Machine Learning in Network Science
Final Project Report

Mathieu TARDY, Jérémie FERON, Clément DE LOUBRESSE

April 2021

1 Abstract

"The recommendation system plays very important role in the e-commerce domain to recommend the relevant products/services based on end user preference or interest" (Karthik and Ganapathy, 2020). We have tried to implement such systems using graphs, which contained information about Amazon products as it will later be detailed. Representing the data in a graph, where two nodes (i.e. products) are linked by an edge if they are co-occurring in online baskets, enabled us to build a content-based recommendation system in the form of a supervised learning task. We wanted to characterize each node with a set of automatically generated features, to be eventually fed into some machine learning algorithms. This is where Node2Vec played a crucial role as it allowed us to extract features for each node by automatically exploiting the graph structure, providing our machine learning algorithms with very relevant embeddings yielding promising results. However, Node2Vec stays limited as it cannot take nodes features into account. GNN are helpfull to do that, and we therefore further performed graph representation learning with the GraphSage framework.

Keywords: Recommendation Systems, E-commerce, Graph Theory, Network Science, Supervised Machine Learning, Node2Vec, Graph Neural Networks, GraphSage.

2 Introduction and Motivation

Recommendation systems have become crucial for companies selling online products. It is indeed plausible to assume that there are large gains to be made if one is able to accurately suggest products that are similar to the one currently in the consumer's basket. First, from the users' perspective, being offered the opportunity to be assisted in their buying decision process is likely to be something customers value. Second, from the seller's perspective, those systems allow for increased average baskets' sizes as the recommendation's accuracy increases, hence significant financial gains.

In this project, we want to investigate whether it is possible to provide accurate recommendations using graphical data.

3 Problem Definition

3.1 Available Dataset: Amazon Metadata

For this project, we will be using a dataset which contains information about Amazon's products, more specifically its co-purchasing network metadata, available at https://snap.stanford.edu/data/amazon-meta.html.
The downloadable dataset is presented in a text file, containing product ids with specific characteristics: Amazon Standard Identification Number (ASIN), title, group (Book, DVD, Video or Music CD), sales rank (which indicates the ranking of the product in terms of sales volume), categories, reviews (which contain the total number of reviews, user, time, and how many users found such review helpful). The natural step was therefore to transform this text file into a dictionary that could be eventually used to create our graphical data.
The dataset presents the following characteristics (*Table 1*). It must be noted that although some features were already available on the given webpage, we thought it was good training to compute these again.

Characteristic	Value
Products (nodes)	548,552
Edges	1,545,228
Average Degree	2.78525
Reviews	7,781,990
Books	393,561
Music CDs	103,144
Videos	26,131
DVDs	19,828

Table 1: Main Characteristics of Amazon Metadata

Moreover, we present here some graphs related to the degree distribution of our graphical data. As it will later be explained, for the purpose of this project we had to use a relevant and representative sub-graph for our analysis, we nevertheless here provide information regarding the full graph.

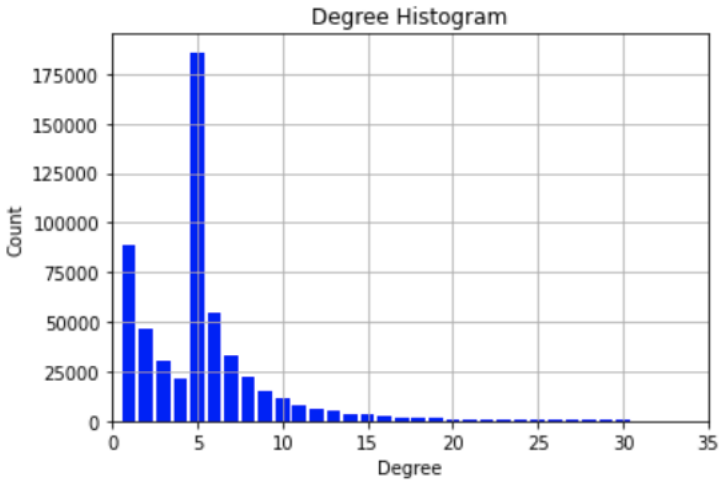

Figure 1: Degree Distribution - Full Graph

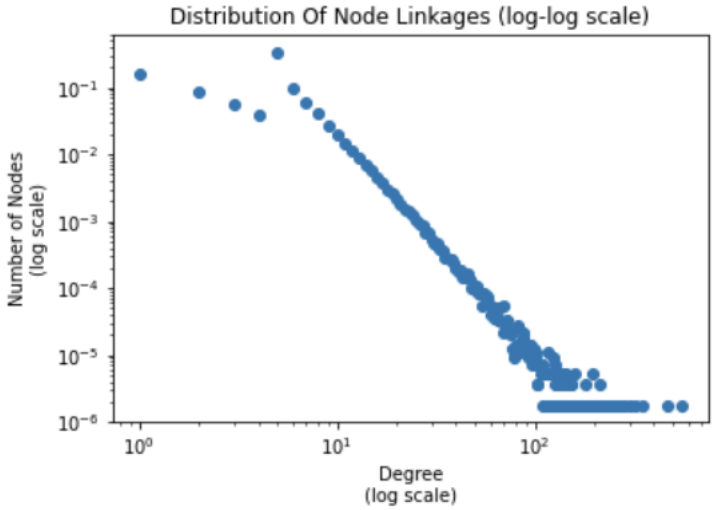

Figure 2: Log-Log Scale Degree Distribution - Full Graph

We note here that the degree distribution, once transformed into a log-log scale can be approximated using a straight line, hence it is plausible to assume that it follows a power-law distribution.

3.2 Creating a Relevant Sub-Graph for Analysis

Because of the size of the initial graph, it was plausible to assume that we would face computational issues in our analysis. Hence, we decided to create a relevant sub-graph on which we could eventually apply the concepts learned in class to

produce a relevant recommendation system.
We therefore created a Python class, namely *GraphProcessor* that is handling the task of creating a representative sub-graph. This class essentially implements Breadth-First-Search sampling. As noted by Kurant and his colleagues (2011), this technique presents some advantages over random walk and other sampling technique, notably by generating a 'plausible graph on its own'. Importantly, literature (Wang et al. 2011, Kurant et al. 2011) has shown that BFS sampling could create samples that are biased towards high-degree nodes. We assumed here that given the degree distribution shown in the previous figure, this technique could generate a representative sample for our purpose.
Our sub-graph contains 5,142 products (i.e. nodes) with an average degree of 2.4, with 12,343 edges, and *Figure 3 and 4* detail the degree distribution characteristics, which comforted us in keeping this sub-graph for our subsequent analysis.

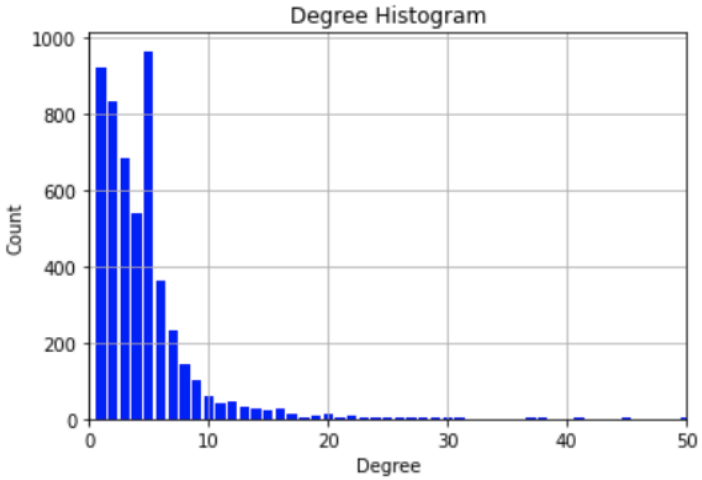

Figure 3: Degree Distribution - Sub-Graph

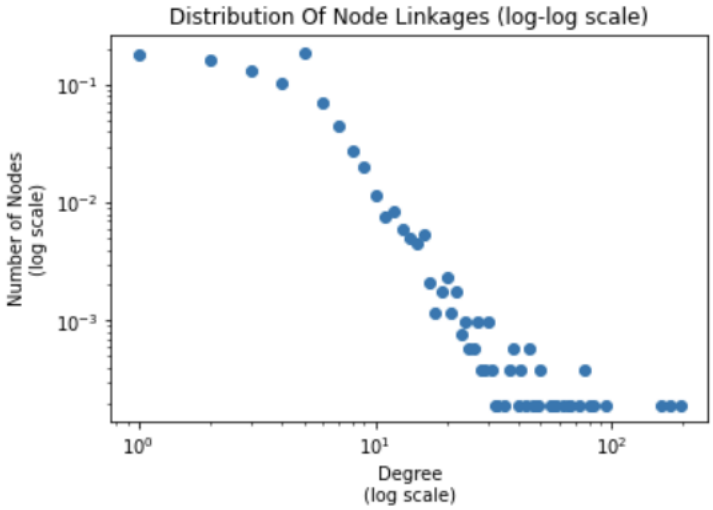

Figure 4: Log-Log Degree Distribution - Sub-Graph

4 Related Work

In 2003, Amazon researchers Greg Linden, Brent Smith and Jeremy York published a paper called '*Amazon.com Recommendations: Item-to-Item Collaborative Filtering*', which described in details how to personalize customer experience on Amazon.com, providing high-quality products recommendations in real-time with the ability to scale to massive datasets.

Since the publication of this paper, a lot of research has been done to improve the quality of recommendation systems, with a growing emphasis on graphical data. Indeed, Prasad et al. (2017) undertook an '*Analysis of the Co-Purchase Network of Products to Predict Amazon Sales-Rank*' in which they introduced a network-assisted approach that would be able to generate suitable features for prediction (notably neighborhood-based) yielding very promising results for the variable in consideration.

As noted earlier, for the purpose of our project, it was strongly recommended to work on a sub-graph due to computational complexity. Hence, our approach detailed in a previous section (namely BFS sampling) relies on past research (Wang et al. 2011, Kurant et al. 2011) which emphasized that such techniques have proved to be reliable for sampling representative graphs.

Lastly, as it will later be detailed, we have implemented an inductive framework called GraphSage introduced by Hamilton et al. (2018), which enabled us to '[leverage] node feature information (e.g., text attributes) to efficiently generate node embeddings for previously unseen data'.

Building up on that related work, we have developed a careful methodology which is detailed in the subsequent section, able to generate very promising results.

5 Methodology

We decided to compare several algorithms along with different representations of the graph. We will first describe how we computed those different representations and next the results of our models.

5.1 Key Metrics for Reliable Predictions

Building up on our experience from the Kaggle Competition (Link Prediction Task), we have computed several promising features to be eventually incorporated in our models.
First, we computed the Resource Allocation Index, defined as $\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(w)|}$ for nodes u and v. It is defined as the sum over all common neighbors of u and v of one over the degree of these nodes. If these two nodes have a lot of common neighbors, then their resource allocation index will be important. Hence, it will give us an intuition about the similarity of two products: if products A and B have a large RA index, hence there exist other products with which they are frequently bought with.
Second, we computed the Adamic-Adar Index which was one of the most important feature for our previous Link Prediction task.
This metric is defined as $\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log |\Gamma(w)|}$, again for nodes u and v. We note that the latter is very similar to the the Resource Allocation Index, with the difference being that we divide by the log of the degree.
Furthermore, it must be noted that we have tried other metrics such as Jaccard Coefficient, number of common neighbors or the Preferential Attachment Score. And lastly, we have provided nodes with additional features, namely centrality ones, either based on distances (notably closeness) or paths (betweenness). In our final output, we decided to keep two features, namely the resource allocation index and the adamic-adar index, because all the centrality measures and neighborhood based features had very low predictive importance for our classifiers.

5.2 Node2vec

Working with a portion of the graph we isolated we generated a 100 walks of length 8 for each node. We then used Node2vec to compute embeddings of size 64 for each nodes, training with a window size = 5 and batch words = 4.

We computed the cosine similarity between the obtained embeddings as a feature for our predictive models (cosine between u and v representations for a (u,v) pair). To see if we could improve performances further we added the raw embeddings to the graph and trained a graph neural network to further improve the representation of the nodes.

5.3 GraphSage

We implemented a neural network with a feature representation layer and a predictor layer (for training purposes). The feature representation part of the model consisted of successive layers of Pytorch's implementation of the GraphSage operator with relu activation in between. The prediction part simply performs a dot product between newly computed embeddings of nodes. For performance and comparison purposes, once the network is trained we save the embeddings outputted by the GraphSage and discard entirely the predictor. Note: Since our predictor doesn't have any weights, the training here is really about getting better embeddings via the sage layer. Again in our final models we computed the cosine between u and v representations for a (u,v) pair.

For training, we created a "negative" graph which has the same nodes as our base graphs but all the edges are edges that do not exist in the base graph. At each epoch our model outputs representations for all nodes of both graphs. We then compute the "score" of each edge with a dot product, note that we did not use sigmoid here for saturating gradients issues. Next, we compute the loss by comparing the obtained score to the true labels (1s for the true graph and 0s for the negative graph) using binary cross entropy. We used Adam as an optimizer and stopped at 500 epochs around which the network started to overfit.

5.4 Predictive Models

The objective of our project is to determine whether two products are often bought together. We have represented these relationships as edges and the products as a node in our graph. We then want to train a model to find those relationships. Our task is a link prediction problem.

In this section, we will detail how we built our training dataset, talk about our feature engineering step to fit different classifiers with the right data and, finally, we explain our training and validation phase.

First of all, we used our GraphProcessor class to build a dataframe to compare pairs of nodes. This training dataset contains all the nodes connected by edges in our subgraph and 10 thousand additional random pairs of nodes. Most of them are not connected by edges. We were careful not to keep duplicates comparison. We then added our dependent variable y which represents if the two nodes are connected by an edge.

Secondly, we performed some feature engineering. As detailed in the previous section, all nodes in our subgraph have two embedding representation computed using node2vec and GraphSage. In order to measure the similarity between two nodes, we computed the cosine similarity of pairs of embeddings. We did this for all pairs of nodes compared and both embedding methods.

Finally, we added those two metrics, the resource allocation index and the Adamic-Adar index to our previous dataframe. This is the final training dataset that we have used to train and test our classifiers performances.

For the model training phase, we tried 11 different classifiers such as the logistic regression model, an XGBoost classifier, an AdaBoost classifier, a Support Vector Machine classifier and many others.

6 Evaluation

Our goal is to predict edges between two nodes. To measure the performance of our models, we computed three evaluation metrics such as the F1-score, the AUC and Accuracy.

Depending on which metrics we decided to input in our training, different models performed best.

Classifier	F1	Accuracy	AUC
LinearSVC	98.9%	99.0%	98.9%
LogisticRegression	98.9%	98.9%	98.9%
XGBClassifier	98.9%	98.9%	98.8%
RidgeClassifier	98.8%	98.8%	98.7%
AdaBoostClassifier	98.8%	98.8%	98.7%
GradientBoostingClassifier	98.8%	98.8%	98.7%
LinearDiscriminantAnalysis	98.8%	98.8%	98.7%
RandomForestClassifier	98.6%	98.6%	98.6%
KNeighborsClassifier	98.5%	98.5%	98.5%
DecisionTreeClassifier	98.1%	98.2%	98.1%
GaussianNB	96.7%	96.8%	97.0%

Figure 5: Results with all metrics

As you can see from the above table, all models perform very well. The difference between the best performing model and the least performing one is of 2.2%. This difference is very low. This shows that our chosen features are very good separators of our data. All classifiers are very accurate in predicting if two products are usually bought together. We can confidently say that these model can be used as recommender systems. If the model predicts a link with very high accuracy, there is almost 99% chance that the two products are usually bought in the same basket.

Moreover, all features do not necessarily have the same predictive power. Node2Vec is overwhelmingly more important relative to the other three (cf. figure 6).

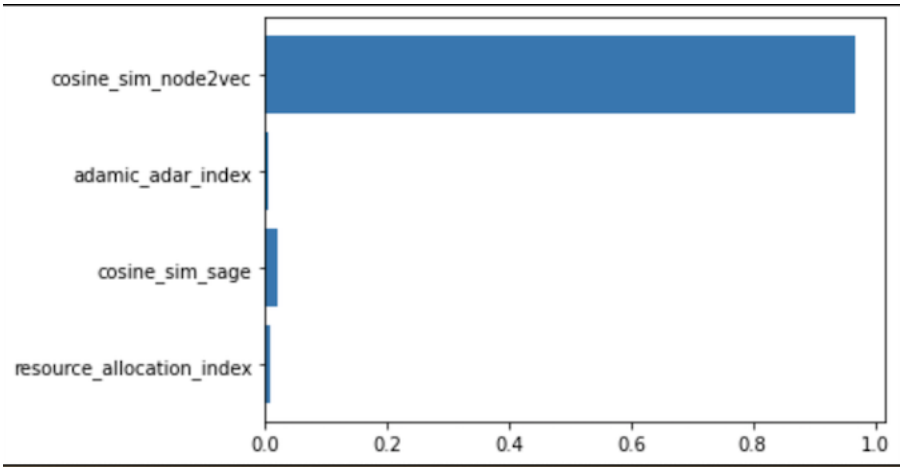

Figure 6: Feature importances of XGBoost Classifier

7 Conclusion and Further Research

In this project, we have shown that we can use graph models to perform product recommendations. We have transformed an Amazon dataset, which describes products that are usually bought together, into a graph. We applied two models to embed each node and computed the cosine similarity between pairs of nodes. We added two powerful neighbourhood based features to our comparative dataset. We then trained classifiers and evaluated our results. We can conclude that our approach was very efficient in this link prediction task. We are confident in saying that we can apply this methodology in the context of content-based recommender systems.

Further analysis can be made to validate the models' performances by increasing our dataset size. To verify the generalisation of our methodology, we could increase the depth used in our BFS and add more random comparisons of pairs of product. Moreover, it would be interesting to dig into the performance of our model with a more imbalanced dataset.

Moreover, we could have added more information about the nodes to compute more sophisticated similarity metrics. We could have exploited even more the metadata of our dataset. We are confident that we could increase our performance even more even we would continue our research.

8 Bibliography

- Inductive Representation Learning on Large Graphs. W.L. Hamilton, R. Ying, and J. Leskovec arXiv:1706.02216 [cs.SI], 2017.
- Node2Vec: Scalable Feature Learning for Networks. A. Grover, J. Leskovec. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.
- R V, Karthik Sannasi, Ganapathy. (2020). Online Product Recommendation System Using Multi Scenario Demographic Hybrid (MDH) Approach. 10.1007/978-3-030-63467-4 20.
- Kurant, Maciej Markopoulou, Athina Thiran, Patrick. (2011). Towards unbiased BFS sampling. IEEE Journal on Selected Areas in Communications - JSAC. 29. 10.1109/JSAC.2011.111005.
- Wang, Tianyi Chen, Yang Zhang, Zengbin Xu, Tianyin Jin, Long Hui, Pan Deng, Beixing Li, Xing. (2011). Understanding Graph Sampling Algorithms for Social Network Analysis. IEEE Computer Society. 123-128. 10.1109/ICDCSW.2011.34.
- Prasad, Utpal Kumari, Nikky Ganguly, Niloy Mukherjee, Animesh. (2017). Analysis of the Co-purchase Network of Products to Predict Amazon Sales-Rank. 197-214. 10.1007/978-3-319-72413-3 13.