Alisha Adam (aadam)
Matthew Pick (mpick)
Rohit Talreja (rtalreja)

CS 224W Final Report: Network-Based Product Recommendation Systems

0. Abstract

We developed a hybrid model for recommending products to customers (also called users) based on both similarity between products and similarity between users. Existing recommendation systems have a number of limitations; some systems, for example, treat user-user similarity as a binary relationship, ignoring the magnitude of similarity between users, while other systems assume that common opinions among users only matter if their reviews are positive. We designed an algorithm to address both of these flaws. Using a dataset of Amazon products and associated customer reviews, we built both a user graph (weighted edges represent the magnitude of similarity between users based on positive and negative reviews) and a product graph (edges represent products that have been purchased together). To create customized recommendations for a given user, our algorithm considered candidate products purchased by similar users, then scored candidates based on distance in the product graph. Our results showed that although co-purchases are far less relevant than we had hypothesized, we effectively captured similarity between users to predict user purchases with 50% accuracy.

1. Problem Definition

Today, the inventories of online retailers such as Amazon and Netflix far exceed those of traditional bricks-and-mortar retailers. Products that aren't popular enough to earn shelf space in physical stores account for a significant portion of online revenue; in the case of Amazon, more than 50% of book sales are generated by titles that aren't even available in most bookstores.[1] These products, collectively known as the Long Tail, represent a huge revenue opportunity. However, while media and marketing tend to expose consumers to more popular products, products in the Long Tail receive far less exposure. As a result, product recommender systems are instrumental in helping consumers find, among the millions of products in the tail, a select number of products that match their tastes. Research into recommendation systems is valuable for companies seeking to maximize revenue because new algorithms can improve relevance and/or novelty, thereby generating recommendations that consumers are more likely to purchase.

Current recommendation systems have two main drawbacks. Firstly, they tend to consider user-user similarity as a binary relationship without considering the magnitude of similarity between each pair of users. Secondly, they tend to ignore negative reviews, modeling similarity only by considering positive co-reviews. Our approach is to recommend products based not only on similarity between products but

[1] Anderson, C.

also the *magnitude* of similarity between users' reviews, factoring in both positive *and* negative sentiment.

2. Literature Review

2.1 Related Work

Existing recommendation systems fall into two main categories: collaborative filtering models, which leverage similarities between users, and content-based filtering, which relies on similarities between items. The former is used, for example, by Amazon and iTunes, while the latter is used by services such as Pandora.[2] Hybrid models have also been explored, which seek to provide more accurate recommendations by combining the two formulations. We examined the literature for these three types of systems to understand their implementations and tradeoffs.

There are two main approaches to collaborative filtering: memory-based and model-based filtering. Memory-based filtering approaches cluster users with common interests, then make predictions for a given user by averaging the preferences of other users in that cluster. On the other hand, model-based filtering uses past data about a single user to compute the expected value of a user's rating on unseen items, given the user's rating of other items.[3]

Rather than measuring similarity between users, content-based filtering approaches construct a product graph where weighted edges between product nodes represent the strength of an association between two products.[4] In the approach described by Zhou et al., recommendations were generated for a given user *u* by constructing a candidate set of products that have been co-purchased with items that *u* already owns. The top *r* items with the highest edge weights are then returned as recommendations. This approach is simple, but it was shown to generate effective recommendations in a dense product graph.

Finally, hybrid systems combine these two approaches with the goal of optimizing the tradeoff between accuracy and novelty of recommendations. One such system is GraphRec, described by Lee and Lee.[5] Starting with a user-item ratings matrix (N users x M items), the authors created a weighted M x M adjacency matrix of positively co-rated items, then recommended products that were repeatedly co-rated with items the user had purchased. Compared to other recommender systems, GraphRec was shown to generate more novel recommendations without a significant tradeoff in accuracy, particularly for users with sufficiently complete profiles.

---

[2] Zahra, S.
[3] Sarwar et al.
[4] Zhou et al.
[5] Lee and Lee

2.2 Opportunities for Improvement

As seen from this discussion, existing algorithms focus primarily on *positive* signals. Zhou et al. assume that purchasing or acquiring an item is synonymous with liking that item. Lee and Lee use only positive ratings to identify similar users. In both cases, incorporating *negative* information would allow us to develop a more nuanced model. In particular, we improve upon the systems discussed above in two key ways:

Firstly, we extend Zhou et al.'s approach by adding a dimension for product satisfaction. Zhou et al. naively assume that users are unequivocally satisfied with all products they purchase. In reality, of course, this is not the case. People frequently regret or dislike items after making a purchase, as evident from negative reviews on Amazon. Incorporating the sentiment of a review (e.g. # of stars, positive or negative) allows us to model a more nuanced relationship between users and the products they purchase.

Secondly, we extend Lee and Lee's approach of modeling similarity between users by using the full spectrum of reviews, rather than focusing exclusively on positive reviews. When forming the adjacency matrix A, Lee and Lee discard negative reviews, implicitly assuming that negative reviews contain no information about user-user similarity. However, we contend that negative ratings can also provide important information about user-user similarity. Consider users *a*, *b*, and *c* who all highly review item *x*. Suppose *c* highly reviews item *y*, while *a* and *b* both negatively review *y*. According to Lee and Lee's GraphRec formulation, *a*, *b*, and *c* are all equally similar, while in reality *a* and *b* are probably more similar because they have the same positive *and* negative tastes. Thus, when constructing a graph of users, we link each user *m* to a set of other users, *U*, based on the similarity of the positive *and* negative reviews that they have in common. In addition to capturing a new dimension of similarity, this approach allows us to create connections between more users, thereby giving us a larger subgraph from which to generate recommendations for each user.

3. Dataset

3.1 Overview

We decided to work on a subset of the Amazon dataset curated by Leskovec, Adamic and Adamic[6] since it contains the most complete information about products and associated reviews. The dataset identifies products that are frequently purchased together, as well as reviews for each product associated with the ID of the user who wrote them.

The products come from four major categories - books, DVDs, music CDs, and videos. Note that these product categories lend themselves to co-purchasing more so than others categories like appliances, furniture, and household items. For example, when purchasing books, customers might choose to

[6] Leskovec, Adamic and Adamic

purchase an entire series (e.g. Harry Potter books 1 - 7). In other categories, this is less likely to occur; for instance, customers purchasing kitchen appliances probably don't buy more than one toaster.

Product Graph	
Metric	**Value**
Nodes	35,852
Edges	65,547
Avg. Degree	2

User Graph	
Metric	**Value**
Nodes	90,370
Edges	42,400,000
Avg. Degree	464

Fig. 1 Dataset Statistics

As seen in Figure 1, the user graph was extremely dense; each node had, on average, more than 400 neighbors. In contrast, the product graph was extremely sparse, with an average degree of just 2. This was largely due to the nature of our dataset, which created a hub-and-spokes graph structure. In the dataset, product entries were generally associated with 5 co-purchased products (it was possible to have fewer co-purchased products, but most entries had exactly 5). In general, however, the co-purchased products did not appear as independent entries in the dataset. Thus, while products that were independently listed in the dataset had approximately 5 neighbors, products that were listed only as co-purchases tended to have only 1 edge. This resulted in a hub-and-spokes structure, visualized in Figure 2 below, with a low average degree.

3.2 Data Parsing

The data file lists products in the following format:

ASIN: [Amazon identification string of the product]
title: [product name]
Salesrank: [Amazon Salesrank metric]
similar: [ASIN of up to 5 products that are co-purchased with the current product]
categories: [several categorizations of the product, eg. books->subjects->fantasy]
reviews:
[list of reviews including customer ID, rating, and publish date]

The data parser, along with the rest of the program, was written in Python to optimize error handling and ease of programming. For each product in the data file, we extracted the list of similar products and customer reviews in order to construct our graphs (discussed in detail in section 5.2).

4. Baseline Model

To benchmark the accuracy of our extended recommendation algorithm, we first implemented the recommendation algorithm described in the Zhou et al. paper. Specifically, we generated a network of products in which the weight of a directed edge from product *j* to product *i* represented the strength of recommending product *i* to a user who purchased product *j*. To calculate these recommendation weights,

we constructed an adjacency matrix A, where $A_{il} = 1$ when object i is owned by user l. The weight w_{ij} is assigned to the edge from product j to i as follows:

$$w_{ij} = \frac{1}{k(o_j)} \sum_{l=1}^{m} \frac{a_{il} a_{jl}}{k(u_l)},$$

$$\text{where } k(o_j) = \sum_{i=1}^{n} a_{ji} \text{ and } k(u_l) = \sum_{i=1}^{m} a_{il}$$

To generate recommendations for user u, we construct a candidate set of products that have been co-purchased with items that u already owns. The top r items with the highest edge weights are then returned as recommendations. Since the Zhou et al. paper did not provide a single value for r (the size of the recommendation set), we evaluated the baseline accuracy on a range of r-values using the test methodology described in section 6 below (results are displayed in Figure 4).

5. Our Hybrid Recommendation Model

In this section, we will discuss our recommendation system, a hybrid model using both collaborative filtering and content-based filtering (i.e. taking into account both product similarity and user similarity).

5.1 Assumptions

When generating recommendations, we make three key assumptions. Firstly, we assume that users' reviews are indicative of their actual preferences. Obviously, users don't leave a review for every item that they purchase; however, we assume that the reviews they do write will accurately reflect their tastes. This assumption is necessary because our goal is to recommend items that users will actually want to purchase, not just to recommend items that are most consistent with their past reviews. Secondly, we assume that similarity between two users' reviews is an accurate measure of similarity between the two users' preferences. This assumption follows from the previous one: if reviews are an accurate indication of preferences, then similar reviews should indicate similar preferences. Finally, we assume that user-user similarity among the sample of reviews that we've seen (i.e. the data used to create the graph) will generalize to capture similarity among users in the test set. We make this assumption because we believe that our test set follows the same distribution as our training data.

5.2 Constructing the graphs

5.2.1 Co-purchased Product Graph

We constructed a graph of co-purchased products, G_p, where each product from our dataset is a node in the graph. For each node u, edges exist between u and the top five products with which u is most commonly co-purchased. This resulted in a hub-and-spokes structure, as seen in Figure 2.

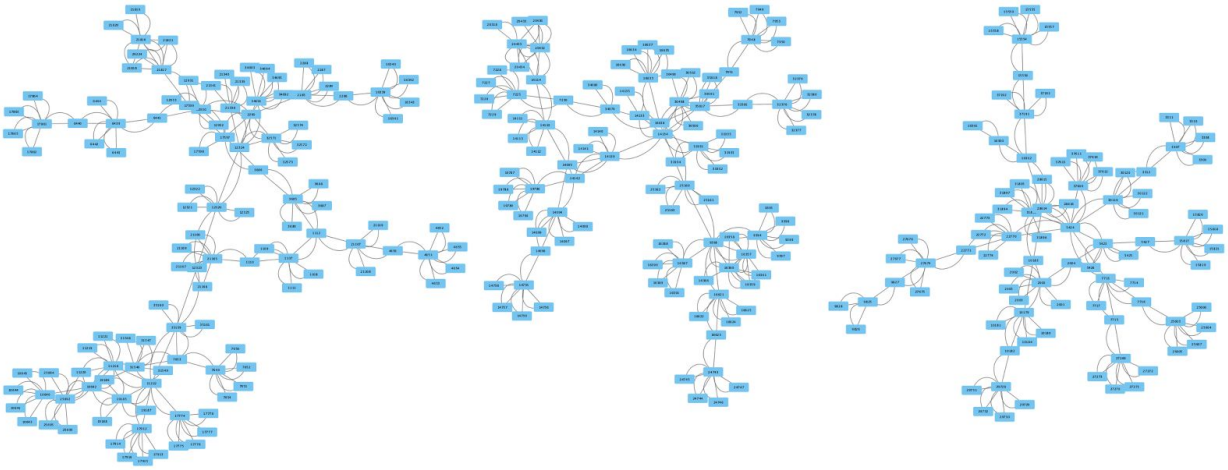

Fig. 2 Subset of the Product Graph, which exhibits a hub-and-spokes structure

5.2.2 User Graph

Next, we constructed a graph of users, G_u, where each user from our dataset is a node in the graph, according to the following procedure:

1. Add edges between all nodes *a* and *b* who reviewed one or more of the same products
2. For edge (*a*, *b*), define the weight w_{ab} as:

$$w_{ab} = \sum_{\substack{all\ co-reviewed \\ products\ (p)}} \begin{cases} 1\ if\ |R_a - R_b| < 2 \\ -1\ if\ |R_a - R_b| \geq 2 \end{cases}$$

where R_a is *a*'s rating for *p* and R_b is *b*'s rating for *p*

This means that each review on which *a* and *b* agree (either both positive or both negative) contributes +1 to the edge weight, and each review on which *a* and *b* disagree contributes -1. Note that under this measure, agreement and disagreement "cancel out": if *a* and *b* agree on 3 products and disagree on 2, then $w_{ab} = 1$ (i.e. we weight the edge as one similar product review)

3. Remove all edges with weight ≤ 0, since we only want to consider users who agree on their reviews.

Thus, in the final graph, edges exist between users who are similar based on common positive and/or negative reviews. The weight of each edge corresponds to the degree of similarity between the two nodes.

5.3 Generating Recommendations

Our algorithm to generate recommendations for a given user, *u*, is as follows:

1) Define the purchased set P as the set of products that *u* has purchased.
2) Define the candidate set C as user-product pairs, (*v*, *x*), where *v* is a neighbor of *u* and *x* is a product that *v* has purchased but *u* has not purchased

3) Score each candidate (v, x) in C, as follows:

$$score(v,x) = \frac{similarity(u,v)}{distance(x,P)}$$

where *similarity(u, v)* is the weight of the edge between u and v in the user graph and *distance(x, P)* is the length of the shortest path between x and any product in the purchased set.

4) Return a recommendation set, R, consisting of the top r candidates (see Figure 4 for the effect of varying r on prediction accuracy).

We also implemented a modified score function that re-weighted the contribution of path length (distance between two products) to the score. From the candidate set C, consider all products x purchased by a given neighbor v. Instead of computing the full set of paths for each new x', we kept track of the length l of the shortest path so far between any product x that we already scored and P (l is initialized to an arbitrary large number). When scoring additional candidates, we cut off the graph search at distance l (so products that are more than l hops away from P are assigned distance l). Keeping track of the best l so far for each neighbor ensures that we accurately score top candidates (i.e. those that are close to the purchased set P) while optimizing runtime by avoiding unnecessarily long path searches (for products that are far from P and thus unlikely to be good recommendations). Furthermore, we hypothesized that after a certain distance, the association between products becomes effectively random. If no products exist within l hops of the purchase set, we decided that recommendations should be based purely on user-user similarity. Since products outside the cutoff are assigned the same distance, the algorithm would return a random product purchased by v.

6. Evaluation

From the dataset, we created evaluation sets of 100 test purchases as follows:

1) Pick a user, u, at random and choose one of their purchases, p
2) Remove that purchase from both the user and product graphs

For each user-purchase pair in the evaluation set, we generate a recommendation set S consisting of r products. We define accuracy as the fraction of recommendation sets S that contain the corresponding test purchase p. For each algorithm, we ran this evaluation metric several times to get the average accuracy.

Since running our algorithm on the full dataset (~500,000 products) would require immense computational resources, we also had to select a subset of the data on which to evaluate our approach. We ran the baseline algorithm and our algorithm (with the original score function and the cutoff score function) on different sized datasets with $r = 10$, as shown in Figure 3, to identify trends in performance as we increase dataset size.

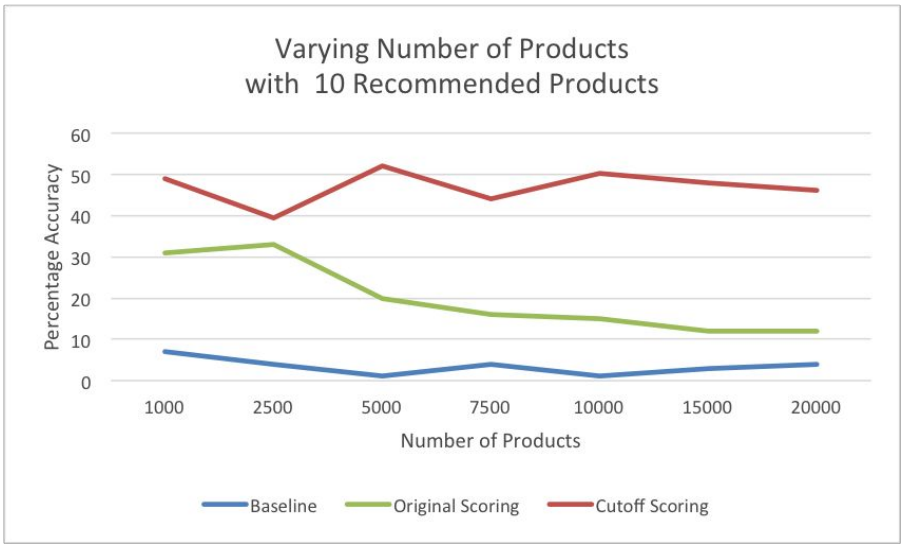

Fig 3. Accuracy on varying sized datasets

The resulting plot shows that the accuracy of all three methods becomes fairly constant for datasets greater than 5,000 products. Thus, we chose to evaluate our results on a dataset containing 10,000 products because we felt it optimized the tradeoff between model accuracy and runtime. Based on the trend in Figure 3, we expect our results on this subset of the data to scale to larger datasets without a significant tradeoff in accuracy.

7. Results

The results of running each algorithm on the 10,000 product dataset are shown below.

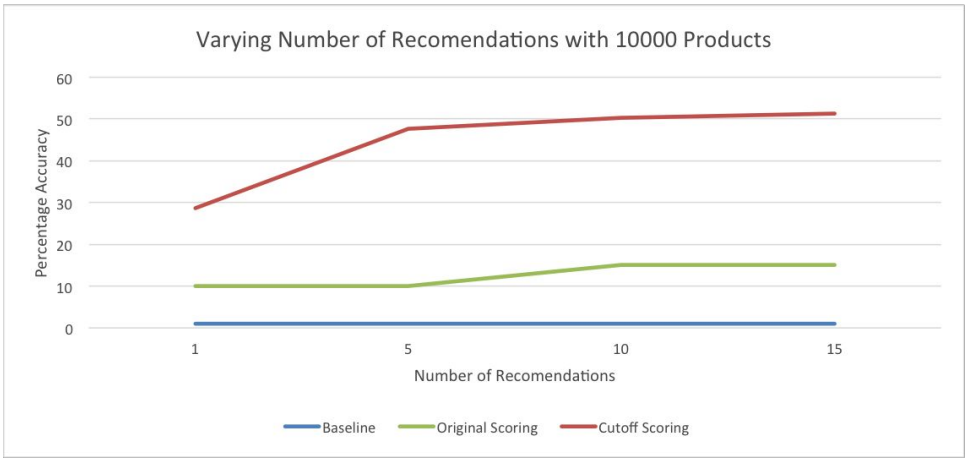

Fig 4. Performance of the algorithms

As seen in Figure 4, regardless of the size of the recommendation set, both versions of our algorithm perform significantly better than the baseline approach. Our original score function achieves between 10

and 15% accuracy, while the cutoff function converges at approximately 50% when making 5 or more product recommendations . This is a dramatic improvement over the baseline, which consistently achieves accuracy of 1 – 3%.

At first glance, this performance seems unexpectedly poor, particularly for the baseline; however, the Amazon dataset forces all three algorithms to model behavior that is inconsistent with true purchasing patterns. Since we have limited knowledge of product equivalence, the algorithm is evaluated on its ability to find the *exact* purchase made by the test user (ex. KitchenAid KMT222 2-slice toaster) rather than a *close enough* recommendation (ex. toaster). The success of real world product recommendations would instead evaluate the suggestive power of a recommendation, namely if a purchase of a similar product was made after viewing the recommendation. Furthermore, Zhou's algorithm performs well on dense graphs but our data file is missing a significant portion of the co-purchased product nodes; thus, many products were left with few edges. Finally, many users in this file purchased few products and thus did not provide enough information on prior purchases to yield accurate recommendations. Thus, when run in a real world situation, we believe that all three algorithms would generate more accurate recommendations than these results might imply. Rather than viewing the accuracy scores as standalone percentages, these values should be evaluated relative to each other, since all three algorithms are all impacted to the same degree. Therefore, we believe that the improvement of our method over the baseline algorithm reflects a true gain in accuracy.

There are two conclusions that can be drawn from these results. Firstly, the accuracy of our algorithm levels off as the size of the recommendation set increases. This demonstrates that both of our score functions generate fairly good rankings; if the test purchase is in our candidate set, it is consistently ranked in the top 5. Furthermore, these results imply that the optimal number of recommendations for our algorithm is 5. This is important because providing too many recommendations in real world applications may lead to oversaturation, reducing the ability of each recommendation to influence purchasing behavior. Thus, it is significant that our algorithm can achieve nearly 50% accuracy by recommending just 5 products.

Secondly, we observe that the cutoff score function generated significantly better recommendations than the original score function. This reveals that one of our initial hypotheses is incorrect. We initially believed that proximity between products x and y in the product graph would imply that y is a good recommendation for users who purchased x (and vice versa); however, the cutoff algorithm reduces the weight of distance in the scoring function, particularly for larger distances, which resulted in significantly higher accuracy. Furthermore, initializing the cutoff to very small values yields almost the same accuracy, which shows that distance is far less important than user-user similarity when generating recommendations. Possible explanations for this observation are discussed in the next section.

In summary, our results show that although our initial hypothesis about product co-purchases was incorrect, we were able to achieve remarkable accuracy by modeling similarity between users. Using the cutoff score function, we could predict test purchases with 30% accuracy by generating just a single recommendation. By increasing the number of recommendations generated to 5, we could correctly predict the test purchase 50% of the time. Recall that our accuracy metric was, in fact, more strict than

most real-world applications, since it measured our ability to find *exact* purchases made by the test user rather than close recommendations. These results show that our model of user similarity, which measured the magnitude of similarity between users based on common positive and negative reviews, is a promising development for future recommendation systems based on collaborative filtering.

8. Further work

As noted above, the cutoff score function generated significantly higher accuracy than our original score function. There are three possible explanations for this observation. Firstly, as noted in section 3.1, the product graph is extremely sparse (average degree is approximately 2). Thus, we may not have enough information about relationships between products to effectively use product similarity when generating recommendations. To test this, we could either run the algorithm on a dense subgraph of the current dataset or find a different dataset with more product similarity data. Secondly, "similar" products in the Amazon dataset are products that are commonly co-purchased. If co-purchased items are those that are purchased in the same transaction, then these relationships may not be a good source of recommendations. Recommendations should capture users' preferences across transactions, not within one particular transaction. To test this, we could use a different measure of product similarity, such as purchase history (i.e. users who buy *x* have bought *y* in the past), and see if proximity in the new product graph improves recommendations. Finally, it's possible that product-product similarity, as used in our algorithm, simply isn't relevant when making recommendations. Before jumping to this conclusion, however, we would like to be able to test the first two options with a more dense product graph and a different measure of product similarity.

It's also possible that the cutoff function generated better results because our original score function was suboptimal. Another area of future research could be to test our algorithm with different score functions, such as:

- linear penalty for increasing path length so that user similarity and distance are equally weighted
- exponential penalty for increasing path length to penalize far away items even more
- logarithmic scaling of the similarity score between users so that similarity scores are underemphasized compared to the current model
- ignore distance entirely, as the results of the cutoff algorithm suggest that distance is less relevant than we thought

Given the results with the cutoff scoring function, we would start with the last option since we were able to boost accuracy by decreasing the importance of distance.

Another area for further work is evaluation. Since we tested the algorithm from Zhou et al. using our evaluation methodology, we would also like to test our algorithm on their methodology. It would be important to assess how similar the problems actually are and whether the 1-3% accuracy of their algorithm was due to mismatch between fundamental assumptions in algorithm design. Furthermore, it's possible that the scarcity of the product graph negatively impacted the Zhou et al. algorithm. We would like to compare our method to other systems that emphasize user similarity, since our user graph is much more dense.

Lastly, we would have liked to be able to run our algorithm on the full 550,000 product dataset; however, this would have taken multiple days on an extremely high-powered computer. It would be ideal to find ways of partitioning the dataset so that processing could be parallelized. Alternatively, we could incorporate heuristics to avoid computing the distance between hundreds of pairs of products.

References

Anderson, Chris. “The Long Tail.” *Wired* (2004).

Lee, K. and Lee, K. “Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items.” *Expert Systems with Applications* 42.10 (2015): 4851-4858.

J. Leskovec, L. Adamic and B. Adamic. “The Dynamics of Viral Marketing.” *ACM Transactions on the Web* (ACM TWEB), 1.1 (2007).

Liben‐Nowell, David, and Jon Kleinberg. "The link‐prediction problem for social networks." *Journal of the American society for information science and technology* 58.7 (2007): 1019-1031.

Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. "Item Based Collaborative Filtering Recommendation Algorithms." *ACM*: 285-95 (2001).

Zahra, S., et al. “Novel centroid selection approaches for KMeans-clustering based recommender systems.” *Information Sciences.* 320 (2015): 156-189.

Zhou, Tao, et al. "Effect of initial configuration on network-based recommendation." *EPL (Europhysics Letters)* 81.5 (2008): 58004.