# CS224W Project Final Report: Using community detection and link prediction to improve Amazon recommendations

**Amit Garg**
Stanford University
amit93@stanford.edu

**Senthilnathan Viswanathan**
Stanford University
senthil@cs.stanford.edu

**Shloka Desai**
Stanford University
shloka@stanford.edu

# 1 Introduction

As the largest online retailer, Amazon has a huge product base and community of reviewers. A lot of work has been done to determine reviewer communities [6], examine the user reviews of products to model opinion evaluation [7], study how information cascades contribute to opinion formation [6] and study product communities to improve recommendation engines. In this paper, we explore how we can use community detection on Amazon user and item graphs to improve link prediction between reviewers and products. We present a basic recommendation engine that takes in the dataset and outputs product recommendations for each user.

The rest of the paper is structured as follows. We first present prior work, followed by a description of the algorithms we used and modified for community detection and link prediction. We then present results, challenges we encountered, and possible future work.

# 2 Literature Review

In this section we will discuss papers that each address one of the subproblems of the problem we are trying to solve, namely community detection and link prediction.

### 2.1 Community Detection

In [6], the authors build a weighted bipartite graph consisting of items and reviewers using SNAP's Amazon product reviews dataset to analyze the different clustering algorithms and ascertain if the clusters were characterized by individual traits. SNAP's Clauset-Newman-Moore community detection - increases the modularity through hierarchical agglomeration - was found to be the best.

The community detection algorithm we used for this project was **Infomap's two-level** community detection algorithm. This algorithm is based on the map equation, which is a flow based method that operates on the dynamics of the network, as described in [10]. The algorithm works as follows. Initially, each node is assigned to a module. Then each node is moved to a neighboring module that would lead to the maximum decrease of the map equation. After each iteration each module from the previous iteration is represented as a single node in the next one, effectively building up modules from submodules. Note that the nodes are moved into neighboring modules in random sequential order in each iteration. A problem with this algorithm is that once a node is assigned to a module it is forced to stay within the module for the rest of the iterations. This could be ineffective in the case where a node move is optimal in an earlier iteration but suboptimal in the later one. To fix this the algorithm allows for submodule and node movements between modules.

### 2.2 Link Prediction

In paper [3], the authors aim to study the *link prediction problem*: given a snapshot of a social network, it seeks to infer which new interactions among its members are likely to occur in the near future. The paper surveys an array of methods for link prediction: methods based on node neighborhoods like looking at common neighbors, computing Jaccard's coefficient along with Adamic/Adar and preferential attachment in growth networks; methods based on the ensemble of all paths like Katz measure, Hitting time, Commute time, PageRank, SimRank; and higher level approaches which can be used in conjunction with any of methods in the first two categories like low-rank approximation and Unseen bigrams.

The authors found that a number of methods significantly outperform the random predictor, suggesting that there is indeed useful information contained in the network topology. The Katz measure based on clustering and low-rank approximation perform consistently well, along with some of the simpler measures including common neighbors and the Adamic/Adar measure.

Though the predictors we have discussed in [3] perform reasonably well, even the best methods

are correct on only about 16% of its predictions and there is clearly much room for improvement in performance on this task by taking better advantage of the information in the training data. We can also improve the efficiency of the proximity-based methods on very large networks by using faster algorithms for approximating the distribution of node to node distances.

# 3 Algorithm Description

We obtained the Amazon dataset from Julian McAuley, UCSD [12, 13]. The data was separated category wise and it contained meta data (information about products) and reviews.

For the milestone, we were using two years worth of data (2011, 2012 in the case of Amazon Instant Video) to predict which products users will buy in the entirety of the next year (2013), but the window we were actually predicting for was much smaller than that, say January 1st to January 15th. This essentially meant that the users were receiving the same recommendations in July as they would have in January, just based on what they bought in December 2012, regardless of what they bought from January to July. So, for the final, to fix this we changed our algorithm to be temporal. We used a rolling window of $x$ days throughout the prediction year. At any point $t$, we predicted the products that will be bought in the next $x$ weeks, loaded in the information about the products that are actually bought during that time period, and rerun our algorithm to predict what the user will buy in the $x$ weeks following $t + x$. So, we repeated the following procedure for weeks 1 to 52, incrementing by steps of $x$ weeks.

### 3.1 Parsing

The first part of parsing involved generating an items graph by going through the meta data, as this allowed us to add items and edges between items if they were bought together. Next, we looked through the reviews dataset, since this provided information about the users and what items they had reviewed. We then added two edges between nodes in the users graph if both the users had bought an item in common and both of them had given a rating of 3. Using the reviews, we were also able to determine which products each user reviewed, and hence made it possible to connect the users and items graphs, as a third graph.

### 3.2 Obtaining Clusters

After trying several clustering algorithms, we found that Infomap gave the best communities for both items and users, since almost every node was connected to every other node in the cluster. Since Infomap only provided the nodes in the clusters, we had to map these nodes with the original graph to obtain the subgraph corresponding to the nodes.

### 3.3 Computing Centrality Measures

Though our objective was to determine the centrality of the nodes on a per cluster basis, it was not possible since the clusters obtained from Infomap were too tightly connected and hence, we had to resort to obtaining the centrality measures by considering the entire graph. We used PageRank and Eigen Value Centrality on the items and users graphs separately as these were the only values that were able to clearly differentiate the importance of each node in the graph. We also found the distance between all the nodes, as this would allow us to use a weighted system for link prediction. We used both **SNAP** and **NetworkX**, as this provides us with larger number of centrality measures to use from.

### 3.4 Link Prediction

Using a weighted measure for the PageRank and Eigen Value Centrality along with the distance between nodes (closer nodes contributing more towards recommendations), we obtained recommendations for each user.

### 3.5 Analysis

Since, we had a temporal dataset, we were able to determine the percentage of correct predictions with the ground truth, on a per cluster basis as well as for the entire graph. For each iteration of the rolling window algorithm we compared our predictions for the window with the items bought during that time period.

# 4 Results

For experimentation purposes, we used smaller snapshots of three datasets, all of Amazon reviews. They are

1. **Amazon Instant Video**: We used the reviews from years **2011** and **2012**, to make predictions for the year **2013**. The dataset used to generate prediction has 41,105 users and 3,499,887 edges between them. Edges between two users exist if both users have reviewed at least one item in common and both of them had given a rating of 3. We used a high threshold i.e. a rating of 3, as this would allow us to prune away unnecessary edges and help improve the prediction algorithm. There

were 41,105 items present in the graph and since this dataset has no co-purchasing information, no edges exist between the items. The large number of items is due to the absence of time information in the co-purchasing data, and hence we had to take all items purchased from 1996 to 2014. We obtained 400 clusters with Infomap , with the largest cluster containing 1,758 users and the $100^{\text{th}}$ largest cluster containing only 158 users. No clusters were obtained for items due to the absence of edges.

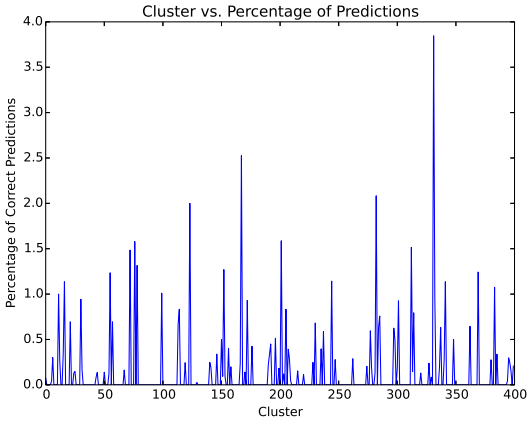


Figure 1: Results for the Amazon Instant Video

The graph above shows accuracy of predictions per cluster. Clusters are arranged by size in descending order. The results obtained were **0.1483%** accurate for 10 predictions per user. In comparison a random sampling on all the items returns an accuracy of **0.0012%**. That's about a **125x improvement** over the baseline, though in absolute the score is quite less. We further discuss some ideas we thought could help improve these results.

2. **Cell Phones and Accessories**: This is the second dataset that we're exploring. We use the years **2008** and **2009** for reviews and predict on **2010**. This dataset had 758,070 edges between 65,207 users. The Amazon Instant Video dataset had about 85 times more edges than nodes in the users graph compared to just 12 times in the Cell Phones and Accessories dataset, which caused our prediction accuracy in the latter to drop to one-tenth of the former. There are 346,793 items in the graph and just 110,674 edges between them, which has led to a large number of isolated nodes. This caused the Infomap algorithm to discard these isolated nodes and hence, made prediction of links difficult. We reduced the number of clusters for both items and users to 400, and in both graphs the largest cluster contained about 10 times the number of nodes as the smallest cluster.

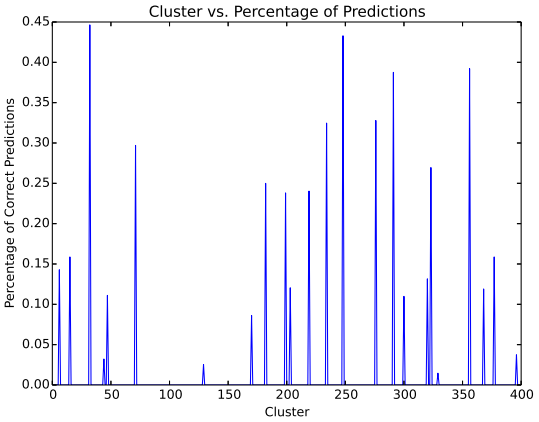


Figure 2: Results for Cell Phones and Accessories

The graph above shows accuracy of predictions per cluster. Clusters are arranged by size in descending order. The results obtained were **0.0148%** accurate for 10 predictions per user. In comparison a random sampling on all the items returns an accuracy of **0.0**, just because of too many items to pick from. Interestingly, when we tried to augment the model using item-item co purchasing data, the accuracy fell to **0.0138%**.

3. **Books**: This is the third dataset that we're exploring. We use the years **1998** and **1999** for reviews and predict on **2000**. Our largest dataset contained 2,370,585 items and just 549,474 edges between them. Despite the large number of items, we just had 109,054 users and 825,482 connections between them. The presence of such a large dataset made both computation and prediction difficult.

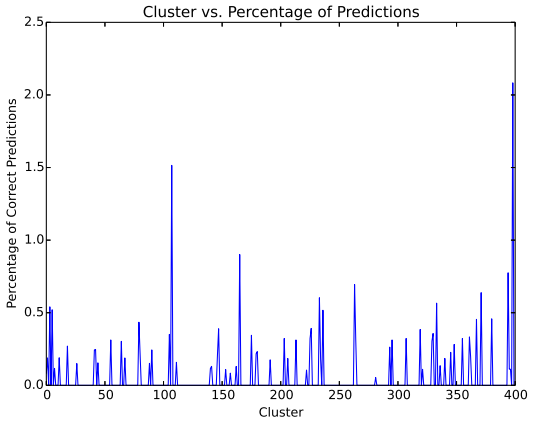


Figure 3: Results for Books

The graph above shows accuracy of predictions per cluster. Clusters are arranged by

size in descending order. The results obtained were **0.0838%** accurate for 10 predictions per user. In comparison a random sampling on all the items returns an accuracy of **0.0**, just because of too many items to pick from. Again, when we tried to augment the model using item-item co purchasing data, the accuracy fell to **0.0824%**.

We believe the reason behind "fall in accuracies upon inclusion of item-item co purchasing data", is the **absence of timestamps**. Since, there is no time information, the co purchasing network serves as an average over a decade, and situations change drastically over such a long time span. For instance, while buying a smart phone, users may be interested in buying a headset. Depending on the year, favored brands and products would change.

# 5 Challenges

While working with the datasets, we came across multiple problems. Here we list some of the key challenges we faced:

1. **Noisy data**: Our goal is to predict the consumers' next purchases, based on the reviews he/she has given. Often, the **items bought are not reviewed**, i.e., there would be many instances in both the learning and testing phases, where a particular user bought an item but didn't review.

   This skews the learning, since **community detection** (and other graph evaluation methods) is heavily dependent on edges between users - which in turn depend on if the users have reviewed at least one common product. Ideally, we would have wanted the edges based on co-purchase of items. Similarly while testing, our algorithm could predict many edges that are not captured by the reviews.

2. **Dense communities**: Once we have the communities, there is very little difference among the users. The **communities are usually completely connected**, i.e., PageRank, Degrees, Betweenness, EigenCentrality and other centrality measures have mostly **identical values**. This was a huge problem, since our approach was to score (and rank) consumers within a community and then use those scores as a proxy for the scores of the products they've bought.

   We got around this problem by calculating the centrality **measures on the entire graph** rather than the community. However, this puts a restriction on what algorithms we can employ, especially since we want to iterate fast, for instance working with **betweenness** gets impractical. Further, **NetworkX** doesn't have pre-compiled C libraries, which makes it quite slow on larger graphs. In fact, some of the simple link prediction algorithms that we tried to work with - Jaccard similarity, negated shortest path - required about a day for complete execution.

3. **Small Variation in purchases**: The communities returned by the algorithm also exhibit a very small set of products that any one in the community bought. We observed that mostly everyone in a community shares all the products, and not just one (which was our criterion to connect). This disrupts the prediction process, where the user has bought most of the top N products we recommend. In fact, in many cases we observed that the community had collectively bought exactly the same products, and so we were unable to make any predictions. As a result, we had to output products with much lower scores.

   Our hypothesis was that users belonging to the same community, in general had a small time window where they all purchased the product. We tried to validate this hypothesis next, but found it to be wrong. We compared our results to the next 3 months of purchase rather than a full year. Further, we tried multiple **Rolling windows** - 1 week, 2 weeks, 4 weeks and 8 weeks. However, none of them were able to give a score better than those over the full year. For this, we made $m$ predictions per window, and $m * n$ predictions for the entire year and compared the results - $n$ denotes the number of rolling windows in the year. A reason for not getting better results was because the predictions were now compared against a smaller window, thereby reducing the accuracy drastically.

   We also tried to pool the results obtained over the entire set of rolling windows and then look at the review data for the entire year, but that too was unsuccessful.

4. **Sparse co-purchasing data**: Another key challenge that we haven't been to tackle is the sparsity of data in the co purchasing network. Most pairs of items (as expected) are seldom bought together. For instance, we wouldn't often see a smartphone purchase end up with another purchase of exactly same earphones. While many users could prefer to buy earphones, they often have personal tastes, and exact matching is unable to capture this.

5. **Lack of computation power**: One of the

major issues we faced was that we were unable to to execute all the steps of our algorithm quickly. So, if we changed the first part of the algorithm, it could take up to several hours to finish execution. It also made it difficult to work on much larger datasets of Books, which could have improved our link prediction accuracy. Girvan-Newman community detection, Jaccard's coefficient and Adamic-Adar for link prediction were some of the algorithms that could not be implemented due to the lack of processing power.

Here is a short list of some things we tried that didn't respond well:

- Using **Clauset-Newman-Moore** and **Girvan-Newman** community detection algorithms besides the one available in the **Infomap** module. However, the results were much worse for the first with many nodes ending up as singletons, and clusters being smaller (we wanted the communities to be larger to allow for a larger selection of products to recommend from). Due to time constraints and a large graph (many edges), we could not get the second algorithm to finish even with a couple of hours, and decided to not pursue it.
- **Item-item collaboration** was another misfire. It only deteriorated the accuracies. We believe the large time difference between the training and testing snapshots to be the cause.
- **Weighted edges** also didn't contribute towards improving the final accuracies. While the communities and various scores were different, the predictions were very similar. Since **SNAP** doesn't support weighted PageRank, we employed **NetworkX** here.
- **K-core** also wasn't able to add to the information we already had. By itself, it would only return one community, and employing it on the communities obtained from **Infomap** either resulted in empty or the complete community as the output (depending on values of K).
- **Jaccard's Coefficient** and **Adamic/Adar Index** for link prediction. Because of how large our datasets were, these algorithms did not come back in an reasonable amount of time (more than 24 hours) even with some of our smaller datasets (300,000 nodes) to allow us to work with them.
- **Rolling window for link prediction** was another failed experiment. For some weeks, our first few predictions were good but overall we were not able to match the scores we obtained by predicting over the entire year. One of the reasons behind this could be seasonal shifts, for instance **the New year** period where there are more sales. Since we make predictions over all the weeks irrespective of the purchasing history of that time of the year, we make just as many predictions for months that have significantly lower purchase history, thereby decreasing our scores substantially.

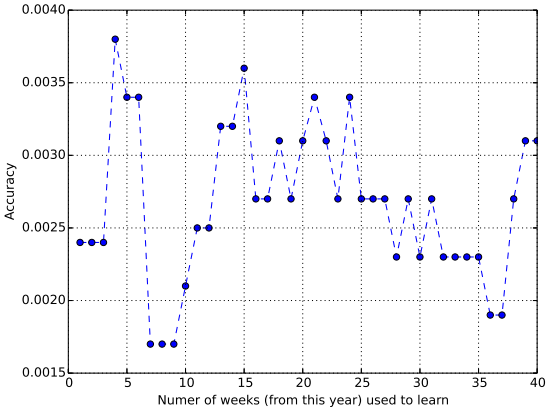


Figure 4: 1 week window on Amazon Instant Video

# 6 Conclusion

In this project we explored how community detection on Amazon user and item graphs could be used to improve link prediction between reviewers and products. **Our algorithm is more than 100x better than randomly predicting** over all the items for smaller datasets (like Amazon Instant Video). For bigger datasets like Books and Cell Phones and Accessories, the accuracy of the random predictor is 0, whereas our algorithm was able to make some meaningful predictions.

# 7 Future Steps

Features available in the items and users dataset can be leveraged further to get better clustering of nodes. This is particularly important in the items dataset of Amazon Instant Videos due to the absence of edges between items. Attributes such as brand, also bought and title can be used to link similar nodes together. Keywords obtained from the description of items and user reviews can be used to cluster items and users respectively. With the availability of time stamped user reviews,

machine learning algorithms could also be implemented to get better link prediction.

Further, time complexity issues for algorithms like Jaccard's coefficient can be somewhat tackled by using platforms like Hadoop MR or Apache Spark.

# 8 References


[1] Newman, Mark EJ, and Michelle Girvan. "Finding and evaluating community structure in networks." Physical review E 69.2 (2004): 026113.

[2] Zhang, Jun, Mark S. Ackerman, and Lada Adamic. "Expertise networks in online communities: structure and algorithms." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.

[3] D. Liben-Nowell, J. Kleinberg. 2003. The Link Prediction Problem for Social Networks. CIKM, pp. 556–559

[4] A. Ceballos, M. Chang, J.Lee, 2014. Using Amazon Product Review Models to Characterize Amazon Reviewer Communities.

[5] Page, L., Brin, S., Motwani, R. and Winograd., T. The Pagerank Citation Ranking: Bringing Order to the Web, Stanford Digital Library Technologies Project, 1998

[6] Im, D., Rhodes, N. Investigating Information Cascades in Amazon Product Reviews (Final Project Report).

[7] C. Danescu Niculescu Mizil, G. Kossinets, J. Kleinberg, L. Lee. How opinions are received by online communities: A case study on Amazon.com helpfulness votes. In Proc. ACM WWW, 2009

[8] S. Song, J. Zhao. Survey of Graph Clustering Algorithms Using Amazon Reviews (Final Project Report).

[9] Fortunato, Santo. Community detection in graphs. Physics Reports, 486(35):75 – 174, 2010.

[10] Rosvall, Martin, and Carl T. Bergstrom. "Maps of random walks on complex networks reveal community structure." Proceedings of the National Academy of Sciences 105.4 (2008): 1118-1123.

[11] McAuley, Julian, et al. "Image-based recommendations on styles and substitutes." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

[12] McAuley, Julian, Rahul Pandey, and Jure Leskovec. "Inferring networks of substitutable and complementary products." Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015.

[13] Link to code: `https://github.com/svnathan/224w_window`