Predicting Friendship Links on Yelp

Shivank Goel : sg2359@cornell.edu

ABSTRACT

Predicting friendship links for any social media platform can have huge profit implications for the platform based businesses. The social media platforms which have more number of friendship connections between it's users, generates more user traffic on their platforms. A higher number of friendship links on a social platform can help the users to gain higher utility by sharing their personal preferences with the other users. Furthermore, more number of users on a platform can create a strong two-sided network effect for the business and attract more number of services and products on the platform. The higher user traffic on these platforms ultimately provides them the opportunities to increase their monetization and revenue. In this work we will be presenting a number of useful features that can be extracted from the Yelp dataset and compare the performance various models that can be used for evolving social network graphs to predict the friendship links between the users on the Yelp platform. All the code for this project is available at https://github.com/shivankgoel/yelplinkprediction.

INTRODUCTION

Link prediction in complex networks has attracted increasing attention from both physical and computer science communities. The algorithms can be used to extract missing information, identify spurious interactions, or evaluate network evolving mechanisms. Lu et. al.,[1] provides a brief summary on the progress of studies on link prediction. They mainly classified various approaches into three categories, i.e., similarity based methods, likelihood methods and probabilistic methods. Lichtenwalter et. al.,[2] shows that link prediction in sparse networks presents a significant challenge due to the inherent disproportion of links that can form, to links that do not form. They present an effective flow-based prediction algorithm that outperformed the then unsupervised link prediction methods by more than 30% (AUC). Bliss et. al., provide an approach to predicting future links in the evolving networks where nodes and links are added and removed over time. They apply the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to optimize weights which are used in a linear combination of sixteen neighborhood and node similarity indices. In this work I use the GraphSAGE algorithm to create node embeddings which I use for training the downstream classification models. GraphSAGE presents an approach to create embeddings for the evolving graphs or entirely new graphs.

The task of link prediction on social media networks can have many interesting applications. Effective methods for link prediction can help us analyze the promising interactions or collaborations that have not yet been utilized within these platforms. Recommending and facilitating these connections between the various kinds of the platform users can significantly generate more traffic and revenue for such kind of platforms.

This project develops effective ML models for predicting friendship links on the Yelp platform.[1] Yelp is a business directory service and crowd-sourced review forum, where the users of the platform write reviews about various businesses. Yelp has a monthly average of over 35 million unique visitors via their Yelp app and 69 million unique visitors via mobile web. Yelp uses automated software to recommend the most helpful and reliable reviews for the Yelp community by using various measures of quality, reliability, and activity on Yelp. The company revenue is primarily generated by online advertising. Moreover, Yelp platform further contributes significantly to the revenues of the businesses on their platform. Luca et, al.,[3] studies the impact of consumer reviews on the restaurant industry. They show that a one-star increase in Yelp rating leads to a 5-9 percent increase in revenue for restaurants which do not have a chain affiliation. Hence, for such review aggregation businesses, user engagement on their platform is a key factor in driving the revenues. In this work, I present a number of different node specific and the edge specific data features, which can be calculated and extracted from the dataset released by Yelp every year. I explore the relative usefulness of these features for the task of friendship links prediction on Yelp. I explore a number of different machine learning models and methods, for this task. I also look at the the recent developments in the literature for the prediction tasks over a graphical structure, i.e., node embeddings, and apply them to our setting.

Node-embeddings are one of the recent machine learning techniques to represent the information present in a graph. Qiu et. al.[4] presents a theoretical analysis to show that the node embedding methods implicitly perform matrix factorizations of the graph laplacian. These representations can not only capture the individual characteristics specific to the nodes, which are the Yelp users in our setting, but they can also incorporate the graph structure into these embeddings. Specifically, these embeddings can capture both the node's local role and it's global position within the graph, which provides more information to the downstream machine learning tasks. In this paper I will explore the relative usefulness of the user specific features, edge specific features and embedding features. These embedding features incorporate the information of the node's relative position within the overall graph structure.

RELATED WORK

The problem of the link prediction in a social network setting looks at how can we model the network by using features intrinsic to the network itself. There are many methods proposed for this task. Parimi et. al.[5] proposed a topic modelling approach using Latent Dirichlet Allocation (LDA) and several hand crafted features. Nowell et. al.[6] presents various metrics to calculate the distance between two nodes in a network. These distances can then be used for the downstream machine learning models to predict the links between the nodes in a graph. We will be using some of these distance metrices, such as, common neighbours, jaccard's coefficient and preferential attachment, to test the usefulness of these features in predicting the friendship links on the Yelp platform.

Fu et. al.,[7] proposes the structure of Yelp network as a two layered network where in the first layer, two users are connected if they are friends. They call this first layer as the social network. In the sec-

[1] https://www.yelp.com/

ond layer, two users are connected if they reviewed the restaurants in at least one of their same clusters and call it as co-foraging network. The link in the co-foraging network has a weight, which they define as the number of clusters of restaurants that the corresponding two users have ever reviewed. They model their task to predict the offline co-foraging behaviors of users, i.e., using the features from the social network as the input, they predict the corresponding link weight in the co-foraging network as the output. Apart from using several similarity indices that are often used for link prediction, such as common neighbors and jaccard coefficient, they also extract edge features by transforming the original graphs into line graphs. In this work I will be using features both from the social network and the co-foraging network as features to predict the friendship links on the Yelp platform. However, I am not using features from the corresponding line graph, which can be an interesting follow up for future work.

Hu et. al.,[8] uses the business review data from Yelp to study business rating prediction. They study the geographical neighborhood influence to users business rating behavior and observe that a businesss rating is positively correlated with its geographical neighbors rating. Incorporating geographical neighborhood over other factors , such as, business category, popularity, review content etc., allowed their model to perform better than the state-of-the-art methods at the time of the publication. In this work I am incorporating the geographical information by looking at the number of common cities for given two users in which they have written the reviews.

Seo et. al.,[9] presented an interpretable dual attention-based CNN model, which they call as *D-Attn* to predict the ratings of restaurants on Yelp platform. The model combines the review text and ratings from the aggregated reviews for product rating prediction. Li et. al.,[10] predict the rating of a user for a given product on a social network. They claim that generating predictions by analyzing the user-item rating matrix perform poorly when the matrix is sparse. Most of the social-network based recommender systems only consider direct friendships and they are less effective when the targeted user has few social connections. They proposed two models to incorporate the overlapping community regularization into the matrix factorization framework. In this paper I will use the embeddings generated from GraphSAGE algorithm which is originally inspired from the Graph Convolutional Networks (GCNs) proposed by Kipf et. al[11]. I will study the impact of incorporating these embeddings features to improve link prediction performance. GCNs care designed to automatically capture the node's local position and it's position within the global community.

Kipf et. al.,[11] introduced GCNs for semi-supervised classification. For every node in a graph they created a node embedding which then they used for the node classification task. These embeddings capture the structure of the graph beyond the simple assumption that the connected nodes in a graph are likely to share the same label. GraphSAGE[12] extended the GCN architecture to the task of inductive unsupervised learning, so that embeddings can be generated for unseen nodes . They proposed the aggregation of the node embeddings of the neighboring nodes using trainable functions. The idea of these approaches is to leverage the additional information that the edge connections in a graph may carry beyond the fact that the two nodes connected by an edge might be similar. I am using a similar strategy

as GraphSage to aggregate the information from the neighboring nodes.

DATASET

The Yelp Dataset[2] is a subset of their businesses, reviews, and user data which is publicly available for personal and academic purposes. The Yelp dataset includes the data of over 1.6 million users spread across United States and Canada. The data includes information from major ten metropolitan areas in these countries. I construct the Yelp friendship network using this data. Specifically, I consider each user as a node of the graph. Two given nodes have an un-directed edge between them if the corresponding users are friends with each other. In Yelp dataset, out of the 1.6 million users, 948,076 users have at-least one friendship link with some other user in the dataset. The total number of friendship links in the data are 7,392,305. However, this is a highly sparsely connected graph, since it includes 175,804 connected components. The largest connected component contains 765,251 nodes and 7,385,225 edges. We note here that the second largest component contains only 7 nodes and 8 edges.

Creating Train and Test Set of Edges

For the purpose of this project, I will be using only a small sub-graph of the largest connected component. Specifically, I picked the first 20,000 users of the largest connected component in the order of the user ids provided in the Yelp dataset. Then, I randomly allocated these 20,000 users into two equal train and test set of users. For the training purposes, I found the largest connected component amongst the train set of users, to create the *Train Connected Component*. Finally, the corresponding train connected component contains 7,165 nodes and 73,429 edges. Similarly, I found the largest connected component amongst the test set of users to create the *Test Connected Component*. The test component which is obtained contains 7,151 nodes and 71,839 edges. Table 1 contains the summary statistics for various graphical components discussed in this section.

Dataset	**Number of Nodes**	**Number of Edges**	**Average Node Degree**
Complete Dataset	1,637,138	7,392,305	4.51
Nodes with atleast one friend	948,076	7,392,305	7.80
Largest Connected Component	765,251	7,385,225	9.65
Training Connected Component	7,165	73,429	10.24
Test Connected Component	7,151	71,839	10.04

Table 1. Data Summary

To create the training dataset, I used all the 73,429 edges in the training connected component and use the corresponding nodes for these edges as the positively labeled examples, i.e., the pairs with an edge between them. We then randomly sampled an equal number of 73,429 node-pairs which do not have an edge between them from the training connected component and use these pairs as the negatively labeled examples. I followed a similar strategy to identify the positive and negative labeled pairs from the test connected component to create the test dataset. Thus, I obtained 71,839 positively labeled pairs and an

[2]https://www.yelp.com/dataset

equal number of negatively labeled pairs for the test dataset.

FEATURE EXTRACTION

Node Specific Features

For each user on the Yelp dataset I identify a number of user-specific features. These features depict the amount of the user engagement on the platform, such as, how many compliments a user has given or received so far on Yelp platform. The features also include what is the average rating given by the user. For reference, these features have been listed in the Table 2.

Feature	**Description**
Review Count	The number of reviews written by the user
Useful	The number of useful votes sent by the user
Funny	The number of funny votes sent by the user
Cool	The number of cool votes sent by the user
Fans	The number of fans the user has
Elite	The number of years user had an elite status
Average Rating	The average of all the ratings given by the user
Hot Compliments	The number of hot compliments received by the user
More Compliments	The number of more compliments received by the user
Profile Compliments	The number of profile compliments received by the user
Cute Compliments	The number of cute compliments received by the user
List Compliments	The number of list compliments received by the user
Note Compliments	The number of note compliments received by the user
Plain Compliments	The number of plain compliments received by the user
Cool Compliments	The number of cool compliments received by the user
Funny Compliments	The number of funny compliments received by the user
Writer Compliments	The number of writer compliments received by the user
Photos Compliments	The number of photo compliments received by the user

Table 2. User (Node) Specific Features

Edge Specific Features

Apart from the user data, the Yelp Dataset also includes the data of their businesses and the reviews received by these businesses. The platform contains the data of over 6.6 million reviews. Each review on Yelp is associated with a unique user id which refers to the user who wrote the review, and a business id which refers to the business for which the review is written. The dataset contains the details of total $192,609$ businesses. The business data maps the given business id to it's name, location, average rating and other business specific attributes. Using this data for business and reviews, I identify common neighbor features, jaccard coefficient features and preferential attachment features for a given pair of nodes. Specifically, for a given user I first identify the set of businesses for which the user has written a review, the set of cities in which these businesses are located and the set of friends a user has. Then I use these sets to obtain the three type of features. Common neighbors feature for a given pair of users i and j is simply defined as, $s_{ij}^{CN} = |\Gamma(i) \cap \Gamma(j)|$, where $\Gamma(i)$ denotes the the set of distinct elements (which can be friends, businesses or cities) for the i^{th} user. Using the similar notation, Jaccard coefficient is defined

as, $s_{ij}^{JAC} = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}$, and preferential attachment is defined as $s_{ij}^{PA} = |\Gamma(i)| * |\Gamma(j)|$. I call these features as edge specific features because they are determined given a pair of users, although a friendship link between them may or may not exist. These features have been listed in Table 3.

Feature	**Description**
Common Businesses	The number of common businesses
Common Cities	The number of common cities
Common Friends	The number of common friends
Jaccard Businesses	Ratio of number of common businesses to union number of businesses
Jaccard Cities	Ratio of number of common cities to union number of cities
Jaccard Friends	Ratio of number of common friends to union number of friends
Preferential Attachment Businesses	Multiplying the number of businesses for the two users
Preferential Attachment Cities	Multiplying the number of cities for the two users
Preferential Attachment Friends	Multiplying the number of friends for the two users

Table 3. Edge Specific Features

Embedding Features

I also obtained the embedding features for every node using the GraphSAGE algorithm. I will discuss more on how these features are obtained in a later section.

TASK

I model the task of friendship link prediction on Yelp platform as a binary classification task. Specifically, for the given two users on Yelp, the task is to predict whether there is an edge that exist between those users. A machine learning model which is good at predicting an edge between two users, can then be used to recommend friends to new users on the platform. For training and testing the task performance, I have curated a dataset of a total of $290,536$ pairs of users which includes an equal number of the pairs which are positively labeled, i.e., they have a friendship link between them and the negatively labeled pairs without any friendship link between them.

MODELS & RESULTS

I will be training two models, i.e., a logistic regression model and a multi layer neural perceptron model. For each of these models I will be using three set of features as mentioned previously, i.e., the node features, edge features and embedding features. First I will present results using only the node features and edge features. I will then extend these results by using the embedding features. However, I find that using embedding features only provide us insignificant improvements over the other features I use. This might be the case, because Yelp graphical structure might not contain any other extra useful information, which can be captured by these embeddings, for predicting friendship links on the yelp. Results suggest that the edge features are most useful in determining the possibility of friendship link between two users. On the other hand node features relatively does not contain much information that can be useful to predict friendship links. We also find that using a non-linear classifier, such as multi layer perceptron for such tasks, significantly improves the performance.

Logistic Regression

In this section I present the results of the logistic regression model for a number of different input features that are used for the training. In all the results the logistic regression model uses, Limited Memory BroydenFletcherGoldfarbShanno (LM-BFGS) solver. I first train the model on the *Node Features*. For a given pair of users I concatenate the 18 user specific features listed in Table 2 for each of the users in pair, to get a feature vector of size 36. Using the node features, I obtain a baseline accuracy of 0.735 and a F-score of 0.757. This result suggest that the node features are not much helpful by themselves in determining how likely two users are to become friends on the Yelp platform. However training on the *Edge Features* significantly improved the model performance. Using the common neighbor features with a model as simple as logistic regression achieves an high accuracy of 0.912 and a F-score of 0.908. The results suggest that these features can be best exploited by the Yelp platform for predicting links on their platform. Ablation studies for edge features shows that the number of common friends is the most useful features among these for the link predicition. Further, using number of common friends gives the highest precision for the logistic regression model, showing that the common friends can highly determine if two people will eventually become friends themselves.

Surprisingly, using the preferential attachment features does not allow the model to learn anything useful, and gives us an accuracy of fifty percent, which is as good as a random prediction. Hence, I will be discarding preferential attachment features for any of my future analysis. Using jaccard features instead of common neighbor features give slightly inferior results. However, using both the common neighbor features and the jaccard features gives us the best accuracy of 0.913 and a F-score of 0.909 for the logistic regression model.

Using both the node features and the common neighbor edge features does not overall improve over using just the common neighbor edge features, but gave the best recall for the logistic regression model. In general, in my results I find that adding the node features usually add noise and decreases model performance. Also, a depreciation in results can be due to the limited modelling capacity of a simple model such as logistic regression. I found significant improvement in these results using the multi layer neural perceptron model. All the results for the logistic regression model have been summarized in the Table 4.

Multi Layer Perceptron (MLP)

The multi layer perceptron is a deep neural network model which contain two hidden layers and one output layer on top which a sigmoid activation function is applied. Since, I will be obtaining results for multi layer perceptron using different set of input features, I follow the following rule to determine the number of hidden units in the hidden layers. The first hidden layer consists of two thirds neurons compared to the number of number of input features. The second hidden layer further contains two thirds of neurons compared to the first hidden layer. The two hidden layers uses the ‘relu’ activation function.

Features	Accuracy	Precision	Recall	F-Score
Edge Specific Features				
Common Neighbor Features + Jaccard Edge Features	**0.913**	0.957	0.866	**0.909**
Common Neighbor Features	0.912	0.956	0.864	0.908
Jaccard Edge Features	0.871	0.950	0.783	0.858
Preferential Attachments Edge Features	0.500	0.500	1.000	0.667
Common Friends + Cities	0.911	0.955	0.863	0.906
Common Friends + Business	0.909	0.962	0.851	0.903
Common Business + Cities	0.742	0.793	0.655	0.718
Common Friends	0.904	**0.964**	0.839	0.897
Common Business	0.782	0.872	0.535	0.663
Common Cities	0.685	0.796	0.497	0.612
Node Specific Features				
Node Features	0.735	0.699	0.827	0.757
Both Node and Edge Features				
Node Features + Common Neighbor features	0.875	0.866	**0.888**	0.877
Node Features + Jaccard Edge Features	0.736	0.698	0.831	0.759

Table 4. Performance of Logistic Regression Model

The final layer contains a single neuron which outputs probability of edge between given two nodes using a sigmoid activation function. I concatenate all the input features together and use them as an input the MLP model. The results for this model has been summarized in the Table 5. I note that for all the possible input set of features, MLP model performs better than the corresponding results using logistic regression model.

Features	Accuracy	Precision	Recall	F-Score
Edge Features				
Common Neighbor Features + Jaccard Edge Features	**0.932**	0.919	**0.947**	**0.933**
Common Neighbor Features	0.931	0.918	**0.947**	**0.932**
Jaccard Edge Features	0.906	0.934	0.874	0.903
Node Specific Features				
Node Features	0.823	0.847	0.789	0.817
Both Node and Edge Specific Features				
Node Features + Common Neighbor Features	**0.933**	**0.938**	0.927	**0.932**
Node Features + Jaccard Edge Features	0.832	0.809	0.869	0.838

Table 5. Performance of Multi Layer Perceptron

Embedding Features

Finally, I incorporate the embedding features into the above two models. I obtain the graph node embeddings using the GraphSAGE[12] algorithm, which is an extension of the Graph Convolutional Networks (GCNs) so that we can obtain embeddings for the unseen nodes. GCNs were introduced by Kipf et. al.,[11] to encode the graph structure in form of node embeddings using a neural network model. The objective was that these embeddings encode both the node's local position within a graph and it's gloabl

position within the neighboring community of nodes. The key idea behind GraphSAGE is that at every layer of the convolutional neural network, for each node it aggregates the information of the neighboring nodes using trainable functions. It then passes this aggregated information of the neighboring nodes along with the existing node representations through a fully connected neural network layer to obtain the new feature representation of the given node. For the first layer of GraphSAGE, the features are simply the node features in our case. Algorithm 1 describes the mathematical details of this procedure. For obtaining the embeddings, I use $K = 2$, i.e., two aggregation layers and two combine layers. I modified the original GraphSAGE implementation to make it suitable for the edge classification task instead of the node classification, and train the algorithm in a supervised fashion. Specifically, after getting the encoding of each node, I pass the example pair of nodes from the train set of nodes, which uses these encodings to predict the edge between them. The loss obtained is then back propagated, which updates the parameter for the aggregator layers and the combine layers. Figure 1 shows the training loss with the batch iteration, while training the supervised GraphSAGE algorithm. Every batch contains 200 training examples. Further, I used an embedding size of 200 while training.I tried various sizes for the node embeddings, ranging from 10 to 200. I found that results fluctuate a lot (varying results for same embedding size) but does not vary significantly while chossing between these specifications.

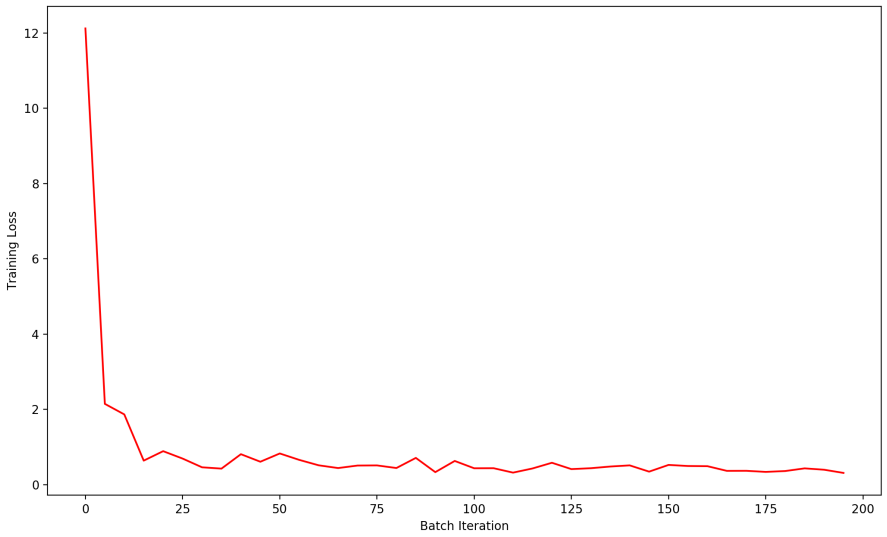

Figure 1. Training loss of node embeddings

for $k = 1....K$ **do**
  for $\forall v \in V$ **do**
    Aggregation Layer : $h^k_{N(v)} \leftarrow AGGREGATE_k(h^{k-1}_u, \forall u \in N(v))$
    Combine Layer : $h^k_v \leftarrow \sigma(W^k \cdot CONCAT(h^{k-1}_v, h^k_{N(v)}))$
  end
end

Algorithm 1: GraphSAGE Algorithm

Table 6 shows the results obtained for the node embedding features. The first row of the table shows the result obtained on the test by using the outputs from the supervised GraphSAGE model directly. Apart from that, I obtained results by using the node embeddings obtained as an input the models we discussed previously. We get the accuracy for the supervised graph-sage to be similar to the case when we used only node ambeddings to train MLP. This shows that embedding features does not contain much more information from the structure of the graph, than what we can get from the node features itself. I find that training the logistic regression model on top of these embedding features and other features described previously does not improve upon the performance obtained by the graphsage algorithm, again may be due to limited modelling capacity. However, training the MLP on top of these embedding features improves the F-score from 0.833 to 0.870. Further, incorporating the common neighbors and jaccard edge-features gives us the best accuracy and F1-score for this paper, i.e., 0.934. However, this is not much improvement over using simply the common neighbors and jaccard edge-features with MLP model which gave us an F-score of 0.933. This might be the case as suggested before, because the extra information regarding the node's position in graph, that is captured by the node embeddings, might not be much useful in predicting a friendship link between given two nodes. Still, node embeddings helped me to obtain the benchmark results for prediction of friendship links on the Yelp platform.

Features	**Accuracy**	**Precision**	**Recall**	**F-Score**
Supervised GraphSage				
Embed Features	0.838	0.811	0.857	0.833
Logistic Regression				
Embed Features	0.807	0.814	0.795	0.805
+ Node Features	0.807	0.804	0.811	0.808
+ Edge Features(Common + Jaccard)+ Node Features	0.821	0.826	0.814	0.820
+ Edge Features(Common + Jaccard)	0.834	0.858	0.802	0.829
Multi Layer Perceptron				
Embed Features	0.863	0.828	0.916	0.870
+ Node Features	0.868	0.847	0.898	0.872
+ Edge Features(including Jaccard)+ Node Features	0.931	0.918	**0.947**	0.932
+ Edge Features(including Jaccard)	**0.934**	**0.935**	0.934	**0.934**

Table 6. Performance of Node Embedding Features using GraphSAGE

CONCLUSION

The friendship prediction task can help social media platforms in better monetization and user engagement. The type of input features for the machine learning models that give a superior performance for predicting friendship links can be peculiar to the particular social media platform. I identify such features for the Yelp platform, and find that the common neighbor features and jaccard edge features often give the best performance for Yelp platform. Further I find that the deep learning models, such as, MLP can provide significant improvements over the linear models for the link prediction task. However, I find little evidence that the information from the node embeddings is helpful on top of the edge features in improving the accuracy of the models.

REFERENCES

1. L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.
2. R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 243–252, ACM, 2010.
3. M. Luca, "Reviews, reputation, and revenue: The case of yelp. com," *Com (March 15, 2016). Harvard Business School NOM Unit Working Paper*, no. 12-016, 2016.
4. J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 459–467, ACM, 2018.
5. R. Parimi and D. Caragea, "Predicting friendship links in social networks using a topic modeling approach," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 75–86, Springer, 2011.
6. D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
7. C. Fu, M. Zhao, L. Fan, X. Chen, J. Chen, Z. Wu, Y. Xia, and Q. Xuan, "Link weight prediction using supervised learning methods and its application to yelp layered network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1507–1518, 2018.
8. L. Hu, A. Sun, and Y. Liu, "Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 345–354, ACM, 2014.
9. S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 297–305, ACM, 2017.
10. H. Li, D. Wu, W. Tang, and N. Mamoulis, "Overlapping community regularization for rating prediction in social recommender systems," in *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 27–34, ACM, 2015.
11. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
12. W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.