Analyzing co-purchasing network for product recommendation
NGSA Project Final Report

Assael Jérémi
CentraleSupélec
Jeremi.assael@supelec.fr

Auriau Vincent
CentraleSupélec
Vincent.auriau@supelec.fr

D'Esquerre Arnaud
CentraleSupélec
Arnaud.desquerre@supelec.fr

ABSTRACT

The main goal of the project is to recommend products to a customer using its past purchases and some patterns extracted from the analysis of a co-purchasing network from Amazon website. We worked on four different methods, using either the graph structure of the co-purchasing network or the meta-data and the information provided with each product (description of the product, reviews, categories, etc…). The first method is based on motifs detection and overlapping communities, the second one on neighbor's detection, the third one on a SVD of the transition matrix and finally the last one on a TF-IDF algorithm on product titles. In this report, we will discuss the different results we obtained, the advantages and drawbacks of each method and why we chose these ones. We also discuss issues about the evaluation of the algorithms, why it's difficult and how it could be possible to overcome these difficulties.

Keywords
Recommendation, Amazon co-purchasing network, neighbors, motifs detection, SVD, TF-IDF

1. INTRODUCTION/MOTIVATION

Recommendation algorithms are particularly used on e-commerce websites. Using customer's interest, a list of recommended items is generated, in order to help the customer to find new interesting items. From one customer to another, the customer experience can be totally different because the recommendations are not the same. The click-through and conversion-rates are two particularly important measures of the effectiveness of a recommendation campaign. It's important for e-commerce platforms to have a well performing recommendation algorithm, in order to keep customers and to increase revenue. . For example, Amazon has one of the best performing recommender system and is known for being particularly efficient. With the growing number of products, it's also important to help the customer finding the right item. The project focuses on experimenting different methods and pointing out their advantages and drawbacks. We finally discuss why evaluation is an issue and how we could tackle it.

2. PROBLEM DEFINITION

Let's describe the data we will use to experiment our recommendation algorithms. We will use a modified version of the Amazon Co-purchasing Network.

The Amazon Co-purchasing Network was collected by crawling Amazon website. 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 a directed edge from i to j. The data was collected in March 02 2003. We will use the smallest version of the datasets collected from the publicly available Stanford Network Analysis Platform [7]. The Dataset statistics are described in the Table 1.

The network is quite dense and the fraction of nodes belonging to

Table 1. Dataset Statistics

Nodes	262111
Edges	1234877
Nodes in largest WCC	262111 (1.000)
Edges in largest WCC	1234877 (1.000)
Nodes in largest SCC	241761 (0.922)
Edges in largest SCC	1131217 (0.916)
Average clustering coefficient	0.4198
Number of triangles	717719
Fraction of closed triangles	0.09339
Diameter (longest shortest path)	32
90-percentile effective diameter	11

the largest strongly connected component is very high (0.922). It means that the products are very closely connected to each other. We can also see this propriety with the diameter which is very small compared to the number of nodes.

We also possess meta-data information for each product. The information and an example is given in the Table 2.

We use a subset of the dataset, in order to limit the computational time. Indeed, only to parse the whole metadata file, a several hours were needed. The dataset is reduced to a smaller graph containing 4869 nodes and 5396 edges. In order to get this dataset, two constraints were used: Only books were picked and the product needed enough reviews (5 at least).

Using this network, the goal is to compare a few approaches to item recommendation. For any product in this graph, the algorithms have to come up with a few items in which a customer of the initial product would be interested in. Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user's purchased and rated items. From there, a typical compartment is created and products are recommended using these similar customers' purchases. Here we will try to work only using the co-purchasing network and thus without a purchase history of the customer.

Table 2. Amazon Product Meta Data

Property	**Value**
Id	1
ASIN	0827229534
Title	Patterns of Preaching: A Sermon Sampler
Group	Book
Sales rank	396585
Similar Products	5 0804215715 156101074X 0687023955 0687074231 082721619X
Categories	2 |Books[283155]|Subjects[1000]| Religion & Spirituality[22]| Christianity[12290] |Clergy[12360]| sPreaching[12368]
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

The different approaches are using either the meta-data provided with the network or the graph architecture of the network.
In an e-commerce context, recommendation algorithms try to optimize click-through and conversion rates on the e-commerce website. Of course, in this project we couldn't make sure our strategies would be efficient in this context. It's why we will discuss in the evaluation part how to proceed to evaluate this kind of algorithms. Moreover, it's better to have a few strategies to test an algorithm before launching it live.

3. RELATED WORK

Recommender systems can be splited in different ways. The most common way is to make a difference between content based and collaborative filtering systems. The former is mainly based on the features of the products such as meta-data: title, categories. The latter on the other hand, does not require any information about the items or the users themselves. It recommends items based on users' past behavior.

Inside the collaborative filtering systems there exist two different type of systems: User-based and Item-based. The first one uses similarity between the target user and other users. The second one is based on the similarity between the items that target users' rates and other items. The main idea behind collaborative filtering is that similar users share the same taste and that similar items are liked by a user.

Another way to divide collaborative filtering techniques is to make a difference between memory-based and model-based methods. The memory-based approach uses user rating data to compute the similarity between users or items. On the contrary in the model-based approach, models are developed using different data mining, machine learning algorithms to predict users' rating of unrated items. There are many model-based collaborative filtering algorithms: Bayesian networks, clustering models, latent semantic models such as singular value decomposition…

In this project we tried to cover each kind of recommender system. The first model can be considered as an item-based collaborative filtering algorithm. It is inspired [3] by a motif-based analysis on the same network. Indeed in this article, motifs are used to describe a network, both on the micro and on the macro level. The second model can be seen as a user-based collaborative filtering system. The third one is a model-based collaborative filtering recommendation engine. The last one is a content-based system.

As we will see in the evaluation part, each of them yields very different results.

4. METHODOLOGY

4.1 First Method: Motifs Detection

In this model we will use a strategy based on motifs detection, inspired by the article [3]. We assume that neighbors and triangles are an important information to detect the closest nodes. We will define a few other motifs and in order to make recommendation, we will use two evaluations:

First, we will apply a Clique Percolation Method in order to define overlapping communities. We assume that nodes inside a same community are really close to each other and their closeness score

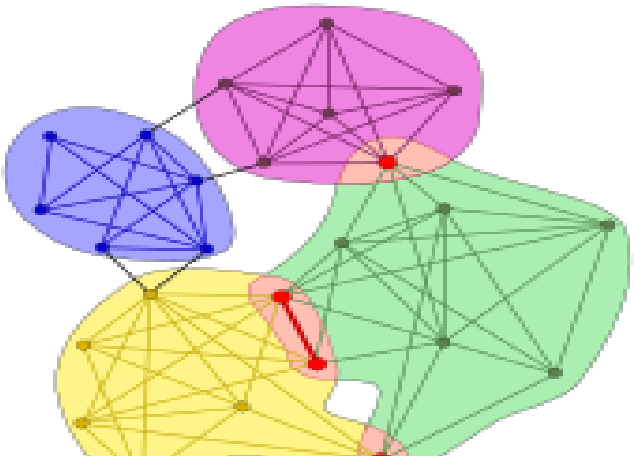

Figure 1 Example of overlapping communities.

will be increased. Secondly, we assume that two nodes that take part into a great number of common motifs will see their closeness score also increased.

The clique percolation method is an approach used for analyzing the overlapping community structure of a network. A community might be defined as a group of nodes that are more densely connected to each other than to other nodes in the network.

The clique percolation method creates communities using k-cliques, which correspond to fully connected sub-graphs of k nodes. Two k-cliques are considered adjacent if they share (k-1) nodes. From this, communities are defined as the maximal union of k-cliques that can be reached from each other through a series of adjacent k-cliques.

This definition allows overlaps between communities. One node can belong to several communities as we can see it in the Figure 1. This image shows four k-clique communities with k=4. Each community has its own color and the overlap are in red. In order to get a fast implementation, it's best to focus on locating all maximal cliques (and thus finding the graph maximal clique, which is a NP-hard problem). In the worst case, the complexity of such an algorithm is exponential in the number of nodes.

The algorithm also exists for directed graphs (using directed k-cliques), however, because the average degree of the graph is low (and most nodes have only a few neighbors), we got better results working on non-directed graphs. Even if it changes the definition of the graph, it doesn't sound absurd as two products are most of the time equivalently connected to each other.

Another feature that will describe the closeness of two nodes is the number of common motifs they are involved in. On this purpose, we define a few motifs. We can see examples of these motifs in the Figures 2-5. In the paper Motif Analysis [2], we can see that the Amazon co-purchased network has many motifs. We thought that it could be meaningful to use this property in order to make recommendations. Let's take a product a customer has bought. For each other product (represented by a node in our graph), we will check how many of common motifs these two nodes take part in. The top ranked nodes are the ones we want to recommend because they are closely connected to our studied product. The motifs can be seen as a wider description of the closeness of two nodes (as it is done with triangles).

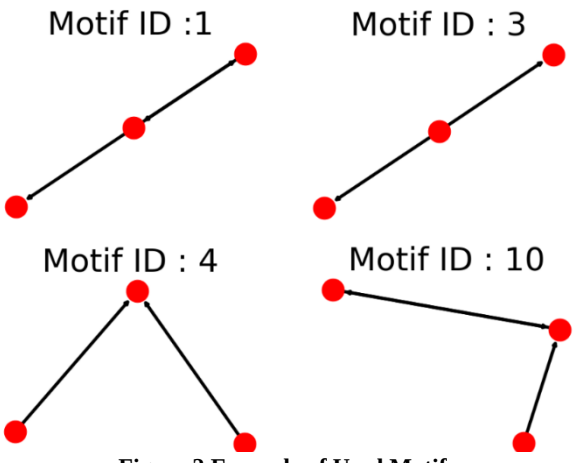

Figure 2 Example of Used Motifs.

Combining the two strategies, we can come up with a recommendation strategy. Each node is given a score that is increased using communities and motifs. We finally recommend the top ranked nodes. In order to have a better recommender system, we could define more motifs or study the one that are the most meaningful. This method has strong limitations as it will mostly recommend close neighbors. This strategy can be really interesting combined with another one. Also, we only worked on a small network as we had strong calculation limitations, but with really dense networks, it might give better results.

4.2 Nearest Neighbors in a Weighted Directed Graph

Let's consider that we have the Amazon co-purchasing graph built with the nodes whose information had been parsed.

The graph being a co-purchasing network, it gives us information of products that are frequently bought together. This algorithm [8] assumes that if a client buys a product, he is likely to like products that are co-purchased with this one. These products are the nearest neighbors of the considered article in the co-purchasing graph. However, this information is not enough. Products may be bought together but it does not bring any information on whether the other product was effectively liked or not. Therefore, we are transforming our directed graph into a weighted directed graph considering the average review, on a scale from 0 to 1, of each product provided by the clients as well as how helpful these reviews were for past clients, on a scale from 0 to 1. A rating is assigned to each product according to one of the following models, according to the user choice:

- Additive: This model allows us to find a balance between how much weight we want to put on the average review and how much on the helpfulness. The α coefficient, determining this weight, is tunable by the coder.

The rating of product i is $r_i = \alpha \times average\ review + (1-\alpha) \times helpfulness$

- Multiplicative: in this model, we cannot choose the weight of the average review and the weight of the helpfulness, but it may provide better recommendation. This idea could be checked during evaluations procedures like A/B testing.

The rating of product i is $r_i = average\ review \times helpfulness$

We then compute the weights of the different edges in such a way that the better rated two products are, the closer they are in the graph. We then define the weight of the edge between the nodes i and j as follow:

$$w_{ij} = (1 - r_i)(1 - r_j)$$

Now that we have computed the weighted-directed graph, we only have to find the nearest neighbors of the considered product. These would be the recommendations. We considered two main methods:

- We can either assume that a user is going to like only product that are frequently bought with the considered one. Then, we are only going to consider the out-neighbors. We called this model the "only_with_it" model.

- Or, we can assume that a user is going to like products bought together, no matter if product A is frequently bought with product B or if product B is frequently bought with product A.

Then, we are going to study all neighbors of the considered article, meaning the successors as well as the predecessors. This is the "all" model.

This choice of neighbors can be tuned by the user. An evaluation of our recommendation algorithm would allow us to make the best choice.

Having the neighbors, we just have to rank by the distance to our product. The k first will provide k recommendation and the distance to our product will even provide a score for these recommendations.

4.3 Matrix Factorization via Singular Value Decomposition

The idea behind models 3 and 4 was to a compare recommender system based on graphical models to other commonly used techniques. It was also a way to exploit metadata as much as possible. Indeed, on the top of the co-purchased data, the titles and categories as well as the ratings of the users are available in this dataset.

Matrix factorization consist of decomposing one matrix into a product of multiple matrices. There are many different ways to factor matrices, but singular value decomposition is commonly used for making recommendations. At a high level, SVD is an algorithm that breaks down a matrix R in a smaller latent space. Mathematically, it decomposes R into two unitary matrices and a diagonal matrix:

$$U, \Sigma, V^t = SVD(R)$$

where R is the user ratings matrix, U is the user features matrix, Σ is the diagonal matrix of singular values and Vt is the product features matrix. U and Vt are orthogonal. U represents how much users like each feature and Vt represents how relevant each feature is to each product.

To get the lower rank approximation, we take these matrices and keep only the top 50 features, which we think of as the 50 most important underlying taste and preference vectors.

We used numpy's function to do the singular value decomposition. Having broken down the rating matrix, we were able to predict the rating of any user for any product. But this was not the purpose of this project. What we did instead was to analyze the Vt matrix. As it represents some features weights, we could use it as a measure of similarity between products. Therefore, for each item for which we wanted a recommendation we just had to compare the related column in the Vt matrix with any other column of this matrix. In order to do so we used cosine similarity:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

where A_i and B_i are components of vector A and B respectively.

This way we obtained a recommender system that was only based on the ratings.

4.4 Categories and Title TF-IDF

The idea with the fourth model was to extract as much information from text features to compare it with the data from the graph.

The metadata contains many thousands of unique categories that can be very precise. Here is an example: "|Books[283155]|Subjects[1000]|Romance[23]|Contemporary[13354]|General[524200]"

We used these as a first guess for the recommendation. Thinking about how a human would build a measure of similarity, it was clear for us that titles would be very valuable. One of the common way to find relevant similarities between text data is to use TF-IDF as defined below:

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears

This way we were able to get a sense of how similar two titles are.

In order to merge both information it was necessary to do some kind of weighting. As categories seemed to recommend very precise product and similarity based on term frequency seemed to yield very low scores (<0.2) we decided to weight the category with 0.5. This way we insured that if items from the same category existed, they were recommended. With this coefficient title similarity still has an important weight.

5. EVALUATION

The evaluation of recommender systems is a tricky issue. The ideal would be to test an algorithm on the e-commerce website in order to see how the click-through and conversion rates are affected. However, in our case, it was not possible and it is always better to have an idea of the performances of a strategy, before launching it live.

In the industry, recommendation systems are usually tested with A/B tests. For instance, we could give different recommendations to different sets of customers and ask them whether or not they think the recommendation is relevant. In our case we couldn't achieve this. Thus, we only tried to understand how we got our results and see how clever they were.

5.1 Evaluation of graph-based models

5.1.1 Evaluation of the second model

Using an additive model with an α equal to 0.6 and considering all neighbors, we got the following results, asking for 4 recommendations.

```
We are only able to provide  4  recommendations for product  1468  :
1468 [('337', 0.102522), ('1154', 0.099748), ('481', 0.073234), ('2452', 0.008597)]
2051 [('18586', 0.137511), ('36044', 0.071332), ('28411', 0.06539), ('3146', 0.053103), ('2050', 0.040044), ('3145', 0.009646)]
6714 [('768', 0.054756), ('181', 0.038053), ('17138', 0.029924), ('4194', 0.018971), ('11333', 0.015939), ('11334', 0.003952)]
We are only able to provide  4  recommendations for product  8857  :
8857 [('8854', 0.113462), ('28590', 0.107432), ('6960', 0.003186), ('28589', 0.0)]
```

Figure 3 Example of results of the second model (additivive with α=0.6), all neighbors

Using an additive model with an α equal to 0.6 and considering all out-neighbors, we got the following results, asking for 4 recommendations.

```
We are only able to provide  4  recommendations for product  1468  :
1468 [('337', 0.102522), ('1154', 0.099748), ('481', 0.073234), ('2452', 0.008597)]
We are only able to provide  4  recommendations for product  2051  :
2051 [('3146', 0.053103), ('2050', 0.040044), ('3145', 0.009646), ('1797', 0.002833)]
We are only able to provide  4  recommendations for product  6714  :
6714 [('768', 0.054756), ('181', 0.038053), ('11333', 0.015939), ('11334', 0.003952)]
We are only able to provide  4  recommendations for product  8857  :
8857 [('8854', 0.113462), ('28590', 0.107432), ('6960', 0.003106), ('28589', 0.0)]
```

Figure 4 Example of results of the second model (additive with α=0.6), out-neigbors

As we explained when we describe this model, many parameters can be tuned. For instance, we choose here an additive model with an α equal to 0.6 because we thought that the average review should have a higher weight than the helpfulness as we considered this latter difficult to evaluate. The best parameters of our model could be tuned using a kind of validation procedure thanks to A/B testing.

Different improvements of our model could be imagined. For instance, we could consider a bigger graph including not only mode co-purchased products but also products bought during the same day or week as well as products frequently viewed during the same day or week.… The weights of the edges being of course tuned so that the weights in the graph include a hierarchy: co-purchased products are closer in the graph than ones only bought during the same day but not together... The graph could be weighted by using many other elements: cosine similarities between the descriptions of two products, number of common motifs the two products are involved in…

5.1.2 Comparison with the first model

```
1468 ['1154', '481', '337', '2452']
2051 ['1797', '2050', '3145', '18589']
6714 ['4194', '17138', '768', '181']
8857 ['6960', '8854', '28589', '28590']
8109 ['7439', '13596', '279', '97']
```

Figure 5. Example of results of the second model

We can see that for the two algorithms using the network proprieties, the results are quite similar. Indeed, the recommended products are all neighbors and thus couldn't be really different. It could be interesting to compare it with the ranking of the most purchased product along with the studied item. These two methods are quite sure as they will always suggest related items. However, they might not bring much information. It is something that we couldn't really evaluate, as we were lacking some information. They might become handy with really big networks, when each product has many neighbors (which is the case in current Amazon co-purchasing network). We have to find clever ways to rank these neighbors because they are not all interesting, and these two strategies look promising to achieve this.

5.2 Evaluation of metadata-based models

The results obtained by these strategies 3 and 4 are totally different. Indeed it focuses on a totally different source of information. Here the assumption is that one customer will be interested in the same kind of items he has already bought. The previous reasoning was that we can fit the liking of a customer in a category with other similar customers.

Sadly, the main drawback of the SVD technique is that it is not interpretable.

For model 4, results look like what we expected. To explain we will look into two examples.

-Book 1: That's Not My Tractor: Its Engine Is Too Bumpy (Touchy-Feely Board Book)

-Recommendations (increasing similarity):

-The Book of Classic Board Games

-Tractor

-Robert Crowther's Most Amazing Hide-and-Seek 1-2-3 Numbers Book

-Peek-A-Boo Gingerbread House (Lift & Look Board Books)

The top two results are from the same category. The top one is also a board game as the 4th one. This is reassuring. The 4th was found only with title similarity. The third seems a bit off since it has a good score only because it is a one-word title and because this word is rare. However, it seems very difficult to actually rank and evaluate these recommendations as we don't have any expertise on those products. Moreover, recommender systems might want to add different degrees of originality in their results, and the quality of the results is very subjective.

6. CONCLUSIONS

As we could see, the main issue of the project was the evaluation of the results. It really limited the interpretation of the algorithms' performances. However, during this project, we were able to understand the importance of recommendation algorithms, what is at stakes and how they work. We came up with a few ideas that use the Amazon Co-Purchasing Network to recommend products to a user, only using an item he liked, purchased or even looked at. The approach using the motifs could also be thought as a strategy to analyze a network with more precision. We only looked at these motifs on the micro level. It could be interesting to see if by cutting the graph into communities, these motifs could be found on the macro level and what kind of proprieties it can involve.

A lot of improvements could be achieved. For example, these methods could be mixed together or with another algorithm that uses the customer's purchase history. More tests could also be undertaken with more data. We were really limited by the computation power of our computers. This could raise questions such as what approaches require the less computational time and are the most compatible with Big Data Frameworks such as Spark. Indeed, an e-commerce website like Amazon sells billions of products and the network would be gigantic. In our case, we were only able to work with approximately 2.5% of the nodes of the original network. Using all the nodes and more recent data, the results could be really different.

7. REFERENCES

[1] J. Leskovec, L. Adamic and B. Adamic. *The Dynamics of Viral Marketing*. *ACM Transactions on the Web (ACM TWEB)*, 1(1), 2007.

[2] G. Linden, B. Smith and J. York, *Amazon.com Recommendations, Item to Item Collaborative Filtering, .*

[3] A. Srivastave. 2010. *Motif of Analysis in the Amazon Product Co-Purchasing Network*

[4] S. Sodsee and M. Komkhao 2017. *Item Recommendation by Item Co-Purchasing Network and User Preference*

[5] K. Yang and L. Toni 2018. *Graph-Based Recommendation System*

[6] B. Shams and S. Haratizadeh. *Graph-Based Collaborative Ranking*

[7] J. Leskovec and A.Krevl, 2014, *SNAP Dataset: Large Network Dataset Collection, http://snap.stanford.edu/data*

[8] M Ben Fraj, 2018, *Graph-based recommendation engine for Amazon products,* https://towardsdatascience.com/graph-based-recommendation-engine-for-amazon-products-1a373e639263