CS224W Project Final Report: Applying Link Prediction to Amazon Product Recommendation*

Shiyu Huang

December 10, 2018

1 Introduction and Problem Definition

Recommendation system has become very common in today's business world and is present in many shapes and forms on online platforms such as Q&A sites and social media. One of the most prominent is perhaps Amazon's product recommendation system where co-purchasing history is used to make future purchase recommendations to users. Needless to say, having a good product recommendation system is crucial for both the sellers and the consumers. On the one hand, sellers increase their potential revenues by increasing awareness of products that may not be popular enough for prominent advertising space. On the other hand, consumers gain a more streamlined and varied shopping experience with more options and a much higher probability of finding the products they need.

Traditionally, product recommendation is often built using some form of content-based systems which examine properties of the items recommended, or using the collaborative filtering approach which is based on similarity measures between users and/or items. In this project, however, we will take a network approach to this problem by focusing only on network/node properties and structure of the network to perform product recommendation, which is modeled as prediction of missing links in the co-purchasing network.

Specifically, we want to achieve the following: given a product p, we want to be able to retrieve a ranked list of products that should be recommended to the buyer who purchased p. In other words, given an incomplete graph $G(V, E)$ where V denotes the set of products and there is an edge between product i and j if they are frequently bought together. Let U be the universal set of all $\frac{|V|(|V|-1)}{2}$ possible edges, then the set of nonexistent edges is $U - E$. Some of the edges in $U - E$ may appear in the future, and the task of link prediction is to find these links.

2 Literature Review

2.1 The Link-Prediction Problem for Social Networks [5]

This paper provides an overview of similarity-based link prediction methods using the co-authorship network by comparing relative effectiveness of network similarity measures. The paper classifies these measures into two main categories: (1) methods based on node neighborhoods, i.e. local methods (e.g. Common Neighbors, Jaccard's Coefficient, Adamic/Adar, etc.) and (2) methods based on the ensemble of all paths between two nodes, i.e. global methods (e.g. Katz, PageRank, SimRank, etc). The paper finds that there is no clear winner among these methods and the performance is dependent on the dataset, but a number of them significantly outperforms the base-line predictors, with some simple measures (such as Common Neighbors) doing particularly well.

2.2 Hierarchical Structure and the Prediction of Missing Links in Networks [2]

The paper first observes that many networks exhibit hierarchical organization, where vertices divide into groups that further subdivide into groups of groups, and so forth. Such structure can be uncovered by fitting the hierarchical model to observed network data using a maximum likelihood with a Monte Carlo sampling algorithm on the space of all possible dendrograms. Combined over a large number of samples, we then derive a likely model of the data.

*Code at: https://drive.google.com/drive/folders/1n-dgN86EzHKR615vrGxSqmabBkOgRsD-?usp=sharing

To apply this method to link prediction, we generate a set of hierarchical random graphs based on the incomplete network, then we look for pair of unconnected nodes with high probability of connection in the hierarchical random graphs - these will be likely candidates for missing links. On the test networks, the hierarchical structure model does better than baseline chance model. Unlike similarity based models which works well for strongly assortative networks, hierarchical models make sense in both assortative and disassortative structures.

2.3 Graph-based Features for Supervised Link Prediction [3]

This paper tackles the IJCNN Social Network Challenge to separate real edges from fake edges in a set of 8960 edges sampled from an anonymized, directed graph depicting a subset of relationships on Flickr. For feature extraction, the model employs a large number of a variety of techniques, including extracting local subgraphs which is relevant to nodes in question, using SVD, kNN, EdgeRank etc as well as traditional similarity measures such as Common Neighbors, Jaccard etc. It then performs repeated classification using the posterior probabilities from Random Forests.

3 Data

3.1 Overview

This project uses the Amazon co-purchase network available from the Stanford Network Analysis Project (SNAP), with ground-truth community defined. It is based on *Customers Who Bought This Item Also Bought* feature of the Amazon website. If a product i is frequently co-purchased with product j, the graph contains an undirected edge from i to j. There are 334863 nodes and 925872 edges in the network originally. Each product category provided by Amazon defines each ground-truth community with top 5000 highest quality communities specified.

3.2 Pre-processing and Train-Test Split

Generally we do not know which links are missing now (but might appear in the future), therefore in order to evaluate various approaches, we will randomly divide the observed edges into two disjoint set: one for training, and one for testing. The training set of edges will simulate the graph at a past point in time, whereas the held-away test set will simulate "future" edges. Before we perform any analysis, we first follow the following steps to pre-process and split our data.

Algorithm 1 Data Pre-processing

Let us denote the original complete graph by $G = (V, E)$.

1. Among the all the nodes V, we sample a random 25% of the nodes. Call this set V_1 and the Graph induced be $G_1 = (V_1, E_1)$. We use only a subset of the original graph to keep computation time manageable.
2. Keep only the largest WCC from G_1. Call it $G_2 = (V_2, E_2)$
3. Remove low-degree nodes from G_2. Here we remove any nodes with degree < 3. This is because nodes with smaller degrees (fewer interactions with other nodes) are arguably less relevant for our link prediction task. Let V_{main} be the set of remaining nodes, and the sub-graph induced by V_{main} be $G_{\text{main}} = (V_{\text{main}}, E_{\text{main}})$
4. We now perform the train-test split. Among all edges in E_{main}, we sample 10% of the edges, call this set $E_{\text{test, }+}$. We hold away this set of edges by removing them from the current graph. Now we are left with the training graph $G_{\text{train}} = (V_{\text{main}}, E_{\text{train}})$ where $E_{\text{train}} = E_{\text{main}} \setminus E_{\text{test, }+}$).

The following table summarizes basic statistics of the graph before and after processing.

	No. of Nodes	No. of Edges
G	334863	925872
G_1	83716	58376
G_2	15228	19535
G_{main}	3573	4883
G_{train}	3573	4395

4 Similarity-Based Methods

Similarity-based methods is the most studied category of link prediction methods. The underlying assumption of similarity-based methods is that two entities are more likely to interact if they are similar. And as such, defining a similarity function $\text{Sim}(x, y)$ that assigns a score for every pair of nodes x and y becomes the key task in these methods.

A large number of heuristics have been developed in the past, including local similarity measures such as Common Neighbors, The Resource Allocation Index, and The Jaccard Index, as well as global approahes such as The Katz Index and Random Walks. In this section, we experiment primarily with different local similarity measures to perform the link prediction task.

4.1 Computation and Evaluation

We will evaluate each measure using **Prec@K**. The following information boxes outlines how we will compute and evaluate each similarity measure.

Algorithm 2 Compute Similarity Measures

for all pair of nodes $u, v \in V_{\text{main}}$ and $u \neq v$ **do**
  if $(u, v) \in E_{\text{train}}$ **then**
    continue
  end if
  if u, v has no common neighbors **then**
    continue
  end if
  $s = \text{Sim}(u, v)$ based on G_{train}
  Store (u, v) and s as a tuple `((u,v), s)` in a list `simScoreList`
end for

Algorithm 3 Evaluate Similarity Measures using Prec@K

1. Sort `simScoreList` in descending order of similarity score
2. Output the top K (u, v) pairs as our predicted edges, call this set of edges $E_{\text{pred},K}$.
3. Now compare $E_{\text{pred},K}$ with the withheld test set $E_{\text{test},+}$, We have
   $\text{Prec@K} = \frac{|E_{\text{pred},K} \cap E_{\text{test},+}|}{K}$

The next section outlines specific similarity measures used in our experiment. Results are summarized in section 7.2.

4.2 Local Similarity Measures

- **Common Neighbors (CN)**: Common Neighbors captures the idea that, the more common neighbors two nodes share currently, the more likely a link will form in between them in the future.

$$\text{Sim}(u, v)_{CN} = |\Gamma(u) \cap \Gamma(v)|$$

- **Jaccard Index (JA)**: Compared to Common Neighbors, Jaccard Index futher takes into account the number of neighbors each of the nodes already has, by computing the fraction of common neighbors between two nodes among all neighbors the two nodes have.

$$\text{Sim}(u, v)_{JA} = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

- **Preferential Attachment (PA)**: The "rich gets richer" intuition is that the larger the current neighborhood of the two nodes, the more likely the future connection.

$$\text{Sim}(u,v)_{PA} = |\Gamma(u)||\Gamma(v)|$$

- **Adamic/Adar (AA)**: Adamic/Adar also considers common neighbors between two nodes, but gives more weight to common neighbors with smaller degree.

$$\text{Sim}(u,v)_{AA} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log(|\Gamma(z)|)}$$

- **Resource Allocation (RA)**: RA index is similar to AA, but is motivated by the process of resource allocation. There is a higher penalty for high degree common neighbors in RA than in AA.

$$\text{Sim}(u,v)_{RA} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(z)|}$$

4.3 Enhanced Local Similarity using Community Information

One idea we came up with to improve on the local similarity model is to incorporate additional information beyond just node-node similarity. For example, community information, extracted by either some ground truth or by community detection algorithms such as Louvain method, can be used to supplement similarity measures to improve link prediction accuracy.

Suppose we are computing the similarity score $\text{Sim}(u,v)$ between node u and v based on common neighbors. Suppose common neighbor a is not in the same community as u and v or shares community with only one of the node, whereas common neighbor b share the same community with both u and v then possibly b can be given higher weight.

- **Modified Common Neighbors (CN^*)**: We modified Common Neighbors $\text{Sim}(u,v)_{CN}$ as follows. Let $C(x)$ denote the communities containing node x. We start with $CN(u,v)$ and for every community u, v shared, we add α points. Then for each neighbor i shared by u and v, we add an additional β point for every community that a, b and i share.

$$\text{Sim}(u,v)_{CN^*} = |\Gamma(u) \cap \Gamma(v)| + \alpha|C(u) \cap C(v)| + \beta \sum_{i \in \Gamma(u) \cap \Gamma(v)} |C(i) \cap C(u) \cap C(v)|$$

- **Modified Resource Allocation (RA^*)**: Similarly, we can also modify $S(u,v)_{RA}$ to give additional weight to common neighbors that are in the same community as u and v.

$$\text{Sim}(u,v)_{RA^*} = \sum_{i \in \Gamma(u) \cap \Gamma(v)} \frac{1 + \alpha|C(i) \cap C(u) \cap C(v)|}{\deg(i)}$$

Results are summarized in section 7.2. Top 5000 Ground truth communities is used.

5 Matrix Completion

The global topological information can be exploited through the adjacency matrix, where the nonzero entries denote the connections between vertices, while missing links and non-existing links are both denoted by zero entries.

In this context, link prediction can be framed as a matrix completion problem. Specifically, suppose we have an observed network represented by adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ which is a subset of the original network $\mathbf{G}^*$. In the set-up of this project, $\mathbf{G}^*$ is G_{main} and $\mathbf{A}$ is extracted from G_{train}, E_{train} is the set of edges we observe in $\mathbf{A}$ and E_{test} is the hidden set of edges we want to recover. Our goal is to recover a network $\mathbf{G}$ that is sufficiently close to $\mathbf{G}^*$ based on observed $\mathbf{A}$.

Let matrix $\mathbf{X}^* \in \mathbb{R}^{n \times n}$ be the backbone structure that describes the evolution of the network, and $\mathbf{X}$ be the subset of $\mathbf{X}^*$ containing only the new links. In other words, if we take $\mathbf{X}^*$ and compare with $\mathbf{A}$ and change all entries in $\mathbf{X}^*$ corresponding to non-zero values in $\mathbf{A}$ to 0, we obtain $\mathbf{X}$. We can think of $\mathbf{A}$ as a noisy observation of $\mathbf{X}^*$, and $\mathbf{X}^*$ can be obtained from $\mathbf{A}$ by subtracting an error matrix $\mathbf{E} \in \mathbb{R}^{n \times n}$. In other words, we have the following relations.

$$\mathbf{A} = \mathbf{X}^* + \mathbf{E}$$

$$\mathbf{G} = \mathbf{X} + \mathbf{A}$$

Principle Component Analysis (PCA) is a tool we can use to obtain $\mathbf{X}^*$ and $\mathbf{E}$ simultaneously by converting $\mathbf{A}$ into a set of linearly uncorrelated set of principle components. Here we use the Robust PCA [1] to obtain $\mathbf{X}^*$ and $\mathbf{E}$ because classical PCA requires $\mathbf{A}$ and $\mathbf{E}$ to have low rank property, which is hard to satisfy. Once we have $\mathbf{X}^*$, we compare it against $\mathbf{A}$ where a non-zero entry s in $\mathbf{X}^*$ with corresponding zero entry $\mathbf{A}$ denotes a predicted edge with likelihood s.

Algorithm 4 Matrix Completion

1. Compute the adjacency matrix $\mathbf{A}$ from graph G_{train}
2. Compute $\mathbf{A} = \mathbf{X}^* + \mathbf{E}$ where $\mathbf{X}^*$ is low rank, and $\mathbf{E}$ is sparse using Robust PCA [4]
3. Symmetrize $\mathbf{X}^* = \mathbf{X}^* + \mathbf{X}^{*T}$
4. Output prediction.
   for all entries (i, j) in $\mathbf{X}^*$ **do**
   if $\mathbf{A}(i, j) = 0$ and $\mathbf{X}^*(i, j) \neq 0$ **then**
   `score` $= \mathbf{X}^*(i, j)$
   `u` is the node ID corresponding to i and `v` is the node ID corresponding to j
   Store `(u,v)` and `score` as a tuple `((u,v), score)` in a list `scoreList`
   end if
   end for
5. As before, evaluate `scoreList` using Prec@K

See section 7.3 for experimental results.

6 Supervised Binary Classification

So far we have experimented with unsupervised methods. Another class of approach is to train a supervised classifier on the graph. We again first perform train-test split on the edges in the network to obtain $E_{\text{train, }+}$ and $E_{\text{test, }+}$. These existing edges will be positive examples. We then append a similarly sized set of negative examples (pairs of nodes not connected by edges) to the training set, and obtain $E_{\text{train}} = E_{\text{train, }+} \cup E_{\text{train, }-}$. All of the rest of the negative edges will be part of the test set along with $E_{\text{test, }+}$.

And for each pair of nodes in the network, we extract a feature vector. Here we used some standard graph features and some similarity measures, including source node degree, destination node degree, Common Neighbors (CN), Jaccard Similarity (JA) and Resouce Allocation (RA) to form a feature vector of length 5. We then train an SVM with linear kernel to predict on the test set of node pairs, with probability indicating the confidence of edge existence. We then rank these predictions like before and compute Prec@K.

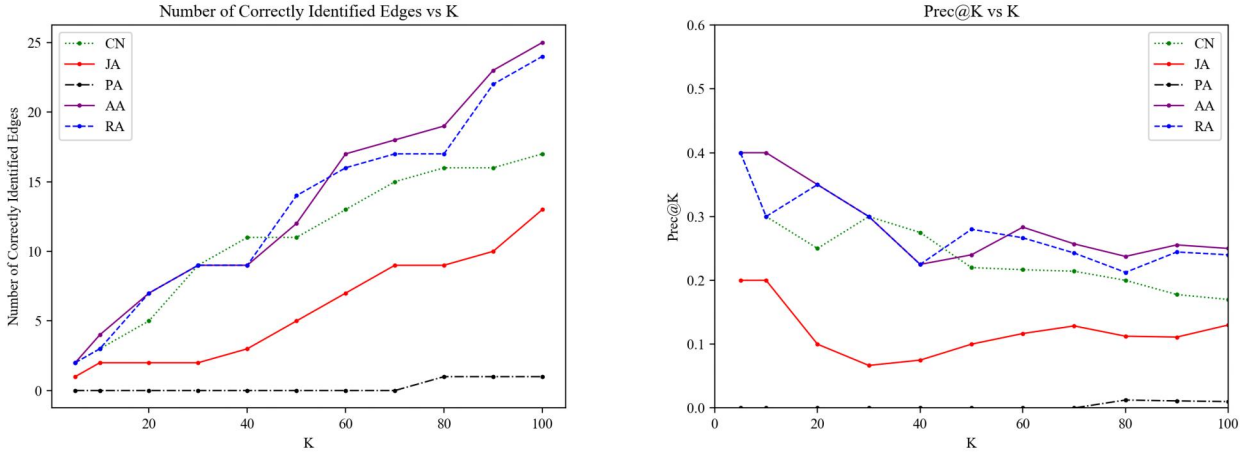
See section 7.4 for experimental results.

7 Experimental Results

7.1 Local Similarity Measures

	K=5	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80	K=90	K=100
CN	0.4	0.3	0.25	0.3	0.275	0.22	0.2167	0.214	0.2	0.178	0.17
JA	0.2	0.2	0.1	0.067	0.075	0.1	0.1167	0.129	0.1125	0.111	0.13
PA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0125	0.011	0.01
AA	0.4	0.4	0.35	0.3	0.225	0.24	0.283	0.257	0.2375	0.256	0.25
RA	0.4	0.3	0.35	0.3	0.225	0.28	0.267	0.243	0.2125	0.244	0.24

Table 1: Prec@K for local similarity measures



(a) Number of Correctly Identified Edges vs K

(b) Precision @ K vs K

Figure 1: Local Similarity Measure Results

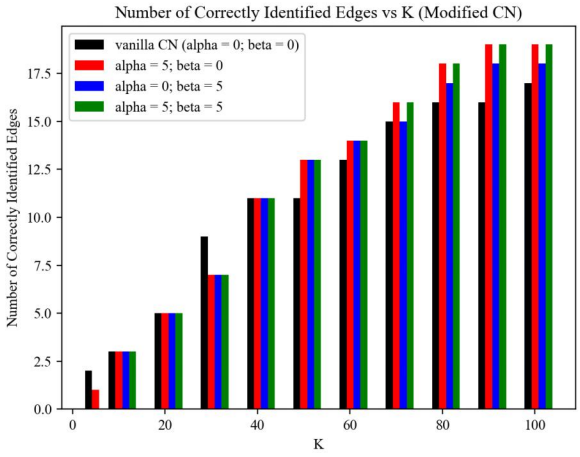
7.2 Enhanced Similarity Measure

	K = 5	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80	K=90	K=100
CN	**2**	**3**	**5**	**9**	**11**	11	13	15	16	16	17
$CN^*(\alpha = 5, \beta = 0)$	1	3	5	7	11	**13**	**14**	**16**	**18**	**19**	**19**
$CN^*(\alpha = 0, \beta = 5)$	0	3	5	7	11	13	14	15	17	18	18
$CN^*(\alpha = 5, \beta = 5)$	0	3	5	7	11	13	14	16	18	19	19

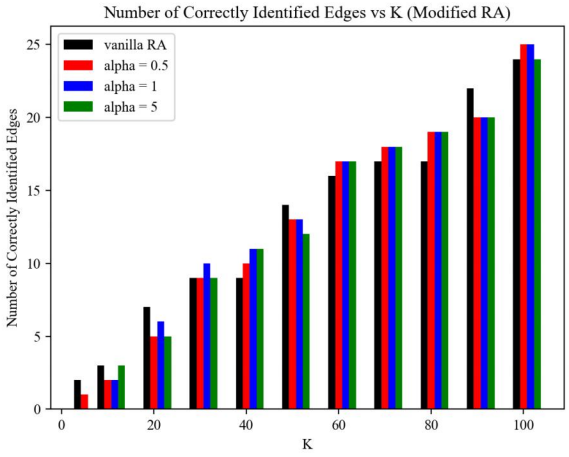
Table 2: Number of correctly preidcted links for CN^*

	K=5	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80	K=90	K=100
CN	**2**	**3**	**7**	9	9	**14**	16	17	17	22	24
RA^* (alpha = 0.5)	1	2	5	9	10	13	**17**	**18**	**19**	20	**25**
RA^* (alpha = 1)	0	2	6	**10**	11	13	17	18	19	20	25
RA^* (alpha = 5)	0	3	5	9	**11**	12	17	18	19	20	24

Table 3: Number of correctly preidcted links for RA^*



(a) Enhanced CN



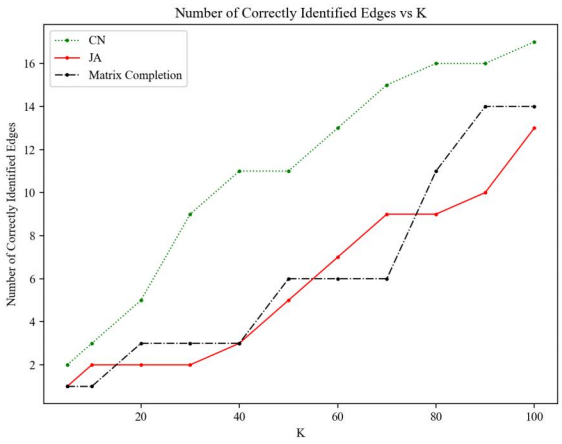
(b) Enhanced RA

Figure 2: Enhanced Similarity Results

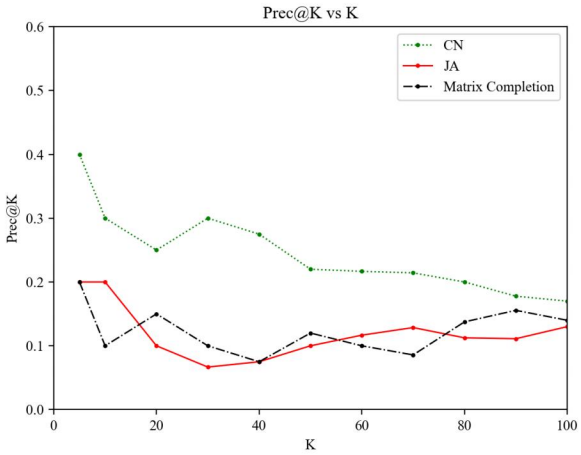
7.3 Matrix Completion

	K=5	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80	K=90	K=100
Matrix Completion	**1**	1	**3**	**3**	**3**	**6**	6	6	**11**	**14**	**14**
JA	1	**2**	2	2	3	5	**7**	**9**	9	10	13

Table 4: Number of correctly predicted links for Matrix Completion



(a) Number of Correctly Identified Edges vs K



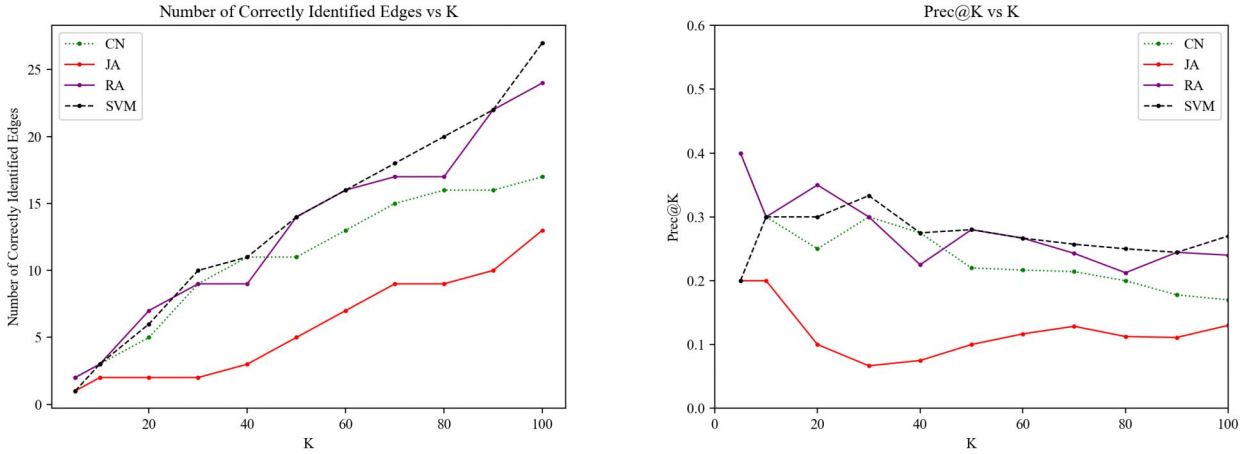
(b) Prec@K vs K

Figure 3: Matrix Completion Results

7.4 Supervised Classification

	K=5	K=10	K=20	K=30	K=40	K=50	K=60	K=70	K=80	K=90	K=100
SVM	1	**3**	6	**10**	**11**	**14**	**16**	**18**	**20**	**22**	**27**
JA	1	2	2	2	3	5	7	9	9	10	13
CN	2	3	5	9	11	11	13	15	16	16	17
RA	**2**	3	**7**	9	9	14	16	17	17	22	24

Table 5: Number of correctly predicted links for SVM



(a) Number of Correctly Identified Edges vs K

(b) Prec@K vs K

Figure 4: SVM Results

7.5 Analysis and Discussion

- Among simple similarity measures, common neighbor based measures Adamic/Adar (AA) and Resource Allocation (RA) performed well even though they are conceptually simple. Put in the context of Amazon product network, it makes sense that we should give higher weight to more "rare" common neighbors because if a third product which is seldom purchased together with other product is a common neighbor between two products, then there is a high chance that these two products are very relevant.

- Despite our high hopes, our "enhanced" local measures did not improve performance very noticeably beyond the base similarity measures. This may have to do with the fact that we are only using the top 5000 communities - if we are able to obtain more extensive and higher-quality community information, these measures may perform better.

- Despite being relatively quick to compute and based on completely different concepts, matrix completion via PCA performed similarly to local similarity benchmarks, which proves again that there are many different frameworks to approach link prediction. It would also be interesting to compare and contrast similarity measures against matrix completion on networks of different level of densities.

- SVM performed well against the best-performing basic similarity measures, consistently upper-bounding prediction precision of the base similarity measures used as features. This makes intuitive sense as all of these similarity numbers are taken into account when making predictions using SVM. We should also note that the SVM is trained with the most basic linear kernel using only 5 features. Expanding feature sets or using kernels with higher degree of flexibility might improve performance.

References

[1] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[2] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008.

[3] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1237–1244. IEEE, 2011.

[4] Robust PCA Implementation. https://github.com/dganguli/robust-pca.

[5] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58:1019–1031, 2007.