Link Prediction in Bipartite Networks - Predicting Yelp Reviews

Kevin (Junhui) Mao, Xiao Cai, Ya Wang

December 8, 2015

Abstract

In this paper, we aim to predict new user reviews on businesses in the Yelp social network. We formulate this as a network link prediction problem by modeling Yelp dataset as a bipartite network between users and businesses. We implement link prediction algorithms with various proximity metrics, thoroughly evaluate the effectiveness of each algorithm and conclude that Delta, Adamic-Adar and Common Neighbors algorithms perform the best in precision.

1 Introduction

Social networks are dynamic and evolve over time. Predicting novel links that may occur in the future has many applications in advertising, information retrieval, national security, urban planning, recommendation systems, network analysis, etc. Yelp Dataset Challenges [11] gives data scientists around the world an opportunity to use the data for researches on a wide spectrum of fields. In this paper, we model the Yelp dataset as a bipartite network between users and businesses. We implement the algorithms of link prediction with various proximity metrics [8][1] and aim to predict future user-business relationship in terms of reviews in Yelp social network. We then compare the effectiveness of different proximity metrics and conclude with our final thoughts on the algorithms and possible directions for future research.

2 Prior Related Work

As a key research problem in dynamic network analysis, there are abundant literatures on link prediction [1][4][5][7][8]. Liben-Nowell and Kleinberg [8] surveyed an array of methods for link prediction and divided them into three categories: Node Neighborhood Based Methods (common neighbors, preferential attachment, Jaccard coefficient, Adamic-Adar, etc.), All Paths Based Methodologies (PageRank, SimRank, etc.) and Higher Level Approaches (low-rank approximation, unseen bigrams and clustering).

Allali et al [1] proposed a new approach of internal link prediction for bipartite graphs. They defined the concepts of *induced link* and *internal links* in bipartite graphs.They first use induced link to create a projection graph, assign neighborhood based weights to the edges in the projected graph, then identify internal links which will probably appear in the future. Once the internal links are generated, their future appearance is predicted by comparing the assigned weight against certain threshold.

Hasan et al [7] summarized major categories of link prediction methods: network feature based classification model, probabilistic model and linear algebraic model. They discussed in detail the characteristics of each type of link prediction methods. For example, the probabilities model, though can easily incorporate both features inside and outside the network dimension, has the disadvantage of computational complexity. They enumerated the challenges frequently associated with link prediction, such as those imposed by the sheer size, the dynamic nature and the extreme *class skewness* of the social network and discussed methods customized to cope with these challenges.

3 Data Collection and Processing

We collected Yelp Dataset Challenges [11] to serve as base data for our project.

3.1 Dataset Information

The Yelp data contains five datasets: 366K users, 61K businesses, 1.5M reviews, 500K tips and 45K aggregated check-ins over time for the businesses.

3.2 Preprocessing

We are mainly interested in the review dataset, which is modeled as a bipartite network for link prediction between users and businesses. The meta data is in JSON format. We preprocessed the data and converted it into TSV format suitable for network analysis. The preprocessed review data has these fields: userId, businessId, stars, date and votes.

3.3 Analysis

We derived some summary statistics to understand the general network structure. All the data analysis was performed on the entire review dataset. There are 10 WCCs, but only one is dominant with 99.64% nodes. The degree distribution follows power law with $xmin = 1.0$ and $alpha = 1.621$

Table 1: Yelp Social Network Property

Property	**Value**
Number of Users	366,715
Number of Businesses	60,785
Number of Reviews	1,521,160
Number of Nodes	427,500
Number of Edges	1,521,160
Diameter	14
90% Effective Diameter	5.95
MaxWccSize	425,978
Connected Component Size	0.9964

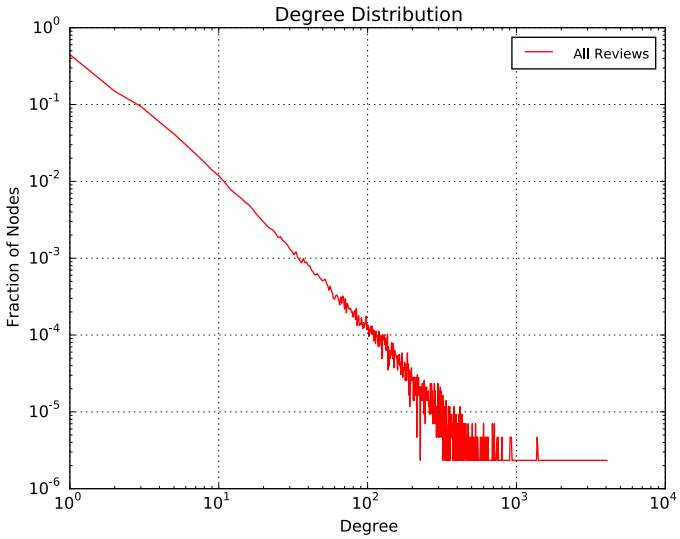

Figure 1: Yelp Reviews Degree Distribution ($xmin = 1.0$ and $alpha = 1.621$)

4 Methodology and Algorithms

Given a training interval $[t_0, t_0']$ and a test interval $[t_1, t_1']$, where $t_0' < t_1$, the link prediction task is to output a ranked list of links not present in $G[t_0, t_0']$, but are predicted to appear in $G[t_1, t_1']$. The link prediction process consists of four major steps:

- Create core data
- Apply link prediction algorithms
- Predict a ranked list of links unseen in the training set
- Evaluate prediction performance

Each of these steps will be elaborated in the following subsections.

4.1 Core Data Creation

We split the Yelp reviews temporally into training set and test set - all reviews before 2014 comprise the training dataset and all reviews after 2014 comprise the test dataset. The training set has 68% of the entire reviews, and the test set 32% of the entire reviews.

Most of our link prediction algorithms requires $O(MN)$ operations, where M is the number of users and N is the number businesses. It becomes computationally infeasible when the network is very sparse and huge. Hence we randomly sample 1% of unique users with $degree \geq 5$ to create the core datasets. We use a random seed to make our result reproducible. Algorithm 1 depicts how to generate the core datasets and Table 2 provides a summary of the core datasets.

Algorithm 1 Core Data Creation

Per section 3.3, the network has a dominant WCC covering 99% nodes. We will first extract the MaxWcc from the training set. Assume $G_{train}[t_0, t_0']$ is the network of full training data and $G_{test}[t_1, t_1']$ of full test data, we create $TrainCore(U_{tr}, B_{tr}, E_{tr})$ and $TestCore(U_{te}, B_{te}, E_{te})$ networks as follows

1. Extract MaxWcc of $G_{train}[t_0, t_0']$
2. Generate U_0 as a set of unique users in MaxWcc, such that $\forall u \in U_0$, $degree(u) \geq 5$
3. Generate B_0 as a set of unique businesses in MaxWcc, such that $\forall b \in B_0$, $degree(b) \geq 5$
4. Generate $U \subset U_0$ as a set of randomly selected 1% users from U_0
5. Create $TrainCore(U_{tr}, B_{tr}, E_{tr})$ as a subgraph of $G_{train}[t_0, t_0']$, where $U_{tr} \subset U$ and $B_{tr} \subset B_0$
6. Create $TestCore(U_{te}, B_{te}, E_{te})$ as a subgraph of $G_{test}[t_1, t_1']$, where $U_{te} \subset U_{tr}$ and $B_{te} \subset B_{tr}$ and $E_{te} \cap E_{tr} = \emptyset$

Table 2: Core Dataset Summary

Property	**TrainCore**	**TestCore**
Number of Users	5,569	2,583
Number of Businesses	27,537	10,412
Number of Reviews	232,720	25,401

4.2 Algorithms

We implemented three classes of algorithms for link prediction:

- scores based on proximity metrics using node neighborhood features
- scores based on matrix factorization using user ratings on businesses
- random algorithm that serves as baseline for comparison

For each candidate new edge, each algorithm produces a score for it and predicts whether that edge will exist in the future.

Node Neighborhood Definition Before describing each algorithm, we first define node neighborhood in a *user-business* bipartite network. The intuition is that users who review the same businesses are similar to each other.

- $\mathcal{N}(u)$:= set of users who are co-reviewers of a given *user* u for some *business* b
- $\mathcal{N}(b)$:= set of users who reviewed business b

Thus both sets only contain users, which simplifies the definitions for the proximity metrics below.

4.2.1 Random Baseline

This algorithm simply assigns a random score to each candidate edge, and serves as a baseline when evaluating different algorithms.

4.2.2 Common Neighbors

$$score(u, b) = |\mathcal{N}(u) \cap \mathcal{N}(b)|$$

When the number of common neighbors grows higher, the chance that u and b will have a link between them in the future increases.

4.2.3 Jaccard Coefficient

$$score(u, b) = \frac{|\mathcal{N}(u) \cap \mathcal{N}(b)|}{|\mathcal{N}(u) \cup \mathcal{N}(b)|}$$

The Common Neighbors metric can't appropriately measure the strength of relationship between two nodes when the existence of common neighbors is simply owing to the existence of lots of neighbors for each node. Jaccard Coefficient, by contrast, is normalized and is designed to handle this kind of cases.

4.2.4 Adamic-Adar

$$score(u, b) = \sum_{z \in \mathcal{N}(u) \cap \mathcal{N}(b)} \frac{1}{\log |\mathcal{N}(z)|}$$

Adamic-Adar weighs the common neighbors with smaller degree more heavily. Adamic-Adar metric improves Jaccard Coefficient by accounting for the nonlinear nature of the possible influence of proximity, i.e., common neighbors, on the strength of connection.

4.2.5 Delta

$$score(u, b) = \sum_{z \in \mathcal{N}(u) \cap \mathcal{N}(b)} \frac{2}{|\mathcal{N}(z)|.(|\mathcal{N}(z)| - 1)}$$

It measures the relative importance of a shared friend (co-reviewer) in terms of how many friends the user has.

4.2.6 Preferential Attachment

$$score(u, b) = |\mathcal{N}(u)|.|\mathcal{N}(b)|$$

Assume the chance that u will review b is proportional to the number of other users who have already reviewed b, and it's correlated with the number of users who have similar *tastes* as u.

4.2.7 Matrix Factorization

$$score(u, b) = \mathbf{p}_u^T \mathbf{q}_b$$

Matrix factorization (MF) and its variants cover a wide range of applications including recommender systems and link prediction. Given an incomplete matrix $R \in \mathbb{R}^{m \times n}$, MF is a process to find two factor matrices $P \in \mathbb{R}^{k \times m}$ and $Q \in \mathbb{R}^{k \times n}$, such that $R \simeq P^T Q$, where k is the prescribed number of latent factors. We use LIBMF [2] to train a model with factor $k = 80$, which seems working best on the Yelp dataset. Once the factor matrices P and Q are learned, we predict $score(u, b)$ for an unobserved edge (u, b) with the formula listed above.

4.3 Link Prediction

To reduce the false positive links and thus possibly increase prediction precision, we use threshold of *minimum-common-neighbors* to prune the candidate links. Later on, we will evaluate different algorithms for each threshold.

Algorithm 2 Link Prediction

Let $TrainCore(U_{tr}, B_{tr}, E_{tr})$ and $TestCore(U_{te}, B_{te}, E_{te})$ be the training and test core datasets created by Algorithm 1. Neighborhood $\mathcal{N}(u)$ and $\mathcal{N}(b)$ are defined by section 4.2. For each of the $threshold \in \{1, 2, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70\}$ of *minimum-common-neighbors*, run the following algorithm.

1. For each user-business pair $(u, b) \in U_{tr} \times B_{tr}$:
   (a) if $(u, b) \in E_{tr}$: continue
   (b) otherwise compute proximity $score(u, b)$, using some algorithm defined in section 4.2
   (c) if $|\mathcal{N}(u) \cap \mathcal{N}(b)| < threshold$: continue
   (d) otherwise output $(u, b, score(u, b))$
2. Sort the (u, b) pairs by decreasing order of $score(u, b)$
3. Predict top $|E_{te}|$ pairs of (u, b) as new links

4.4 Performance Evaluation

The set of predicted links generated by Algorithm 2 is evaluated against $TestCore(U_{te}, B_{te}, E_{te})$. We use precision, precision@N and recall to measure prediction performance. Details will be explained in the next section.

5 Experimental Results

For the predicted $|E_{te}| = 25401$ pairs of new (u, b) edges, we identify the correctly predicted links that appear in $TestCore(U_{te}, B_{te}, E_{te})$ and denote this true positive set as TP. Then $precision = \frac{|TP|}{|E_{te}|}$ is computed.

- **Precision** - Table 3 shows the result when minimum-common-neighbors $threshold = 30$ is chosen for pruning candidate links. We chose threshold as 30 to represent the moderate case so we can avoid extreme cases like very large or small thresholds. The algorithm gives the best performance is highlighted in the table. Surprisingly Matrix Factorization behaves even worse than the random algorithm. This is perhaps due to the fact that users have different criteria for rating. For example, some users may rarely rate 5 stars for any business, while some users are happy to rate 5 stars if they are satisfied with the services provided by the businesses they visited.

Table 3: Experimental Results (pruning threshold=30)

Algorithm	**Matched links**	**Precision (%)**
Adamic-Adar	396	1.56
Common Neighbors	385	1.52
Delta	585	**2.30**
Jaccard	255	1.00
Matrix Factorization	82	0.32
Preferential-Attachment	293	1.15
Random	139	0.55

- **Precision vs Pruning Threshold** - We also investigated how the prediction precision varies when we choose different minimum-common-neighbors *threshold* to prune candidate links. Interestingly, Figure 2 shows that Adamic-Adar and Common-Neighbors remain constant even when we vary the pruning threshold, but Delta performs better when $threshold > 10$, performs best when $threshold = 30$, then slightly drops down when $threshold > 30$, but is still better in general. One more thing we'd like to point out is that performance of Adamic-Adar is highly correlated to Common-Neighbors. The possible reason is that both algorithms use the same set of use-business neighborhood data for scoring, and both scores are dependent in linear space.

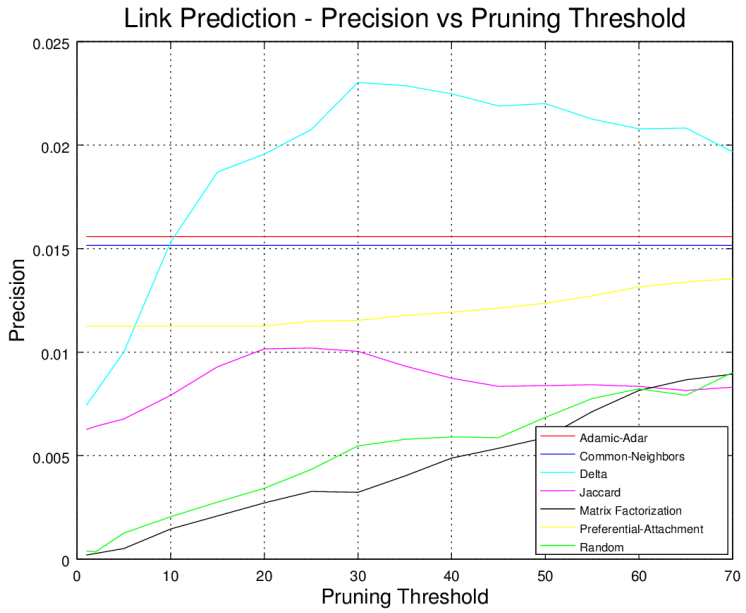

Figure 2: Precision vs Pruning Threshold

- **Precision@N** - In the real world, link prediction can be used to recommend products to relevant users. Hence instead of the overall precision, we would like to know how well each algorithm performs in terms of precision@N for $N = 10, 20, 30, 40, 50$. Figure 3 shows that Adamic-Adar and Common-Neighbors perform way better than other algorithms. It's also notable that Preferential-Attachment performs pretty well when $N \geq 20$.

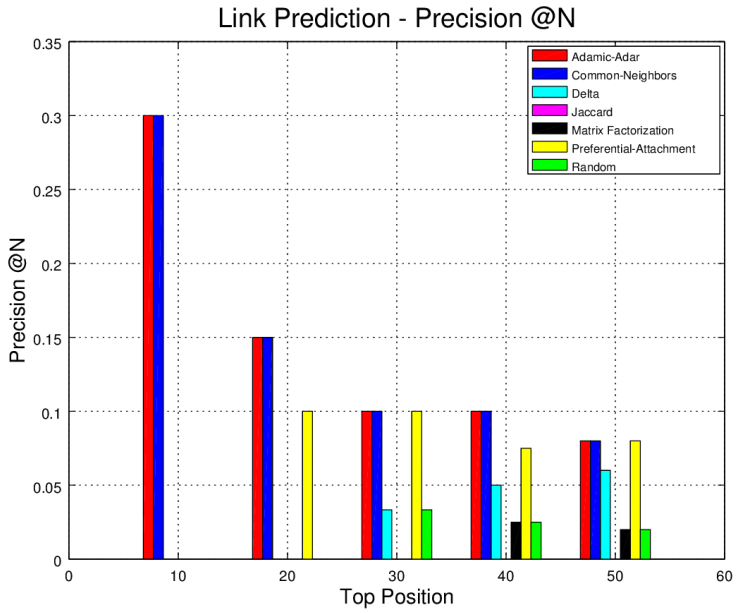

Figure 3: Precision @N (N=10,20,30,40,50)

- **Recall vs Pruning Threshold** - Let C be the set of of all links generated by Algorithm 2 at *step-1* with some pruning threshold. Then $C \cap E_{te}$ is the set of links that are relevant. We define $recall = \frac{|TP|}{|C \cap E_{te}|}$ as the probability that a relevant link is predicted as true positive. We'd like to understand how recall varies for each algorithm when the pruning threshold changes. Again, Delta, Adamic-Adar and Common-Neighbors perform better than other algorithms in terms of recall.

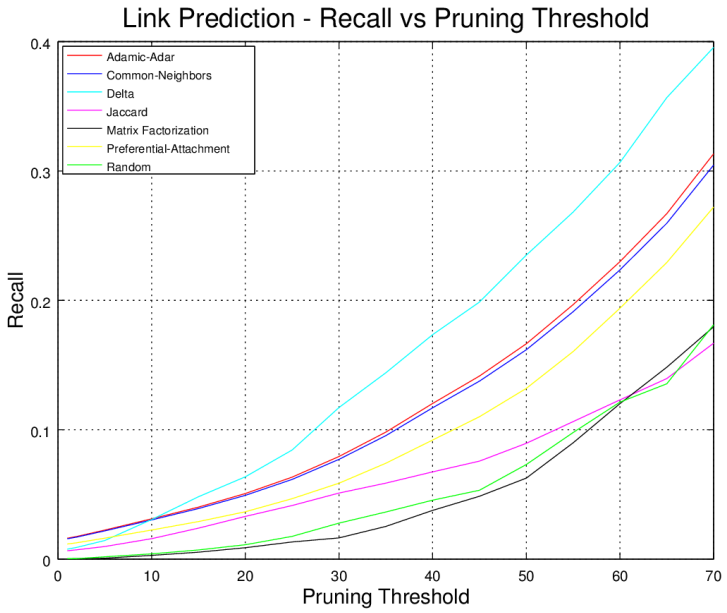

Figure 4: Recall vs Pruning Threshold

6 Conclusion

In this paper, we explored different link prediction algorithms with various proximity metrics to predict future user reviews on businesses in Yelp network. We then compared the effectiveness of different algorithms in terms of precision, precision@N and recall. We conclude our final thoughts on the algorithms as follows.

- Adamic-Adar and Common-Neighbors perform surprisingly well, despite being very simple. Furthermore they are very stable regardless of what pruning threshold is used.
- With a well-chosen pruning threshold (e.g. threshold=30), Delta algorithm performs 1.5 times better than Common-Neighbors.
- Matrix Factorization based user rating metrics performs worse than the random algorithm, which indicates that user ratings may be subjective and highly biased.
- The prediction precisions of the algorithms are generally low, there are still much room for improvement.
- Most link prediction algorithms are very computational expensive and require $O(MN)$ operations. We had to perform some of our tasks (such as link candidate generation and sorting) on Hadoop.

7 Future Work

The low precision issue implies that proximity measures alone may not be sufficient for ranking. We could make use of other features like votes, business category similarity or even review text to understand what topics a user is most interested in. For future directions, one could contrast our results with other algorithms such as Internal Link Approach [1], Supervised Learning Approach [6], Supervised Random Walks [3], etc. Our code is available in GitHub [9].

References

[1] Oussama Allali, Clemence Magnien, and Matthieu Latapy. Link prediction in bipartite graphs using internal links and weighted projection. 2011.

[2] Machine Learning Group at National Taiwan University. LIBMF: A Matrix-factorization Library for Recommender Systems. 2015.

[3] Lars Backstrom and Jure Leskovec. Supervised Random Walks:Predicting and Recommending Links in Social Networks. 2011.

[4] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based Features for Supervised Link Prediction. 2011.

[5] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. Temporal Link Prediction using Matrix and Tensor Factorizations. 2010.

[6] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link Prediction using Supervised Learning. 2006.

[7] Mohammad Al Hasan and Mohammed J. Zaki. A Survey of Link Prediction in Social Networks. 2011.

[8] David Liben-Nowell and Jon Kleinberg. The Link Prediction Problem for Social Networks. 2003.

[9] Kevin Mao. Yelp SNA. https://github.com/kevinmao/yelp-sna, 2015.

[10] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. Scalable Proximity Estimation and Link Prediction in Online Social Networks. 2009.

[11] Yelp. Yelp Dataset Challenge. http://www.yelp.com/dataset_challenge, 2015.