Link Prediction in Bipartite Networks

Anirudh Narasimhamurthy
u0941400
University Of Utah

Soumya Smruti Mishra
u0926085
University Of Utah

Abstract

One of the challenges in network analysis is to predict what new connections will be created in the particular graph/network at some point in future. Many real world networks like user-products or user-business relations have a bipartite nature to them and evolve during time. Predicting links that will appear in them is one of the main approaches to understanding them. This was our main point of interest and we decided to understand link prediction problem in bipartite networks in our project. We also see how this can be extended or formulated to be applied to a recommendation problem.

Classification Keywords

link prediction, collaborative filtering, recommendation, bipartite graph,

1 Motivation

The primary motivation for selecting this project was based on two reasons:

1. Firstly we wanted to explore and learn about link prediction and bipartite graphs. From our literature survey and reading papers [2] [6] [7] [8] we realized that link prediction in bipartite graphs was something which was less explored and hence we were interested to learn more about it through our project.

2. Secondly we were also trying to see if we could use any of the Machine Learning techniques which we had learned about in this project and we believed the problem which we chose would probably help us in achieving it.

2 Introduction

Link prediction is an important research problem in dynamic network analysis. Most of the problems which can be modeled as networks today are dynamic in nature i.e they evolve with node and link additions and removals. Typical examples include user - product relations where users are linked to the products they bought.

[9] states that social networks are not static and keep growing with time with addition of new nodes and links. Similarly user-product networks also evolve with time. This deals with link formation in the graphs i.e given nodes in a network the network grows by forming new relationships with existing nodes. We in this project deal with the question of predicting if a link will be formed between user and product/business.

Being able to predict such links accurately can have important implications in different networks including uses in national defense, predicting future credit card frauds and other related problems.Also a lot of the current research from what we understood focuses on link prediction on graphs with one type of node such as predicting links in social networks(Twitter, Facebook, LinkedIn) which is made up of people only. We wanted to explore methods which could be applied for a bipartite graph setting.

In this project we attempted to use our understanding of the concepts we learned in this course in line with different methods we came across the literature and used the following methods for link prediction in bipartite graph:

1. Similarity measures using Jaccard similarity and common neighbors
2. Approach based on completing the matrix entries using SVD
3. A page rank style approach using Random walks.
4. Supervised binary classification method.

3 Formal Problem Description

Our project attempts to predict new links which can potentially be formed in a network in future, given a snapshot of the current network and to evaluate its effectiveness. We are also primarily involved in applying the different metric to bipartite graphs.

4 Definitions

Bipartite graph: A bipartite graph or bigraph is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent to each other.

5 Assumptions

1. One of the primary assumptions that we make is although collaborative filtering recommendation technique might work with the data set, our interest was to learn about link prediction in bipartite graphs and so we model our data set as bipartite graph.

2. Link prediction is our primary goal and recommendation is a side product of this one and we assume the recommendation or result which we state in our later sections might be a possible result although the sparsity and other factors might point otherwise.

3. Also for our data set without taking the location into consideration, we assume the prediction that we make is reasonable one although in practice there could be a lot of other things needed to make such a prediction.

6 Scope

Our main stating is that the approaches used in our project could potentially be applied to a different data set which exists as or can be modeled as bipartite graph and even if the example data set which we have taken might not sufficiently convince that, we wanted to make it clear that our scope lies in the understanding of methods/measures which could be applied for link prediction in bipartite graphs. Results and other observations stated in later sections are our understanding but the bigger picture could or could not be different.

7 Prior Work

- A lot of work has been done on link prediction in general.In [4] they experiment with several similarity metrics including Graph Distance, Preferential attachment, Katz method and others. These could be modified so that they can be applied to bipartite graphs. In our project we experiment with two similarity metrics stated in Introduction section.

- Link prediction in bipartite networks has not been explored extensively. In [5] they describe how similarity metrics could be used as features in bipartite setting and they describe machine learning approaches to solving this problem. Supervised link prediction methods were also suggested.

- Other works in bipartite link prediction include [2, 3]. In [2] they use a kernel based approach to link prediction while [3] uses a different distance metric approach.

- One of the co-authors of our prescribed text book Jure Leskovec has also done research in this area and in [1] they propose a random walk approach to link prediction which makes use of both a distance metric and Page Rank style approach.

- Our approaches used in our project are primarily influenced by the above mentioned works.

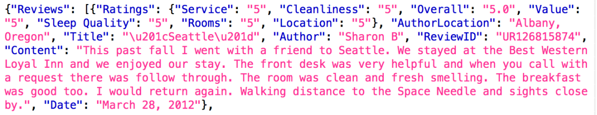
8 Data

8.1 Dataset used for the Project

We used the TripAdvisor dataset for our project. It was obtained from Database and Information Systems Laboratory of University of Illinois, Urbana Champaign from the following url `http://times.cs.uiuc.edu/~wang296/Data/`

8.2 Dataset description

The raw dataset consisted of set of reviews written by different users for different hotels along with the ratings given by the user to different aspects of the hotel. Every review also had a date associated with it. The date range for the reviews in the dataset was from 2002 till 2012. The size of the dataset was 2.06 GB. A snapshot of a the raw review data is shown below for better understanding:



```
{"Reviews": [{"Ratings": {"Service": "5", "Cleanliness": "5", "Overall": "5.0", "Value":
"5", "Sleep Quality": "5", "Rooms": "5", "Location": "5"}, "AuthorLocation": "Albany,
Oregon", "Title": "\u201cSeattle\u201d", "Author": "Sharon B", "ReviewID": "UR126815874",
"Content": "This past fall I went with a friend to Seattle. We stayed at the Best Western
Loyal Inn and we enjoyed our stay. The front desk was very helpful and when you call with
a request there was follow through. The room was clean and fresh smelling. The breakfast
was good too. I would return again. Walking distance to the Space Needle and sights close
by.", "Date": "March 28, 2012"},
```

Figure 1: Raw review data

8.3 Data processing

The most important aspect of our problem is to model the given dataset as a bipartite graph. From the definition mentioned earlier, to model the given data as a bipartite graph, the following processing was required to be done:

- In order to lessen the load on our processing we removed the review text from all the files and then created a cleaned up reviews.json. We adopted this approach with the assumption that we would not be making use of review content to do a prediction. The cleaned up reviews file had the following fields

    1. Ratings
    2. Author name or User name
    3. Author Location
    4. Date of Review
    5. HotelID
    6. Hotel Location

- Using the reviews.json created in the above step, we then created two more files one for users/authors and the other for hotels. The users.json had the following fields:

    1. AuthorID (a unique ID assigned by us)
    2. ReviewCount (aggregation of all the reviews written by this author identified by Name/ID)
    3. Last Review Date(date of the last review written by this author/user)
    4. UserLocation

- The hotels.json file on the other hand had the following fields:

    1. HotelID
    2. ReviewCount (aggregation of all the reviews written for this Hotel identified by ID)
    3. HotelLocation

- At the end of this processing, the count of hotels and users are given below:

Hotels count	11797
Users count	781237

Table 1: Hotels and users count

- These two files users.json and hotels.json would then help us model the bipartite graph as we would have all the users on one side of the graph and we would have all the hotels on the other side of the graph and an edge goes between user and hotel if a user has written a review for that hotel.

- As stated earlier that prediction will be done based on the snapshots of the network or graph created at different times. We used t(training) as Jan 1, 2013 meaning all data before it and t'(test) as Jul 1,2013 meaning all data between t and t' to create train and test data respectively. To get the required graph in terms of nodes and edges,we had to do further processing.
- From the hotels and users file, we picked out the IDs and then assigned a unique ID for each of the HotelID and UserID which is a running sequence so that there is better clarity when we look at the graph data file. Once IDs were created we then created the graph.txt file for both test and train data which will contain all the edges in the current dataset represented by nodes on either side.
- Once we created graph.txt to get statistics about the graph, we used some of the built in methods from SNAP.py package and also referred to code blocks online to get the following details:

Property	Value
Number of nodes	793034
Number of edges	1088791
IsConnected	No
Average Degree of a node	1.372944

Table 2: Statistics on the graph.txt

8.4 Creating examples from input data

We also wanted to experiment with supervised learning methods for link prediction and hence we had to create examples so that it could be used by our classifier to predict the output. During the creation of examples, we did a further filtering to ensure data is not overly skewed and it is at an acceptable level. This idea was obtained from one of the projects of the Stanford Data Mining Class handled by Jure Leskovec. The following filtering was done:

1. Only taking users who have been active in the last six months. To do this we made use of the Last Review Date field which we had included while creating users.json.
2. Only taking hotels which are at a distance 3 or 3 hop distance from current user as candidate edges. To compute the hop distance we made use of methods in SNAP.py package

For creating examples we again made use of graph.txt as the input and also took data snapshots at two different times t and t' to create train and test examples.

The way users review hotels evolves over time and so test and training may have huge differences but we assume that those differences would be small enough that supervised learning would still be effective. We try to see if that is the case. Furthermore we felt it was justified in the sense even in real world scenario this is how data would probably end up being because we would have information only about the previous links.

8.5 Statistics on examples and positive negative stuff

<<<<<<<<<<<<<TO BE COMPLETED
I did not understand what is being done there. But I realize it is important for the supervised learning part. So please kindly fill in your understanding and then tell what those values represent
Also
describe
about
the
matrix.json file
and
edges.txt
which gets created
in this
section
I don't understand
what
they are
doing

>>>>>>>>>>>>>>>>>>>>>>

9 Similarity Measures for Link Prediction in Bipartite Graphs

- Similarity metrics can be used to compute the score of each candidate edge or potential links which might appear in future.

- Metrics used in [4] compare the neighbors of two nodes in the edge. But to make it for bipartite graphs we have to modify it. Because of the property of bipartite graphs nodes on the same side or neighbors of nodes on opposite sides do not intersect. Hence we would want to model them in such a way that we have both sets of nodes to contain the same type of node i.e users/ hotels.

- To do this, we choose two sets S(x) and S(y) for our similarity metrics given nodes(x,y) on opposite sides of a graph. We define S(x) and S(y) in a way which is similar to how we used hop distances for creating examples.

  - S(x) represents nodes which are 2 hops away from x. Effectively this produces nodes on same side of the network as x because traveling two hops would be equivalent to going to x's neighbor and then coming back again to the same side of x.
  - S(y) represents y's neighbors. These represent nodes on opposite side of y, which is nothing but same side of x, because we have modeled the graph as bipartite in this case.

- Both the sets from the description above will comprise of node's on the side of x. This enables us to use them to compute distance metrics. We used two such metrics in our project:

9.1 Jaccard Similarity

Jaccard Similarity is one of the widely and most commonly used similarity metrics. Jaccard similarity for our bipartite case is defined by:
$\frac{|S(x) \cap S(y)|}{|S(x) \cup S(y)|}$

Jaccard similarity has the advantage that it is relatively easier to implement and also it is computationally less expensive than say cosine similarity for the same graph.

9.2 Common Neighbors

Common neighbors is a metric which is commonly used in graph networks where similarity could be measured in terms of the neighbors. For the bipartite case, it is simply defined as :
$|S(x) \cap S(y)|$

9.3 Intuition behind similarity metrics and link prediction

- Computing similarity metrics over these set of nodes is effective for link prediction because S(x) returns nodes which are similar to x since they are 2 hops away and on same side of x.

- In other words users who review hotels that you do are likely to be similar to you. It is also reasonable to further assume if lot of them review the same hotel and there is a large overlap between them and s(x) then you are also likely to review that hotel in future. This in a way helps us to uncover the new links which might appear in the graph in future which is our primary motive.

- As stated in the first, this is an assumption which we make and think is reasonable although it has been pointed out that this isn't so straightforward. But the bigger picture from this is, we learnt how to modify existing similarity metrics to be applied to a bipartite case, which was our primary goals.

10 Random walk approach to Link Prediction

In this section we describe how random walks can be used for link prediction in bipartite graphs.

10.1 Ideas used from the class/course material

We wanted to model our problem to be a multi-class classification problem. Also having gone through different literature and considering the coding and effort involved, we decided to use one vs all decomposition technique as opposed to all vs all technique.

On a lighter note, we performed cross-validation for selecting the decomposition technique by checking out the technique used by the top 10 finisher of this Kaggle problem contest and we probed on ideas on the below methods:

1. Random forest using bagging & one vs all.
2. SVM using SGD & one vs all.
3. Ensembles of Decision trees & one vs all.

10.1.1 Multiclass classification one vs all

In this problem we have five labels (open,OT,NC,NRQ,TL), hence we would have 5 decision trees (or) 5 SVM models if we were to build the multi-class classifier using our existing binary classifiers.For instance, Labels OT and the complement $\overline{OT}$ are decided by one of the models. Similarly, there would 4 other models.

Hence, we would use 5 binary classification methods to achieve multiclass classification for prediction of the status of each post.

The final predicted label corresponding to max value of the H_{final} would be final prediction using one vs all decomposition.

10.1.2 Random forest using bagging & one vs all

We developed upon our class homework and we randomly sampled examples with replacement and created 5 data sets. We created decision trees for each of the 5 data sets as indicated in [**?**]. That is the decision trees had only a subset of the total features available for the children.

After creation of such decision trees, final predicted label for any test example would be based on majority vote from these 5 decision trees. Then, as usual one vs all will be applied with majority vote as well.

10.1.3 SVM using SGD & one vs all

Similar to the above model, we created 5 SVM models for each of the labels. For instance, OT and $\overline{OT}$ would have a SVM model.Each of the models had a weight vector.

The final prediction for any example was the label corresponding to max value of wx. Stochastic Gradient Descent algorithm was used and the regularizer used in objective function in our implementation was C=1. Since we are in the online setting and fitting in all the examples at once wasn't necessarily good, stochastic gradient descent worked well by taking on example at a time and developed the prediction model.

10.1.4 Ensembles of Boosted Decision trees & one vs all

We performed 5-fold cross-validation on training data. Boosting based on decision trees was harder than expected. We need to boost the examples which were misclassified, but our feature values were -1s and 1s. Hence, boosting or penalizing 1 or -1 is not possible respectively. This was not done primarily because of the fact that the complexity of the decision trees would increase as feature space.

We have partially implemented this since this suggestion was given in the intermediate report, but we couldn't eventually complete the suggested idea due to the above reason.

11 Ideas Explored but not used

This section describes some of the ideas which we brainstormed and further wanted to explore. Although both of us couldn't concur with the views and validity of the idea proposed, we thought it would be good to mention it here:

11.1 Combination of SVM and Decision Tree

1. We would know the prediction label for each example from each of the methods - SVM and RF.
2. We could consider the prediction values from the 4 methods as feature vectors.

3. Final prediction label for a example x would correspond to the label with higher wx, where w is the normalized prediction accuracy of that particular model

4. Say accuracy of the methods 1,2,3,4 are a1,a2,a3,a4. Then the w1=a1/(a1+a2+a3+a4)

5. But this model was purely **based on intuition**, we didn't have enough proof to assume so.Hence we discarded implementation of this model.

12 Evaluation of test dataset

We executed each test example in the models above. The final accuracies are tabulated. Refer 3 for the accuracy of each method.

As per paper[**?**], prediction based on both user and post datasets was 73

12.1 Inference about the result

AgeofPost, CBCount - number of code blocks, Reputation at post creation date and BodyLength-length of the post were the most relevant features. Most of the decision trees created for bagging had these 4 features in common. SVM weight vector values for these 4 features were among the top 5 as well. Reasoning about this higher weights would give the intuition that higher the weight more it will contribute to the argmax for one vs all.

13 Conclusion

The features used which were based on the posts, tags and user information did a decent enough job for the prediction model. The baselined features were slightly limited, coming to think of it in broader perspective and some more additional features or other derived features which could have been obtained from the user and post data could have improved the accuracies of our model.

We see that the vowpal wabbit implementation did relatively better than our model but the fact that we were able to implement a basic multi-class classification was satisfying. Also the performance provided by scikit for the random forests was also slightly better because of the internally optimized python code.

Additionally the text sentiment analysis and text comparison in stack overflow database would lead to better results in prediction of whether a stack overflow question will be closed. This is mainly because the content of the question plays a very important role in the diagnosis of whether the question will be closed.

In conclusion we were able to build a multi-class classifier which could predict the status of a question on StackOverflow and the results could be used to then determine if the question should be closed or not.

14 Difficulties faced in implementation

1. First and foremost, we needed to preprocess the data for vowpal wabbit in vw format.

2. We faced too many installation issues for Vowpal wabbit. We didn't have a clear man page for commands to be used unlike scikit.

3. We also needed to preprocess data for custom implementations.

4. Since we implemented bagging(ensemble) of decision trees for class homeworks, - Custom RF (Bagging of Decision tree) One Vs all was relatively easy.

5. Custom SVM One vs All implementation involved creating SVM model objects for each hypothesis similar to Custom RF.

15 Future Work

- We assumed 0 or 1 for real valued features for ease of implementation, which could be improved to include more values to increase accuracy of prediction.

- Due to time crunch, we were not able to increase the number of random sub-samples selected for random forest.

- We could easily find out the link to related user data set, which is not available now. If we had the link, the our prediction accuracies might have been closer to the paper [**?**].

Methods	**Accuracy**
SVM using vowpal wabbit	67.03
RF using scikit	64.58
Custom SVM using SGD one vs all	61.96
Custom RF (Bagging of Decision tree) one vs all	59.19

Table 3: Accuracies for multiclass classification

- The text in the post could be represented as a vector using the tf-idf-weight technique or the Latent Dirichlet Allocation(LDA). We could explore more on this.
- While brainstorming, we had seen combinations of Decision Tree and SVM method called DTSVM in the paper[**?**]. DTSVM is a faster algorithm which produces equivalently good prediction accuracies.

References

[1] Lars Backstrom , Jure Leskovec, Supervised random walks: predicting and recommending links in social networks, Proceedings of the fourth ACM international conference on Web search and data mining, February 09-12, 2011, Hong Kong, China.

[2] J. Kunegis, E. De Luca, and S. Albayrak. The link prediction problem in bipartite networks. In Computational Intelligence for Knowledge-Based Systems Design. 2010.

[3] Y. Yamanishi. Supervised bipartite graph inference. In NIPS, D. Koller, D. Schuurmans, Y.Bengio, and L. Bottou, Eds. MIT Press, 2008, pp. 18411848.

[4] Liben-Nowell, David and Kleinberg, Jon, The link-prediction problem for social networks,Journal of the American Society for Information Science and Technology,http://dx.doi.org/10.1002/asi.20591,2007.

[5] N. Benchettara, R. Kanawati, C. Rouveirol. Supervised Machine Learning applied to Link Prediction in Bipartite Social Networks. In Proceedings of the International Conference on Advances in Social Network Analysis and Mining. IEEE, Los Alamitos, CA, 326-330, 2010

[6] O. Allali, C. Magnien and M. Latapy , ”Link prediction in bipartite graphs using internal links and weighted projection” , NetSciCom , pp.953 -958

[7] Xin Li , Hsinchun Chen, Recommendation as link prediction: a graph kernel-based machine learning approach, Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, June 15-19, 2009, Austin, TX, USA

[8] Zan Huang , Xin Li , Hsinchun Chen, Link prediction approach to collaborative filtering, Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries, June 07-11, 2005, Denver, CO, USA

[9] https://www3.nd.edu/ dial/papers/ICDM12b.pdf