and this cost function models this behavior.

The preferential attachment scoring function gives the highest scores to edges where both nodes have high degree. Noting that our network has almost the opposite structure, where investors with high degree tend to have edges with companies with low degree and investors with low degree tend to have edges with companies with high degree, we also included an inverse preferential attachment score. This score is given by $\frac{1}{|\Gamma(a)||\Gamma(a)|}$ where $|\Gamma(x)|$ is the neighbors of node x.

Our final four extracted edge network features were:

1) Preferential attachment score
2) Internal link score
3) Difference in degree of the two nodes
4) Inverse preferential attachment score

After extracting these network features for each edge, we used them as well as node features in supervised learning algorithms.

B. Supervised learning with network features

We implemented various supervised learning algorithms with network features to predict the amount invested. The input to our supervised learning algorithms are feature vectors which represent edges in the network and corresponding labels which represent the weight of that edge. Given a list of edges, the algorithm then predicts the weight of that link (or 0 if that link is not created).

The feature vectors for each edge are a combination of extracted network features and source and destination node features, combined into one large, sparse feature vector. Because we are predicting edge weights, we need to combine the features from the two connecting nodes (one investor and one company) into a single vector. The feature vector also includes network characteristics which are extracted using the link prediction score functions described above. While the resulting vector will be extremely large, we use principal component analysis (PCA), and other dimensionality reduction techniques to reduce the size of the feature vector.

We used multi-classification/bucketed regression algorithms to predict links. Multi-classification was chosen over continuous regression methods, because predicting the range of investment amount is more valuable than predicting the exact dollar amount of an investment. Buckets were created by looking at a histogram of investment amounts in our training data (Fig 4). The bucket ranges were demarcated so each bucket would have roughly the same number of investments of that size fall into it. As a note, Fig. 4 only shows a partial histogram up until \$140 million invested; the full histogram of money invested has an incredible long tail, which extends up to 1.6 billion dollars invested. As a result, the lower buckets encompassed a much smaller range of money than the later buckets.

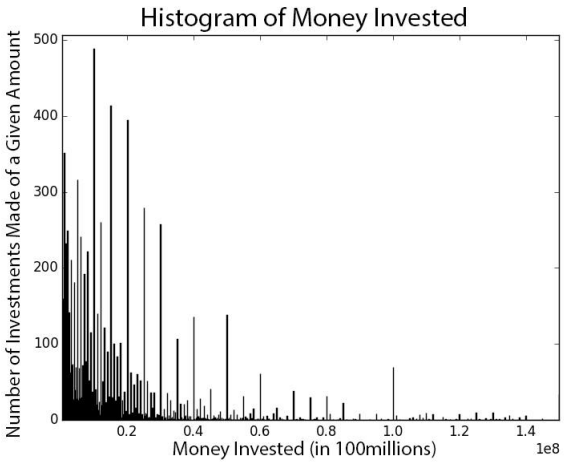


Fig. 4.

Therefore, the y input to our supervised learning algorithms is a multi-class label between 0 and 14. If the multi-classification algorithm predicts the class for 0, then no link is predicted between that source and destination node, while if it predicts the class 3, then an investment in the range of \$300,000 through \$1,100,000, is predicted.

Our problem is one of multi-classification, and our feature vectors are high-dimensional and sparse, with mostly binary features. Thus, we selected the following three supervised learning techniques to attempt classification: L2-regularized Support Vector Machine (SVM), Logistic Regression, and Bernoulli Naive Bayes. All models were implemented using sci-kit.

1) SVM is a standard choice for classification problems, because it is known to be one of the best models for supervised learning. While L1 penalization is typically regarded as better in the case of a sparse dataset, we chose to penalize the SVM with the L2-norm because not all of our features were binary and it appears better at minimizing prediction error based on its better baseline results. Additionally, because our data may not have been linearly-separable, we started with a Radial Bias Function (RBF) which maps features to a non-linear higher dimensional space, unlike the Linear kernel. This left our Logistic Regression model to get a goood baseline for data that is fairly linearly separable.

2) Logistic Regression is the most common discriminative algorithm choice for supervised learning classification because of its ease to implement and reasonable accuracy. Logistic regression allows us to take a non-linear problem and classify it in a linear form. For logistic regression, we experimented with

varying values of C (cost) for each L1 and L2 penalization. We used the sci-kit baseline of L1 regularization and C = 1.0 in our starting model.

3) The Bernoulli Naive Bayes model converts non-binary features to 1. Additionally, it is a generative algorithm and utilizes the probability that a binary feature will be 0 or 1 to conduct classification. Since 6,740 of our 6,745 features are binary, we chose to implement this model in the event that the binary features have a significantly greater impact on the classification results than the remaining non-binary features.

After implementing and tuning each model, to reduce the risk of overfitting, we applied dimensionality reduction to each model. We chose to use Principal Component Analysis (PCA), which uses an orthogonal transformation to convert possibly correlated variables into linearly uncorrelated principal components, which occupy a lower dimensional space.

C. Supervised random walks

The second method we tried is using supervised random walks (SRW). This algorithm utilizes all of the features used in the supervised learning algorithms as well as the full structure of the network to predict edges. The first step of the algorithm is to learn the probabilities of taking each edge using node features and an optimization problem. The next step is to then preform personalized page rank with these learned weights.

The given optimization problem for supervised random walks is described as follows:

$$min_w F(w) = ||w||^2 + \lambda * \sum_{d \in D, l \in L} h(p_l - p_d)$$

where L is the set of nodes not in D, and λ is a regularization parameter. In this equation, w refers to parameters which assign weights to each edge for the random walk. The goal is to optimize w so that the page rank scores of nodes that are neighbors of starting node s have a higher page rank score than the nodes that aren't neighbors of node s. The optimization function also uses h, which is a loss function. This allows us to treat the problem as if there is a soft constraint so that it can fit the data better without overfitting.

Since we are using a directed bipartite graph, we must modify the SRW algorithm, because a random walk would always terminate at the first company node it traveled to. We modify the optimization problem to create temporary links between company nodes that are weighted by the similarity of their feature vectors to use in the random walk. We modify the optimization problem as follows. We create an adjacency matrix w which holds the personalized page rank weights. We train these weights by calculating the Jaccard similarity between the feature vectors for each pair of company nodes and adding that to the matrix. An edge from an investor to a company node is added with a weight proportional to the number of links between that investor and that company. Note that since our graph is a multigraph, it is possible for one investor to invest in a company more than once. We then normalize the matrix to achieve our final personalized page rank scores for each start node. This modification satisfies the objective of the original optimization problem, because the weights for all neighbor nodes will be higher than all nodes that are not neighbors. It does not find the global optimum values to satisfy the objective function, but we used this methodology as a reasonable alternative for bipartite graphs.

After solving for the weight matrix, we then run the personalized page rank algorithm with these scores for each investor as a start node. The algorithm returns n number of the most probable destination nodes for each start node. This gives a set of the most probable future links predicted by the SRW algorithm. We then predict weights to the edges of our graph, using the supervised learning techniques similar to those discussed above.

VI. Results and Analysis

A. Network features

We implemented some basic link prediction algorithms to get an understanding of how well each of these network features would preform. For each algorithm, we removed a random set of the edges from our network to reserve for testing. For the initial baselines, the test set was 30% of the original network.

The first algorithm that we implemented as a trivial baseline, was random links. In this algorithm, we consider all of the links from investors to companies and select a random subset of them. This gives a precision of about 0.00089 and recall of about 0.00089. Note that the precision and recall values recorded here will all be the same, because we plotted the precision recall curve for each scoring function and chose the value where the precision and recall were equal (see Figure 5 as an example).

The next algorithm was preferential attachment for link prediction. As expected, this preformed poorly with a precision and recall of 0.00179. The results make sense, because the early stage investors often invest a little money in a lot of companies in order to maximize the probability that one of these companies will succeed. On the other hand, late stage investors invest a lot of money in a few proven companies in order to maximize their returns. Since early stage companies have less investors than late stage ones, investor nodes with high degree are more likely to be linked to company nodes with low degree, and vice-versa.

Noting these results, we created a sort of reverse preferential attachment algorithm, where the score of an edge is defined as the difference in degrees between the two edges. This gives high scores to links where on node has high degree and the other has low degree. This algorithm gave us a precision and recall of 0.05804.

The final algorithm was internal links for link prediction. We initially consider an unweighted projection of our graph into an investor-investor graph. We then calculate the cost of an edge by calculating the number of induced edges that are not in the original graph. This algorithm gave us a final precision and recall of 0.09204.

7.

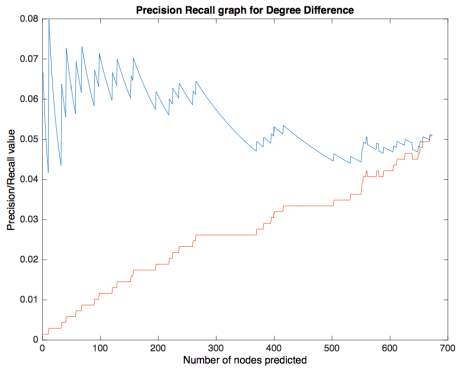


Fig. 5.

B. Supervised Learning with network features

In order to improve our models, we tested our training data with various variations of C and Gamma for the SVM model, and penalizations and C values for logistic regression. While Naive Bayes performed significantly better than our baseline SVM model, we choose to forego further training with Naive Bayes. We believe that Naive Bayes would not be able to incrase accuracy as dramatically as either SVM or Logistic Regression, since only a modification of the feature vectors would improve its results. Because much of the work in machine learning algorithms involves tuning parameters, we include here a few plots on the results of tuning C, the penalization norm, and using PCA on logistic regression, which ultimately gave us our optimal solution.

First, we show the effects of tuning C and the penalty norm for logistic regression in Fig. 6.

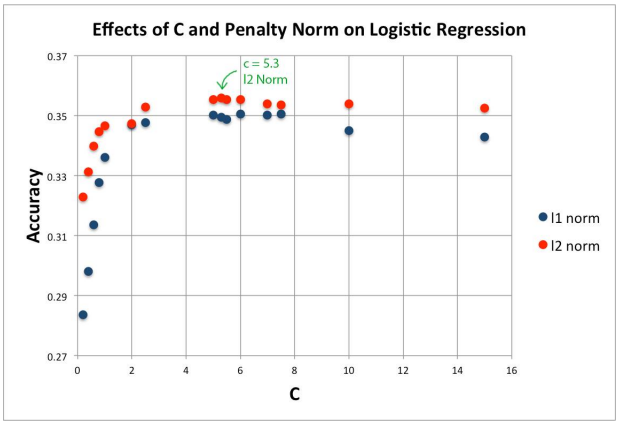


Fig. 6.

After plotting the accuracy results, we discovered that we obtained highest prediction accuracy for a C value of 5.3 using l-2 normalization.

Next, we show the effects of reducing the dimensionality of our feature vector on the logistic regression prediction accuracy using principle component analysis in Fig. 7. Because each principal component accounts for an amount of variability in the data we modified the number of principal components we wish to end up with.

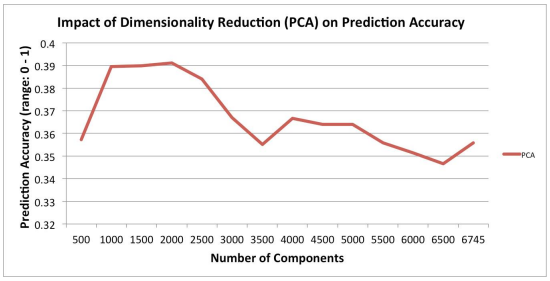


Fig. 7.

The results indicate that reducing our 6745 feature dimensions to a 2000-dimension feature space yields the highest prediction accuracy.

After normalizing each algorithm on our training set, we achieved the results in Fig. 8. Our SVM model did not work as well as expected, which could be due to to the kernel that we decided to use. Because we know that in practice SVMs and Logistic Regression perform comparably, we decided to use the RBF kernel for our SVM model. However, the non-linear RBF kernel gave us lower baseline results than even the Naive Bayes classifier. It is possible that our data is fairly linearly-separable. Additionally, it is possible that there was a lot of noise in the data, and Logisitc Regression (using maximum-likelihood techniques) handles noise better than SVMs. Since SVMs and Logistic Regression have comparable performance on linearly separable data, and since our Logistic Regression baseline was also higher for linear data - we continued tuning the logistic regression model rather than focusing on the SVM.

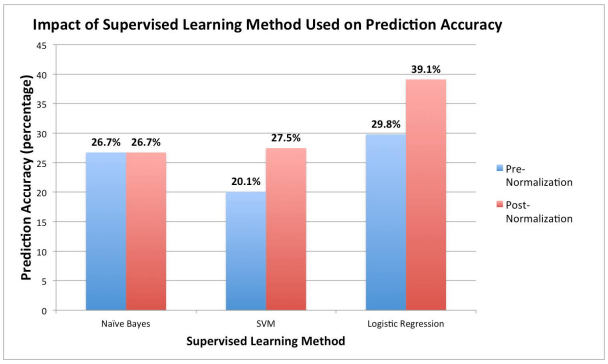


Fig. 8.

Since we are using a multi-classification algorithm, the precision and recall scores and ROC scores for a particular bucket may be artificially low (or high). Here, we plot the averaged ROC curve, and find that we are able to successfully predict links and investment amounts.

In Fig. 9 the area under the curve for the average ROC curve is 0.72, which is between a random classifier's area under the curve of 0.5 and a perfect classifier's area under the curve of 1.0.

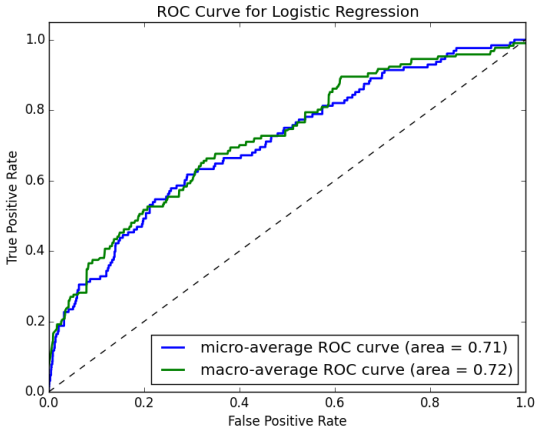


Fig. 9.

C. Supervised Random Walks

The SRW binary link prediction step has an accuracy lower than our results with supervised learning (Table 1). As a result, the multi classification step would only make this value worse. We noted that there are many ways to tune parameters for SRW to optimize this value. These include changing alpha, which is a parameter of the personalized page rank algorithm that gives the probability of terminating the walk at each step, or adding in random restarts. Our initial tests showed that tuning these parameters did not significantly improve our accuracy. We believe that the best method for significantly improving the performance of SRW is to better generalize the optimization problem to directed bipartite networks. Since our current supervised learning algorithms with network features were outperforming the SRW approach, we focused our resources on tuning parameters there and leave generalizing SRW as future work. As future work, we hope to design and test different optimization problems for learning page rank scores given node features in a directed bipartite graph. We also hope to test different methods of solving the optimization problem such as stochastic gradient descent.

	Accuracy
Naive Bayes	27.5%
Support Vector Machines	26.7%
Logistic Regression	39.1%
Supervised Random Walks	32.6%

Table 1. Final Performance

VII. Error analysis

Our SRW algorithm currently has worse performance than the supervised learning algorithms with network features. We believe that this is due to a simplified optimization problem. As future work, we hope to better generalize the SRW optimization problem to bipartite graphs and use it to increase our prediction success for investments. We also hope to optimize the algorithm, because the computation time of our current algorithm, given the size of our network edges and node feature space, is on the order of several hours.

One additional difficulty is that this network does not exhibit similar properties to many social networks. For example, higher preferential attachment scores is a good metric for link prediction, however in this graph lower preferential attachment scores are better metrics for link prediction. Therefore, link prediction algorithms that work well on some networks preform very poorly on this one.

In addition, this network is bipartite and relatively sparse, so we have to be careful when choosing what link prediction algorithms to consider, because it doesn't make sense to predict an edge between two companies or two investors in our network. We read many papers on link prediction in bipartite graphs and sparse networks in order to make educated decisions about what network features and algorithms to implement.

VIII. Conclusion and Future Work

In this paper we created a bipartite multigraph of investors and companies, with edges representing an investment from an investor in a company. Edge features included things like the series the investment was in and the total amount the company raised in that series, which the investment amount itself is proportional to. Node features included things about the investors and companies like their location and industry.

Our goal was to predict what companies investors should invest in, and how much they should invest. We used several techniques to predict these investment edges, including link prediction algorithms, supervised learning using network features, and supervised random walks. Because our graph was a sparse bipartite multigraph, several of the algorithms had to be modified to work well with our network. For example, we initially thought SRW would be the best method but due to the structure of our network, supervised learning performed better.

We were able to achieve the highest accuracy using supervised learning with network features, specifically with a logistic regression. We were able to successfully predict investments and investment amounts and significantly improved our results from the trivial link prediction baseline described in lecture (9%) to almost 40% accuracy.

For the future, we might try to enhance our investor-company network by adding in links that demonstrate connections between investors and companies other than just connections from investments. For example, if an investor and company shared common employees at some point, we would add in a link between the investor and company. In order to make our network less sparse and no longer bipartite, we could also add in the connections between companies, such as whether they have a common founder, and connections between investors. This enhanced and more dense network should hopefully give us better accuracy in link prediction and supervised random walks as our network will look more standard and less sparse.

IX. Breakdown of Work

- Tara: Supervised learning implementation and plotting, data parsing, feature statistic plotting, report
- Maya: Classifier for ROC and plotting, data parsing, poster, report
- Arushi: Problem formulation, data scraping, data parsing, network feature extraction implementation, SRW implementation, report

X. References

[1] Yuxian Eugene Liang, Daphne Yuan Sor-Tsyr. "Investors Are Social Animals: Predicting Investor Behavior using Social Network Features via Supervised Learning Approach."

[2] Neville, J. & Jensen, D. (2000). Iterative classification in relational data Proceedings of the AAAI 2000 Workshop Learning Statistical Models, pp. 42 - 49.

[3] Backstrom, L. & Leskovec, J. 2011. Supervised random walks: Predicting and recommending links in social networks. In WSDM '11: Proceedings of the ACM International Conference on Web Search and Data Mining.

[4] Kunegis, Jrme, Ernesto W. De Luca, and Sahin Albayrak. "The link prediction problem in bipartite networks." Computational intelligence for knowledge-based systems design. Springer Berlin Heidelberg, 2010. 380-389.

[5] Allali, Oussama, Clmence Magnien, and Matthieu Latapy. "Link prediction in bipartite graphs using internal links and weighted projection." Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. IEEE, 2011.

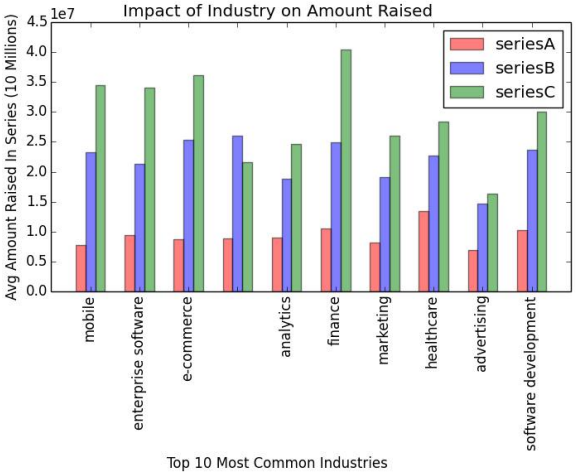
[6] Li, Xin, and Hsinchun Chen. "Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach." Decision Support Systems 54.2 (2013): 880-890.

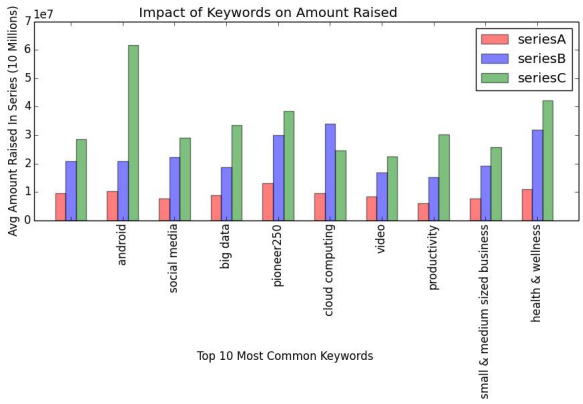
[7] Nguyen, Canh Hao, and Hiroshi Mamitsuka. "Latent feature kernels for link prediction on sparse graphs." Neural Networks and Learning Systems, IEEE Transactions on 23.11 (2012): 1793-1804.

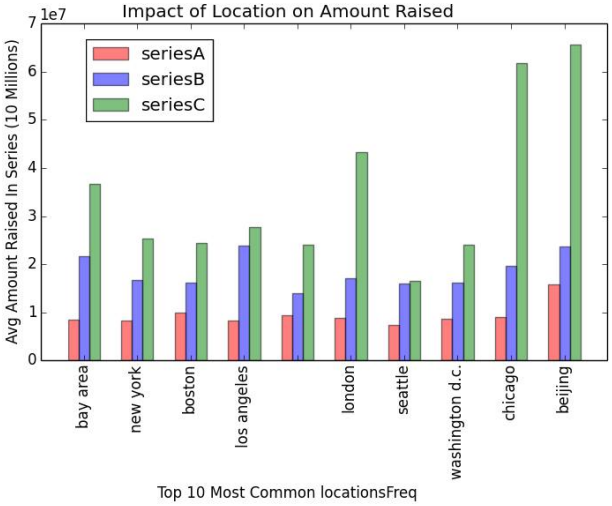
[8] Source: Crunchbase

[9] Source: Mattermark

XI. Appendix







All figures found in the Appendix illustrate the change in amount raised for some of the most common features a company could have attributed to themselves.