Applying Link Prediction to Recommendation Systems for Amazon Products

CS224W Project Final Report

Evan Darke, Zhouheng Zhuang, and Ziyue Wang

Abstract

In this paper, we analyze various link prediction algorithms on the Amazon product co-purchasing dataset. We formulate the problem as an instance of information retrieval where we are given some query node q representing a product, and we want to retrieve a ranked list of the top products likely to be co-purchased with q. Although the Amazon dataset contains a lot of metadata relevant to machine learning for each product, this paper will focus primarily on recommendations based on inherent properties of the network structure. Furthermore, we will analyze how the performance of each approach degrades when less co-purchasing information is available.

1 Introduction

Recommendation systems are important in online marketing to both sellers and consumers. On the one hand, for sellers, recommendation systems are used to maintain high levels of user engagement. On the other hand, for consumers, recommendation systems help efficiently identify items they are looking for. For example, Amazon recommends products to users based on their purchasing history to encourage future purchases, and news websites recommend related articles to readers to maximize ad revenue.

However, recommendation systems usually face cold-start problems. For example, Amazon uses the co-purchasing information to recommend products, but if a product is new, there is little co-purchasing information available. Furthermore, it is often not computationally feasible to track co-purchase statistics for all pairs of products, and therefore only estimations of the top co-purchased products may be available for each product. The question is, how much co-purchasing information is needed to make good recommendations?

Suppose you are an online retailer, and when a user adds an item to their shopping cart, you have real-estate to display k recommended products, for some value k dependent on the size of the user's device. You, however, only have resources to track the top $m < k$ co-purchases for each product. How well can we predict the top k co-purchases given the top m? This differs from work in [6] where the authors want to predict a variable number of links that form in a given time step by learning a metric with some threshold.

We propose several ranking measures to identify potential co-purchased product pairs, and analyze their performance with various levels of prior co-purchase information. We show recommendation quality generally increases with more prior information, but decent recommendations are obtainable even without any co-purchase information.

2 Related Work

Grover and Leskovec [6] propose a new algorithm, node2vec, which generates feature embeddings for nodes of a graph. The algorithm uses biased random walks to learn the neighborhood and structural role of each node in the graph and encode it in a low dimensional euclidean feature space. For link prediction, the authors composed node embeddings into edge embeddings and fed the edge embeddings into a logistic classifier. The authors achieved state of the art results using this method for link prediction in undirected social networks.

Given our initial analysis of the network, we suspect that the community embeddings learned by node2vec will be useful for predicting co-purchases. However, our graph is directed and sparser than the networks in this paper. For our directed graph, we will experiment with composing node embeddings with asymmetric functions to encode the directionality of each proposed edge.

Liu et al[4] use several machine learning approaches to predict co-purchase links in the Amazon product co-purchase dataset. The authors treat link prediction as an instance of binary classification, where given some proposed edge, they classify whether the edge looks like a co-purchase relationship base on product ratings, titles, time of first review, etc. The authors report their 0-1 accuracy on a test set with an equal number of positive and negative examples.

For our purposes, we can use any binary classifier to solve the ranking task by simply ordering nodes by the score outputted by the classifer. However, our results show that a high accuracy in this binary classification formulation does not necessarily translate to high performance in the ranking task. In 5, we will describe an evaluation procedure which addresses the short comings of this approach.

3 Dataset and Data Preprocessing

3.1 Raw Dataset

We use Amazon product co-purchasing network metadata (http://snap.stanford.edu/data/amazon-meta.html) to find similar products for recommendations.

The data was collected by crawling Amazon website and contains product metadata and review information. It contains 548,552 different products (Books, music CDs, DVDs and VHS video tapes) and 1,788,725 product-product edges. Each product description contains most of the following information (See example 9.1 in appendix):

- Id: Product id (number 0, ..., 548551), unique for each item
- ASIN: Amazon Standard Identification Number, unique identifier for each item
- Title: Name/title of the product
- sales rank: sales rank[8] is the rank of the sale of this product among all the items. The smaller the number is, the better the sale of this product is.
- List of 'similar' products (that get co-purchased with the current product)
- product categorization
- Product reviews: time, customer, rating, number of votes, number of people that found the review helpful

3.2 Data Preprocessing

Since nodes differ in popularity, co-purchase relationships are not necessarily symmetric. In

fact, only 27.23% of co-purchase links are bi-directional. We represent the dataset as a directed graph with a node for every product and a node for every category. There is a directed edge from product node p_1 to p_2 if p_2 appears in top co-purchase list of p_1. There is an edge from a product node p to a category node c if product p is a member of category c.

From this original graph G, we create 6 directed subgraphs. First, we define G_n for $0 \leqslant n < 5$ by repeating the process of creating G but only including up to n co-purchase links per node. For each graph G_n, we will attempt to predict co-purchase edges that appear in G but not G_n. Note that G_0 contains only category information and no co-purchase information, we will therefore also refer to this as the category graph. We also define the co-purchasing network as the graph created by removing all category nodes from G so that only product nodes and co-purchase edges remain.

3.3 Training and Test set

Most of the products (more than 300,000) have five outgoing co-purchase edges. For the training set and the test set, we only choose the products among these 5-out-degree products. We randomly select 200,000 product-product edges as our training set for machine learning method and 1,000 products as our test set.

We run our algorithms on G_0, G_1, G_2, G_3, and G_4, separately. Due to the large number of nodes in the network, we only consider nodes that share at least one category with the query node. This greatly reduces the computational power required for our task while still enabling us to predict co-purchases on up to 96% of nodes, as illustrated in 1.

4 Network Structure

Our intuition is that similar products should be purchased together, but initially we cannot be sure if co-purchasing relationships represent substitutes, complements, or unrelated related products. We can gain insight on these relationships by analyzing the structure of the network.

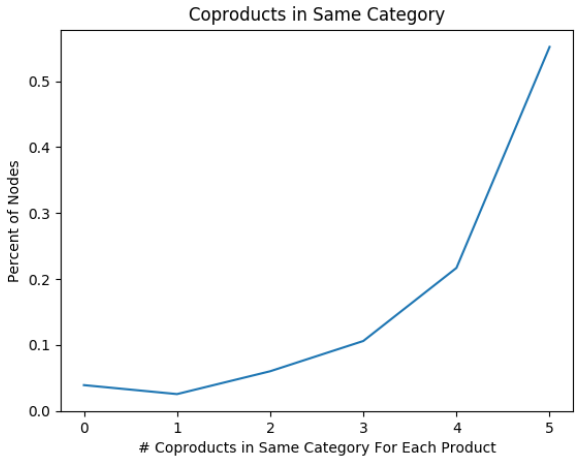

Figure 1. This figure shows for each products, how many of its co-purchased products are in the same category.

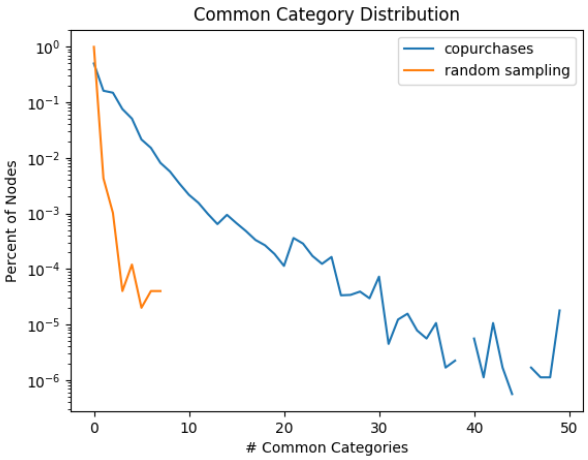

Figure 2. Common category distribution.

Fig 2 shows that most co-purchases have multiple categories in common as opposed to randomly sampled nodes which have disjoint categories 99.4% of the time.

As shown in 3, when we include categories and other known co-purchases in the nearest neighbor calculation, we see an even greater discrepancy between co-purchases and randomly sam-

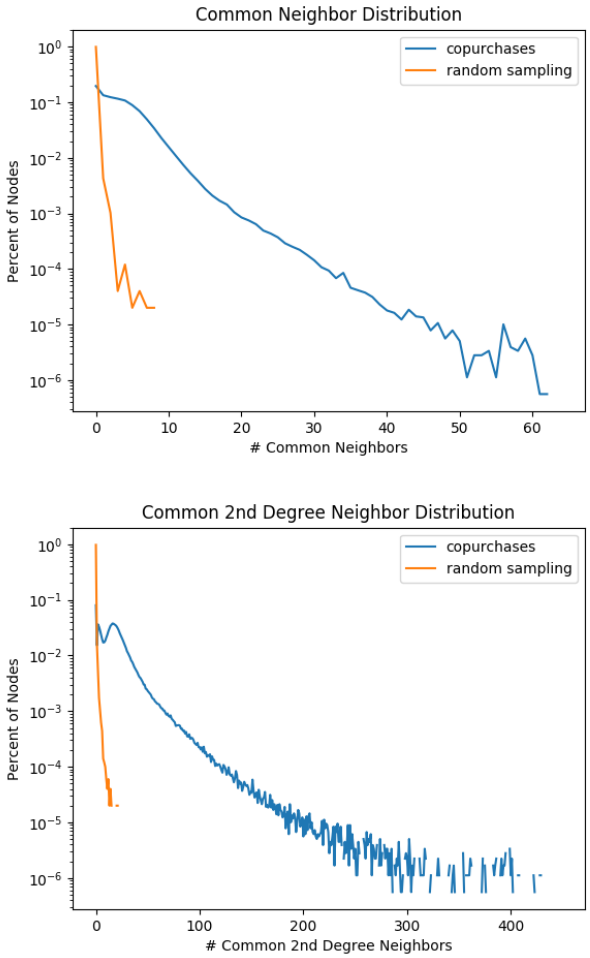

Figure 3. The upper plot shows the one-level common neighbor distribution, while the lower one shows the two-level common neighbor distribution.

pled node pairs. Furthermore, we measured a clustering coefficient of $\approx .3$ on the co-purchase network. These properties suggest that similar products are in fact likely to be co-purchased.

5 Methodology

5.1 Metrics

Previous work on this dataset [4] used 0-1 accuracy on classification of proposed edges. We found this metric to be misleading, so we instead propose two alternative relevant metrics: mean AUC and Recall@K averaged over each query node.

Recall@K

Recall@K is the ratio of the number of relevant products that appear in our top K results to the total number of relevant products. For instance, suppose we have room to display K recommended products to a customer. We're not too concerned about the order in which they are laid out or how many false negatives we get, as long as all the relevant products can be displayed.

AUC

AUC summarizes the expected number of negative examples that are ranked higher than positive examples. A score of 1 indicates that all positive examples are always at the top of the list, a score of .5 is the expected measure of a random ranking algorithm, and a score of 0 indicates the algorithm produces an inverted ranking. AUC is more relevant than recall when the order of our ranking is important, or when we care about more than the top K results.

6 Approaches

6.1 Proximity Measures

We define several score functions S(q, p) that represent relevance of product p to a query node q under various models. Let $\Gamma_i(u)$ be the set of nodes within i undirected hops of u.

Preferential Attachment Model

In the preferential attachment model, the probability of an edge (q, p) forming is proportional to the product of degree of q and p [6]. Since our graph is directed, we modify this definition to be

the product of the outdegree of q and the indegree of p.

$$S(q,p) \propto deg^(p)$$

Common Neighbors (CN)

Our initial data exploration suggests there is a strong correlation between common neighbors and co-purchases, therefore we suggest:

$$S(q,p) = |\Gamma_1(q) \cap \Gamma_1(p)|$$

Resource Allocation (RA) Model

The resource allocation model gives each node one unit of influence, which divides evenly among its neighbors [7]. Intuitively, this augments the common neighbors measure by recognizing that finding a common category that has size 2 is far more informative than finding a common category that, for example, has size 1,000.

$$S(q,p) = \sum_{z \in \Gamma_1(q) \cap \Gamma_1(p)} \frac{1}{deg(z)}$$

Jaccard Index

A similarity metric based on common neighbors [4].

$$S(q,p) = \frac{|\Gamma_1(q) \cap \Gamma_1(p)|}{|\Gamma_1(q) \cup \Gamma_1(p)|}$$

Common Two-level Neighbors

Number of common nodes within 2 hops of both p and q.

$$S(q,p) = |\Gamma_2(q) \cap \Gamma_2(p)|$$

Common Two-level Neighbors Weighted by RA

Sum of reciprocal degrees of all nodes within 2 hops of q and p.

$$S(q,p) = \sum_{z \in \Gamma_2(q) \cap \Gamma_2(p)} \frac{1}{deg(z)}$$

6.2 Logistic Regression Trained Feature Vector

This approach generates a feature vector based on the relationship between the two products whose edge probability we want to predict on. Our goal is to train the weight vector on the edge features such that the dot product of the weight vector and edge features of two products will be high for positive edges (if they are co-products) and low for negative edges (if they are not co-products).

6.2.1 Choice of Feature Vector

We picked the following as our edge features:

Jaccard of common categories (JacCategory). From the graph analysis, we know many products share the same categories.

Destination sales rank score (SalesScore) calculated from the sales rank r_s: $1/(1+ln(r_s))$. We assumed that if an item has a higher sales rank, it sells more than any other items and will more likely to have more incoming edges in the co-product graph. Because sales rank ranges from 1 to 100,000s, we choose to take $1/(1 + ln(r_s))$ as our feature so that the sales rank score falls between 0 and 1 and for large sales rank value, the sales rank score wont get too small.

Title similarity (TitleSim). We assumed people are likely to buy books with similar titles together. We utilized nltks wordnet to calculate sentence similarity from word similarities. We utilized corpus-based measure [1] and some of its implementation codes for the calculation[2]. The similarity score ranges from 0 to 1, with 0 meaning no similarity at all and 1 meaning the same title.

Jaccard of common reviewers (JacReview). We assumed that more common reviewers between two products mean more potential common buyers and they are more likely to be co-products.

For graphs with at least one co-product information, we also used the following features: **Jaccard of co-product neighbors** (JacNeighbor).

We assumed that two products having same co-products are likely to become co-products as well.

Jaccard of co-product neighbors total categories (JacNeighborCat). We assumed if the neighbors categories are similar for two products, they are likely to become co-products as well.

Jaccard of co-product neighbors reviewer set (JacNeighborRev). We assumed that if the neighbors reviewers are similar, they are likely to share common buyers as well.

Each of the features chosen is on a range between 0 to 1 so that its easier for training and easier for us to compare the weight between different features.

6.2.2 Parameter Choice

We chose logistic regression as our model because it produces continuous output which is interpretable as a probability estimation of an edge being positive. Using this probability, we can rank other nodes from high to low probability given a specific query node, representing how likely they are to be similar products.

We also need to balance the positive and negative samples [3]. This is because for each node, we only have at most five co-products in the ground-truth data of over 500,000 nodes. Positive edges are exceptionally rare. If we dont balance the samples, the logistic regression result will tend to predict every edge as a negative example, which yields nearly 100% accuracy when sampling edges between two random nodes. This result is useless when we try to predict links. Therefore, we balanced positive and negative samples by putting more weight on the positive samples during the training so that a mis-categorization of the positive sample will receive more penalty. The weight is equal to the ratio of the positive to negative edges so that the positive edges in total has the same impact as the negative edges.

7 Evaluation

7.1 Baseline Results

We define three baseline algorithms. Our first baseline is to rank products randomly. Second, we recommend products according to their popularity (salesrank). Unfortuntely, salesrank data is missing for many nodes, but we hypothesize that the indegree of a product is inversely correlated to its salesrank. Figure 4 verifies this intuition, so we include scoring for the preferential attachment model in our baselines as a popularity metric which can be directly inferred from the graph structure and is more robust to missing data. The baseline results are shown in tables 1 and 2

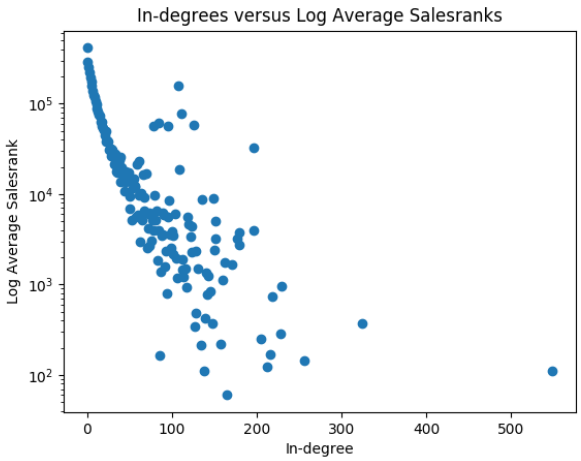

Figure 4. log mean salesrank per indegree.

In tables 1 and 2, we see that the preferential attachment model seems to quickly approximate the performance of using a node's popularity, even with little available co-purchase information and does significantly better than randomized rankings. However, it cannot discriminate between nodes in the G_0 case, so it does not address the cold-start problem.

7.2 Network-Structure Approaches

Since many of our proximity measures take on discrete values, we always break ties in the re-

	Random	SalesRank	P.A.
G_0	0.504	0.703	0.501
G_1	0.502	0.702	0.674
G_2	0.477	0.705	0.712
G_3	0.491	0.714	0.748
G_4	0.496	0.714	0.772

Table 1. Baseline AUC Results

	Random	SalesRank	P.A.
G_0	0.012	0.038	0.008
G_1	0.006	0.036	0.040
G_2	0.009	0.040	0.053
G_3	0.008	0.038	0.057
G_4	0.008	0.029	0.062

Table 2. Baseline Recall@10 Results

	R.A	Node2vec	CN	Jaccard	2-level CN	2-Level CN with R.A.
G_0	0.82	0.60	0.74	0.74	0.74	0.82
G_1	0.86	0.60	0.84	0.84	0.90	0.92
G_2	0.90	0.62	0.91	0.91	0.94	0.95
G_3	0.93	0.64	0.94	0.94	0.96	0.97
G_4	0.95	0.68	0.96	0.96	0.97	0.98

Table 3. Proximity AUC Results

	R.A	Node2vec	CN	Jaccard	2-level CN	2-Level CN with R.A.
G_0	0.17	0.02	0.13	0.13	0.13	0.17
G_1	0.32	0.02	0.24	0.24	0.40	0.47
G_2	0.52	0.02	0.40	0.40	0.54	0.63
G_3	0.60	0.02	0.52	0.52	0.60	0.66
G_4	0.68	0.02	0.60	0.60	0.61	0.74

Table 4. Proximity Recall@10 Results

maining measures by computing the nodes' P.A. scores.

Of all the proximity measures we tested, we summarize results of interest in tables 3 and 4. Clearly, common neighbor based approaches work well when we have sufficient co-purchasing information, but perform poorly in the cold-start scenario. Conversely, the resource allocation model makes good predictions in the cold-start scenario, but doesn't benefit as much from the addition of more co-purchasing data. Therefore, we decided to try weighing our second-level common neighbors in a similar fashion to the resource allocation model and found the resulting measure dominates others in both in terms of AUC and recall.

Interestingly, most of the above measures other than P.A consider edges to be undirected. We found that Jaccard, common neighbors, and 2-Level common neighbors following directed edges performs slightly worse their undirected variants. This is probably because if node u is a top co-purchase of v, then v is likely a common co-purchase of u even if it does not make u's top 5. In this case, considering both incoming and

Additionally, we found that performance of these proximity based measures increase nearly monotonically as we include more prior co-purchasing information.

We also ran node2vec on each subgraph to obtain 128 dimensional embeddings for each product node. We experimented with various functions to create edge embeddings from node embeddings including subtraction, the Hadamard product, and absolute value of the difference between two node embeddings. We trained a logistic model to recognize whether edges represent co-purchases or not by sampling from a population with 50% positive and 50% negative examples. On most graphs, we were able to achieve a validation accuracy around 85% by defining edge embeddings as the difference between the

Graph	G_{m0}	G_{m1}	G_{m2}	G_{m3}	G_{m4}
JacCat	0.37	0.04	-0.19	-0.29	-0.34
SalesScore	0.20	0.18	0.16	0.16	0.15
TitleSim	1.49	1.34	1.22	1.10	1.03
JacReview	0.43	0.31	0.23	0.17	0.16
JacNbr	-	0.72	1.28	1.57	1.72
JacNbrCat	-	1.50	1.94	2.19	2.37
JacNbrRev	-	0.22	0.65	0.92	1.06

Table 5. Weight vector trained on each graph

embeddings of its source and destination node. This accuracy is comparable to that found in [4]. When ranking edges by the score from our logistic model, however, ranking performance failed to even match our P.A. baseline. This demonstrates that high accuracy on the recognition task doesn't necessitate high performance on the ranking task.

7.3 Logistic Regression Trained Feature Vector

Table 5 shows the weight vector trained from each graph. From the table, we can learn the relative importance for each feature. We find that the non-neighbor features (first four) become less important when we get to know more co-product neighbors. Also, neighbor's common categories (JacNeighborCat), common neighbors (JacNeighbor) and title similarities (TitleSim) have been important factors for co-product prediction. As we know more co-product neighbors, the common reviewers of neighbors (JacNeighborRev) have also become important. The evaluation result also keeps improving as more co-product information is added to the graph.

8 Time Complexity

For simplicity, we assume that the number of categories to which a product can belong is bounded by some constant. To create a ranked recommendation list for a node q in graph G_k with n nodes, we have to iterate over all product nodes in G_k and for each node compute a proximity measure. For our best measure, second level common neighbors with resource allocation, we must explore up to $O(k^2)$ nodes. Finally, we must sort the nodes by their score which can be done in $O(n \log n)$ time. The complexity of ranking potential co-purchases of a node is therefore $O(nk^2 + n \log n)$ time.

Metrics	Mean AUC	Recall@10
G_0	0.82	0.23
G_1	0.83	0.34
G_2	0.88	0.42
G_3	0.91	0.46
G_4	0.94	0.48

Table 6. Logistic Regression Link Prediction Results

9 Conclusions

We found that proximity measures are a good indicator of whether products are likely to be co-purchased. We also found that network properties alone can produce recommendation rankings comparable to machine learning applied to features extracted from product metadata. Moreover, we demonstrated that the more co-purchasing information we have, the better our predictions are.

Despite modeling the data as a directed graph, we found our metrics usually benefit from considering all edges to be undirected. We suspect this may explain the poor performance of node2vec since it was run on the directed graph. We would have liked to include results using node2vec embeddings from the undirected graph, but found that node2vec required far more computational resources to run on the undirected graph than we had available.

References

[1] Rada Mihalcea, Courtney Corley, and Carlo Strapparava *Corpus-based and Knowledge-*

based Measures of Text Semantic Similarity. In AAAI06, July 2006.

[2] Compute sentence similarity using Wordnet http://nlpforhackers.io/wordnet-sentence-similarity/

[3] Gary King and Langche Zeng. *Logistic Regression in Rare Events Data*. Political Analysis, 9, Pp. 137-163. 2001.

[4] Yilun Liu, Chunhao Wu, and Xiaohui Tong. *Prediction of Co-purchasing Products*

[5] Aditya Grover, Jure Leskovec. *node2vec: Scalable Feature Learning for Networks* 2016.

[6] D. Liben-Nowell, J. Kleinberg. *The Link Prediction Problem for Social Networks. Proc. CIKM, 2003.*

[7] Tong Wang, Xing-Sheng He, Ming-Yang Zhou, Zhong-Qian Fu. *Link Prediction in Evolving Networks Based on Popularity of Nodes. Nature. 2017.*

[8] Amazon, *https://www.amazon.com/gp/help/customer/display.html?nodeId=525376*.

Appendix

Example 9.1 *Raw data for one product.*

```
Id:   1
ASIN: 0827229534
  title: Patterns of Preaching:
  A Sermon Sampler
  group: Book
  salesrank: 396585
  similar: 5  0804215715
  156101074X  0687023955
  0687074231  082721619X
  categories: 2
   |Books[283155]|Subjects[1000]|
   Religion &
   Spirituality[22]|
   Christianity[12290]|
   Clergy[12360]|Preaching[12368]
   |Books[283155]|Subjects[1000]|
   Religion &
   Spirituality[22]|
   Christianity[12290]|
   Clergy[12360]|Sermons[12370]
  reviews: total: 2  downloaded: 2
  avg rating: 5
    2000-7-28
    customer: A2JW67OY8U6HHK
    rating: 5
    votes:  10  helpful:   9
    2003-12-14
    customer: A2VE83MZF98ITY
    rating: 5
    votes:   6  helpful:   5
```