CS 6965 - FINAL PROJECT

LINK PREDICTION IN BIPARTITE NETWORKS

Anirudh Narasimhamurthy(u0941400)
Soumya Smruti Mishra(u0926085)

Inspiration and Objective

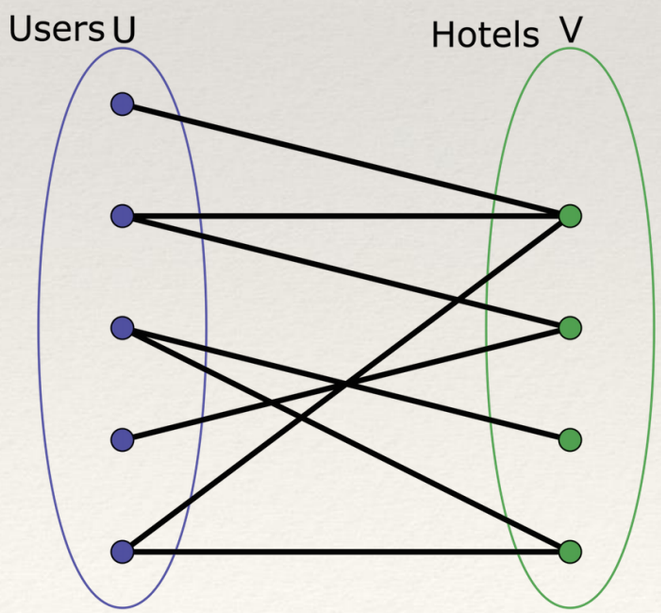
- Predicting when new connections will be created in a particular graph at some point in the future.
- **Why is that important ?** Importance lies in the fact that it has varied applications in defense, banking and even in social networks.
- Most of the current research in link prediction is for prediction on graphs with one type of node.
- **Ex:** Predicting if someone would someone else as friend on Facebook or if X would follow Y on Twitter or if A would connect with B on LinkedIn.
- **Objective:** Try to modify the algorithms used for these type of graphs so that they can be applied to Bipartite Graphs.
- **Definition**:A bipartite graph, also called a bigraph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.

Prior Work

- Extensive work has been done on link prediction in general but not much with respect to bipartite graphs.
- Experiment with node similarity metrics including Graph Distance, Common Neighbors, Jaccard's Coefficient.
- Low-rank matrix approximation.
- The challenge would be to adapt these measures to make it work for bipartite graphs.
- Another challenge is to build a model which combines information about node and edge attributes in a principled way. Supervised learning methods are generally used and supervised random walks are used.
- Bencehttera et al [1] describe how common link prediction measures can be used as features in bipartite setting.

Dataset used

- The dataset which we used for the project is HotelReviews dataset obtained from [2]
- It consists of information about reviews written by the user for a particular hotel along with the ratings.
- So the first step was to model this dataset as a bipartite graph where one side of the network represents users and other side represents the hotel/business.



Primary Objective

- Once we have the given dataset modeled as a bipartite graph, our objective is to do a link prediction.
- More specifically in our data, it would be to **try to predict which hotel or hotels a particular user will review in the future**.
- An edge goes between a user and a hotel if the user has written a review for that hotel.
- We use a combination of factors and on a high level the prediction is based on the similarity metrics and other approximation methods.
- Each of the metrics will give a weight to a candidate edge (which doesn’t exist) in the original graph and then choose those edges which are above a threshold value.

Data Preprocessing


```
{"Reviews": [{"Ratings": {"Service": "4", "Cleanliness": "5", "Overall": "5.0", "Value":
"4", "Sleep Quality": "4", "Rooms": "5", "Location": "5"}, "AuthorLocation": "Boston",
"Title": "\u201cExcellent Hotel & Location\u201d", "Author": "gowharr32", "ReviewID":
"UR126946257", "Content": "We enjoyed the Best Western Pioneer Square. My husband and I
had a room with a king bed and it was clean, quiet, and attractive. Our sons were in a
room with twin beds. Their room was in the corner on the main street and they said it was
a little noisier and the neon light shone in. But later hotels on the trip made them
appreciate this one more. We loved the old wood center staircase. Breakfast was included
and everyone was happy with waffles, toast, cereal, and an egg meal. Location was great.
We could walk to shops and restaurants as well as transportation. Pike Market was a
reasonable walk. We enjoyed the nearby Gold Rush Museum. Very, very happy with our stay.
Staff was helpful and knowledgeable.", "Date": "March 29, 2012"},
```


- Snapshot of the raw data json file having a review for a hotel

- From the given raw file, we created three files namely: users.json, hotels.son and reviews.json.
- users.json will have username, review count and the date of his last review
- hotles.json will have hotelname, hotel_id, review count.
- reviews.json will have details about the user_id, hotel_id, review rating and the date of the review.
- The details in review.json is then used to create the bi-partite graph.Specifically we use the user_id and hotel_id to create the bipartite graph.

Data Preprocessing

- During the data preprocessing we also found that there were certain users who had written more than 400 reviews and it turns out those were generic accounts which did not require a sign-in and so we removed those entries so that our distribution is not skewed.

```
In [49]: h_c.most_common(10)
Out[49]:
[(u'A TripAdvisor Member', 117668),
 (u'lass=', 34856),
 (u'Posted by an Accorhotels.com traveler', 8692),
 (u'Posted by an Easytobook.com traveler', 468),
 (u'Posted by a hotelsgrandparis.com traveler', 178),
 (u'John S', 128),
 (u'David S', 124),
 (u'John C', 122),
 (u'Pawel_EPWR', 122),
 (u'ITA_And_RE_a', 120)]
```

- This filtering was done to ensure we are able to eliminate false positives and provide a recommendation which would be a good reflection of the data and more practical.

- Before we created the graph.txt which is the source for bi-partite we made sure these user_ids are filtered off.

Data Preprocessing


```
{
  "73393": 93,
  "1193354": 1,
  "114079": 48,
  "89379": 72,
  "114075": 1209,
  "228281": 2,
  "550184": 45,
  "1821731": 111,
  "677025": 8,
  "2516107": 24,
  "241678": 78,
  "82443": 176,
  "2327171": 8,
}
```


Sample hotels.json file


```
{
  "Author_id1": {
    'Date': "[u'2011-12-23 00:00:00']",
    'review_count': 2
  }
}
```


Sample users.json file


```
{
  "UR126946257": {
    u'Author': u'gowharr32',
    u'Date': u'2012-03-29 00:00:00',
    u'HotelID': u'72572',
    u'Ratings': {
      u'Business service': u'5.0',
      u'Cleanliness': u'5',
      u'Internet access': u'5.0',
      u'Location': u'5',
      u'Overall': u'5.0',
      u'Rooms': u'5',
      u'Service': u'4',
      u'Sleep Quality': u'4',
      u'Value': u'4',
      u'front desk': u'5.0'
    }
  }
}
```

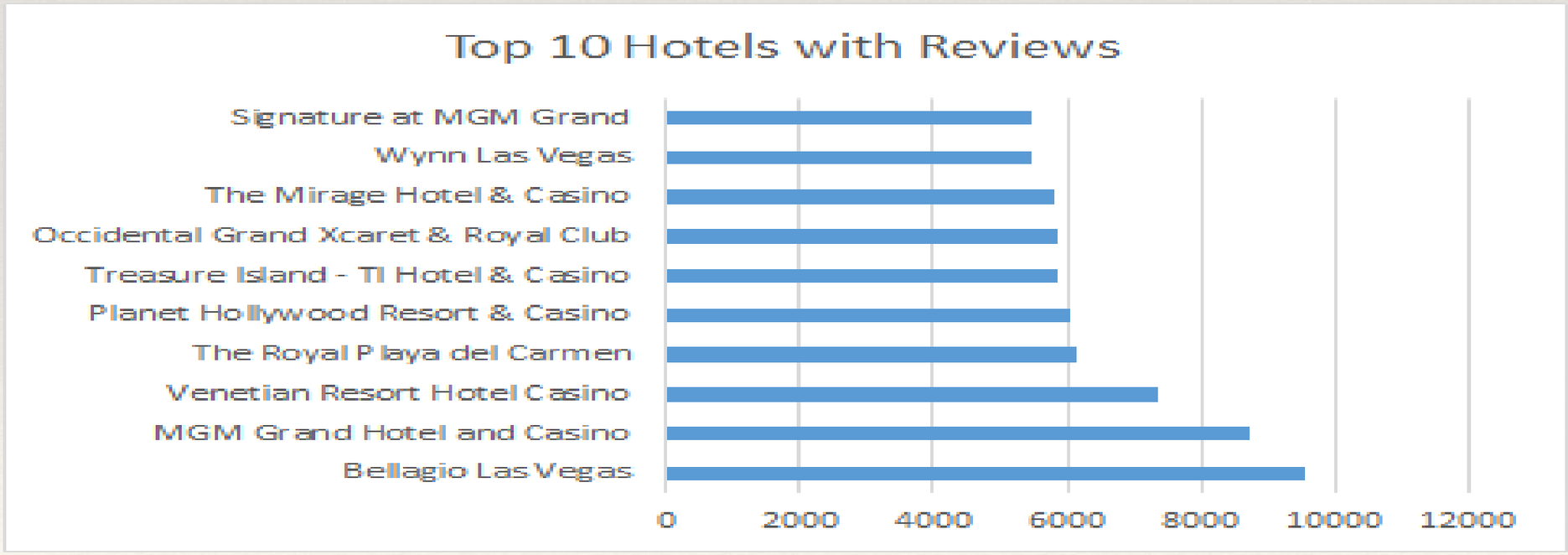

Sample review.json file


```
0 1
2 1
3 4
5 4
6 4
7 4
8 4
9 4
10 4
11 4
12 13
14 13
15 13
16 13
17 13
18 13
19 13
20 21
22 21
23 21
23 21
15 24
25 24
26 27
28 27
29 27
30 27
31 27
32 27
33 27
34 27
35 36
37 36
38 39
40 41
6 41
42 41
```


Sample graph.txt file

Data Analysis

- Details about graph.txt :
- **Number of nodes - 2,99,341**
- **Number of edges - 5,32.456**
- We also found from our data processing that these were the top 10 hotels sorted based on their review count.



Similarity Metrics

- Similarity metrics which were used for link prediction[3] in normal graphs cannot be directly used for bipartite setting because neighbors of nodes on opposite sides of network do not intersect.
- Instead we want both sets of nodes to contain same type of nodes.
- To do this, we chose the approach given in [4]. Two sets S(x)& S(y) was chosen for our similarity metrics given nodes (x; y) on opposite sides of the graph.
- S(x): nodes 2 hops away from x: This produces a set of nodes on the same side of the network as x because traveling distance 2 goes to the other side of the network and back again.
- S(y): y's neighbors: These nodes are on the opposite of the graph of y, so they are on the same side as x.
- Thus both sets only contain nodes on x's side, so we can easily use them to compute distance metrics.

Why the similarity measure works ?

- The intuition is that S(x) returns nodes which are similar to x since they are two hops away and on same side of the network or graph.
- In other words users who review same hotels as you do are similar to you.
- Thus if a lot of users review the same hotel y, i.e S(y) and users who review y has a large overlap with S(x) then you are likely to review that hotel too in the future.

Similarity metrics used

1. Common Neighbors: $|S(x) \cap S(y)|$
2. Jaccard's Coefficient: $\frac{|S(x) \cap S(y)|}{|S(x) \cup S(y)|}$

- These were the similarity metrics which we had used for link prediction in bipartite graphs.
- With this similarity measure every candidate edge would be given a score or weight and the ones above a threshold would be used for predicting the existence of future links.

Low Rank Approximation or SVD

- Another way of viewing link prediction is to say it is a way of predicting new entries in the adjacency matrix of the graph.
- For large matrices like ours, computing a low rank matrix A_k of rank ‘k’ which approximates A is one of the common techniques.
- We used SVD for the approximation and this can be computed efficiently for small ‘k’ even for a large sparse matrix such as an adjacency matrix. (Scipy packages)
- The entries of the low rank approximation A_k is used as scores for candidate edges.
- Bipartite graph has nice properties : a) symmetric b)large number of entries are guaranteed to be zero.
- This allows us to compute SVD of user-hotel matrix instead of the adjacency matrix.
- We are effectively learning k-dimensional feature vectors for each user and each business such that the dot product between them is high if an edge is present.

Unsupervised Random Walk

- We used random walks to produce a score for each candidate edge (u,v).
- Suppose each edge (u, v) belongs to E is assigned a weight w(u,v). Then each edge (u, v) belongs to E can be assigned a transition probability M_(u,v) = w(u, v)/d(u) where

$$d(u) = \sum_{(u,v)\in E} w(u,v)$$

- These transition probabilities tell us or help us in the random traversal of the graph.
- And so when we do this for a long time, we would know which nodes we would tend to visit by this process. We use the stationary distribution p_s
- We used simple heuristically chosen edge weights instead of learning the weights in a supervised way. Intuitively, we want links that are more relevant to have higher weight for the random walk. To capture this, we weighted edges that were created more recently higher.
- Specifically, we gave each edge a weight of w_uv = 1=(c+a_uv) where a_uv is the age of the edge (u, v) and c is a positive constant.

Supervised Binary Classification

- Binary classifier can also be used for link prediction.
- The primary task of the classifier would be to predict, given two nodes, whether a link will occur between them in the future.
- Several considerations : SVM, Logistic Regression, Random Forests etc.
- We also decided to implement xgboost algorithm, which we suspect will give better results than all of the above.
- Trevor Hastie hypothesizes that in terms of model accuracy:
  - Boosting > Random Forest > Bagging > Single Tree
- xgboost follows the principle of gradient boosting. Specifically, xgboost used a more regularized model formalization to control over-fitting, which gives it better performance.

Results and Evaluation

- To predict if an edge will exist between two nodes at a later point in time, we created test and training examples in the given dataset by creating snapshot at time intervals t and t'.
- Train them on 'x' examples and then predict for the test examples whose labels we would already know, so we can verify the correctness and accuracy of the classifier.
- This neatly models the real life scenario where you would only have prior information and need to predict the future values.
- **Issues:** Because of the huge size of the graph, predicting presence or absence of an edge forming between every pair of nodes is computationally infeasible even with sophisticated big data techniques.
- So two levels of filtering were done: 1) User-level where we select only users whose last written review was within six months from current date. 2)Hotel-level filtering, where we only take hotels which are at a distance of 3 from the current user as candidate edges.
- Evaluation Metrics (To be done): Area under curve and user precision at 20*.

References

- [1] 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.

- [2] http://times.cs.uiuc.edu/~wang296/Data/

- [3] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In 12th CIKM,pages 556-559, 2003.