# Graph Based Amazon Products Recommendation System

## PROJECT REPORT

**Social and Information Networks (CSE3021)**

**Submitted By:-**

**Mitarth Jain 17BCE0765**

**Submitted to:-**

**Prof. Manjula R**

**Slot- C1+TC1**


VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

---

November,2019

# ACKNOWLEDGEMENTS

A deepest gratitude and sincere thanks to Prof. Manjula R in helping us complete our Project with several learning outcomes. We feel deeply obliged to thank the SCOPE (School of Computer Science and Engineering) Department and the VIT University for their services rendered and for giving us an opportunity to carry out our studies at the University.

Mitarth Jain(17BCE0765)

# INTRODUCTION

In this project, we are going to study about recommendation systems. Recommendation systems are typically used by companies, especially e-commerce companies like Amazon.com, to help users discover items they might not have found by themselves and promote sales to potential customers. A good recommendation system can provide customers with the most relevant products. This is a highly-targeted approach which can generate high conversion rate and make it very effective and smooth to do advertisements. So the problem we are trying to study here is that, how to build effective recommendation systems that can predict products that customers like the most and have the most potential to buy.

## PRIOR WORK

There has been a lot of work done in this field. For example, one very popular algorithm is Collaborative Filtering. One type of collaborative filtering is user-based collaborative filtering, which starts by finding a set of customers who have purchased and rated similar items with the target users purchasing history. The algorithm aggregates items from these similar customers, and uses the ratings from other similar users to predict the ratings from this user. Another type of collaborative filtering is item-based collaborative filtering, which was first brought up by Amazon and focuses on finding similar items instead of similar 1 customers. For each of the users purchased and rated items, the algorithm attempts to find similar items. It then aggregates these similar items and recommends them. There are also other algorithms that try to exploit graph structures to predict links or ratings. Random walks algorithms could be used in predicting links in complex graphs in a very efficient manner. And also, if we model the user and product graph as a bipartite graph, then it is also feasible to use Bipartite Projection algorithm to calculate the relevance between two customers. So the predicted rating is essentially based on the other relevant customers' ratings. In later sections of this paper, we will introduce three models and algorithms which are derived from the prior work mentioned above with application-specific improvements.

**Data Set Information**

This project will use Amazon Meta-Data Set maintained on the Stanford Network Analysis Project (SNAP) website. This data set is comprised of product and review metadata on 548,552 different products. You can view the data by double-clicking on the file amazon-meta.txt that has been included in SocialNetworkAnalysis.zip file. The following information is available for each product in the data set.

## Data Format

```
Id:   15
ASIN: 1559362022
  title: Wake Up and Smell the Coffee
  group: Book
  salesrank: 518927
  similar: 5  1559360968  1559361247  1559360828  1559361018  0743214552
  categories: 3
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Drama[2159]|United States[2160]
   |Books[283155]|Subjects[1000]|Arts & Photography[1]|Performing Arts[521000]|Theater[2154
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Authors, A-Z[70021]|( B )[70023]|
  reviews: total: 8  downloaded: 8  avg rating: 4
    2002-5-13  cutomer: A2IGOA66Y6O8TQ  rating: 5  votes:   3  helpful:   2
    2002-6-17  cutomer: A2OIN4AUH84KNE  rating: 5  votes:   2  helpful:   1
    2003-1-2  cutomer: A2HN382JNT1CIU  rating: 1  votes:   6  helpful:   1
    2003-6-7  cutomer: A2FDJ79LDU4O18  rating: 4  votes:   1  helpful:   1
    2003-6-27  cutomer: A39QMV9ZKRJXO5  rating: 4  votes:   1  helpful:   1
    2004-2-17  cutomer:  AUUVMSTQ1TXDI  rating: 1  votes:   2  helpful:   0
    2004-2-24  cutomer: A2C5K0QTLL9UAT  rating: 5  votes:   2  helpful:   2
    2004-10-13  cutomer:  A5XYF0Z3UH4HB  rating: 5  votes:   1  helpful:   1
```

- Id: Product id (number 0, …, 548551)
- ASIN: Amazon Standard Identification Number.
  The Amazon Standard Identification Number (ASIN) is a 10-character alphanumeric unique identifier assigned by Amazon.com for product identification.
- title: Name/title of the product
- group: Product group. The product group can be Book, DVD, Video or Music.
- Salesrank: The Amazon sales rank represents how a product is selling in comparison to other products in its primary category. The lower the rank, the better a product is selling.
- similar: ASINs of co-purchased products (people who buy X also buy Y)
- categories: Location in product category hierarchy to which the product belongs (separated by |, category id in [])
- reviews: Product review information: total number of reviews, average rating, as well as individual customer review information including time, user id, rating, total number of votes on the review, total number of helpfulness votes (how many people found the review to be helpful)

## Software Requirements

- Python concepts to read, manipulate and to prepare for analysis.
- Social Network Analysis concepts to build and analyze Graphs.

**Python Packages Used : -**

- **Nltk -** The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.
- **Networkx -** NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
- **Matplotlib -** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

## Preprocessing

Some preprocessing of the dataset is required to read the file as Relational Data Base Management System (RDBMS) to use ASIN as the key and the others as the metadata associated with ASIN.

- Id: same as “Id” in amazon-meta.txt
- ASIN: same as “ASIN” in amazon -meta.txt
- Title: same as “title” in amazon-meta.txt
- Categories: a transformed version of “categories” in amazon-meta.txt. Essentially, all categories associated with the ASIN are concatenated, and are then subject to the following Text Preprocessing steps: lowercase, stemming, remove digit/punctuation, remove stop words, retain only unique words. The resulting list of words is then placed into “Categories”.
- Copurchased: a transformed version of “similar” in amazon-meta.txt. Essentially, the copurchased ASINs in the “similar” field are filtered down to only those ASINs that have metadata associated with it. The resulting list of ASINs is then placed into “Copurchased”.
- SalesRank: same as “salesrank” in amazon-meta.txt
- TotalReviews: same as total number of reviews under “reviews” in amazon-meta.txt
- AvgRating: same as average rating under “reviews” in amazon-meta.txt

```
(Id, ASIN, Title, Categories, Group, Copurchased, SalesRank, TotalReviews, AvgRating, DegreeCentrality, ClusteringCoeff) = \
    ("", "", "", "", "", "", 0, 0, 0.0, 0, 0.0)
for line in fhr:
    line = line.strip()

    if(line.startswith("Id")):
        Id = line[3:].strip()

    elif(line.startswith("ASIN")):
        ASIN = line[5:].strip()

    elif(line.startswith("title")):
        Title = line[6:].strip()
        Title = ' '.join(Title.split())

    elif(line.startswith("group")):
        Group = line[6:].strip()

    elif(line.startswith("salesrank")):
        SalesRank = line[10:].strip()

    elif(line.startswith("similar")):
        ls = line.split()
        Copurchased = ' '.join([c for c in ls[2:]])

    elif(line.startswith("categories")):
        ls = line.split()
        Categories = ' '.join((fhr.readline()).lower() for i in range(int(ls[1].strip())))
        Categories = re.compile('[%s]' % re.escape(string.digits + string.punctuation)).sub(' ', Categories)
        Categories = ' '.join(set(Categories.split()) - set(stopwords.words("english")))
        Categories = ' '.join(stem(word) for word in Categories.split())

    elif(line.startswith("reviews")):
        ls = line.split()
        TotalReviews = ls[2].strip()
        AvgRating = ls[7].strip()

    elif(line==""):
        try:
            MetaData={}
            if(ASIN != ""):
                amazonProducts[ASIN] = MetaData
            MetaData['Id'] = Id
            MetaData['Title'] = Title
            MetaData['Categories'] = ' '.join(set(Categories.split()))
            MetaData['Group'] = Group
            MetaData['Copurchased'] = Copurchased
            MetaData['SalesRank'] = int(SalesRank)
            MetaData['TotalReviews'] = int(TotalReviews)
            MetaData['AvgRating'] = float(AvgRating)
            MetaData['DegreeCentrality'] = DegreeCentrality
            MetaData['ClusteringCoeff'] = ClusteringCoeff
        except NameError:
            continue
        (Id, ASIN, Title, Categories, Group, Copurchased, SalesRank, TotalReviews, AvgRating, DegreeCentrality, ClusteringCoeff) = \
            ("", "", "", "", "", "", 0, 0, 0.0, 0, 0.0)
fhr.close()
```

Now we can filter amazonProducts Dictionary down to only Group=Book, and write filtered data to amazonBooks Dictionary.

```
amazonBooks = {}
for asin,metadata in amazonProducts.items():
    if (metadata['Group']=='Book'):
        amazonBooks[asin] = amazonProducts[asin]

for asin, metadata in amazonBooks.items():
    amazonBooks[asin]['Copurchased'] = \
        ' '.join([cp for cp in metadata['Copurchased'].split() \
                  if cp in amazonBooks.keys()])
```

Now we can use the co-purchase data in amazonBooks Dictionary to create the copurchaseGraph Structure as follows:

- Nodes: the ASINs are Nodes in the Graph
- Edges: an Edge exists between two Nodes (ASINs) if the two ASINs were co-purchased
- Edge Weight (based on Category Similarity): since we are attempting to make book recommendations based on co-purchase information, it would be nice to have some measure of Similarity for each ASIN (Node) pair that was co-purchased (existence of Edge between the Nodes). We can then use the Similarity measure as the Edge Weight between the Node pair that was co-purchased. We can potentially create such a Similarity measure by using the "Categories" data, where the Similarity measure between any two ASINs that were co-purchased is calculated as follows:
- Similarity = (Number of words that are common between Categories of connected Nodes) / (Total Number of words in both Categories of

connectedNodes)

The Similarity ranges from 0 (most dissimilar) to 1 (most similar).

```
copurchaseGraph = networkx.Graph()
for asin,metadata in amazonBooks.items():
    copurchaseGraph.add_node(asin)
    for a in metadata['Copurchased'].split():
        copurchaseGraph.add_node(a.strip())
        similarity = 0
        n1 = set((amazonBooks[asin]['Categories']).split())
        n2 = set((amazonBooks[a]['Categories']).split())
        n1In2 = n1 & n2
        n1Un2 = n1 | n2
        if(len(n1Un2)) > 0:
            similarity = round(len(n1In2)/len(n1Un2), 2)
        copurchaseGraph.add_edge(asin, a.strip(), weight=similarity)
```

We also have to add the graph-related measures for each ASIN to the amazonBooks Dictionary:

- **DegreeCentrality:** associated with each Node (ASIN)
- **Clustering Coeff:** associated with each Node (ASIN)

```
dc = networkx.degree(copurchaseGraph)
for asin in networkx.nodes(copurchaseGraph):
    metadata = amazonBooks[asin]
    metadata['DegreeCentrality'] = int(dc[asin])
    ego = networkx.ego_graph(copurchaseGraph, asin, radius=1)
    metadata['ClusteringCoeff'] = round(networkx.average_clustering(ego), 2)
    amazonBooks[asin] = metadata
```

## Recommendation

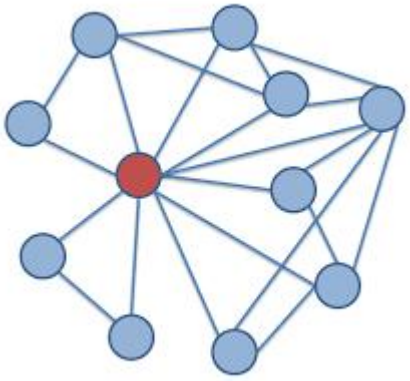
We assume a User who is looking to purchase a book with ASIN = 0875421210, now we need to make recommendations to the user based on the book copurchased data we have, We examine the metadata associated with that particular book that includes Title, SalesRank, TotalReviews, AvgRating, Degree Centrality and Clustering Coefficient.

ASIN = 0875421210
Title = Earth Power: Techniques of Natural Magic (Llewellyn's Practical Magick)
SalesRank = 100261
TotalReviews = 47
AvgRating = 4.5
DegreeCentrality = 14
ClusteringCoeff = 0.67

We notice that this book has a Degree Centrality of 14 which means 14 other books were copurchased with this book by other users and for many other books in our dataset the degree centrality is more than 100 or 200, it will be difficult to recommend books using only Degree Centrality measure. To improve this process of recommendation we use the concept of Ego Networks.

## Ego Networks

In such a network, we take a focal node call it the “ego”, and the nodes that have edges with the ego are termed as the “alters”. Each alter of an ego network forms their own ego network. The intertwining of all the ego networks forms the social network.



Here in above figure the red node is the ego node and all others are alters, for our dataset we will obtain a depth-1 ego network which will consist of book purchased ASIN=0875421210 as the ego node and get the books that have been co-purchased with the purchased ASIN in the past and assign the resulting Graph as pAsinEgoGraph.

```
n = pAsin
ego = networkx.ego_graph(copurchaseGraph, n, radius=1)
pAsinEgoGraph = networkx.Graph(ego)
pos = networkx.layout.spring_layout(pAsinEgoGraph)
M = pAsinEgoGraph.number_of_edges()
nodes = networkx.draw_networkx_nodes(pAsinEgoGraph, pos, node_size=50, node_color='blue')
edges = networkx.draw_networkx_edges(pAsinEgoGraph, pos, node_size=50, edge_cmap=plt.cm.Blues, width=2, alpha=0.1)
ax = plt.gca()
ax.set_axis_off()
plt.title('Depth-1 Ego Network')
plt.figure(0)
plt.show()
```

Now we need to filter down the obtained graph to most similar books, using Island method which is used to split the network components using some threshold value to differentiate between the components, by applying this method on pAsinEgoGraph to only retain edges with threshold > = 0.5 and assign the resulting graph as pAsinEgoTrimGraph.

```
threshold = 0.5
pAsinEgoTrimGraph = networkx.Graph()
for f,t,e in pAsinEgoGraph.edges(data=True):
    if e['weight'] >= threshold:
        pAsinEgoTrimGraph.add_edge(f,t, weight=e['weight'])
pos = networkx.layout.spring_layout(pAsinEgoTrimGraph)
M = pAsinEgoTrimGraph.number_of_edges()
nodes = networkx.draw_networkx_nodes(pAsinEgoTrimGraph, pos, node_size=50, node_color='blue', label=True)
edges = networkx.draw_networkx_edges(pAsinEgoTrimGraph, pos, node_size=50, edge_cmap=plt.cm.Blues, width=2, alpha=0.1)
ax = plt.gca()
ax.set_axis_off()
plt.title('Depth-1 Ego Network (After applying Island Method using threshold of 0.5)')
plt.figure(1)
plt.show()
```

The alters or the neighbours of this graph will be our recommendations stored in pAsinNeighbours.

```
pAsinNeighbours = pAsinEgoTrimGraph.neighbors(pAsin)
```

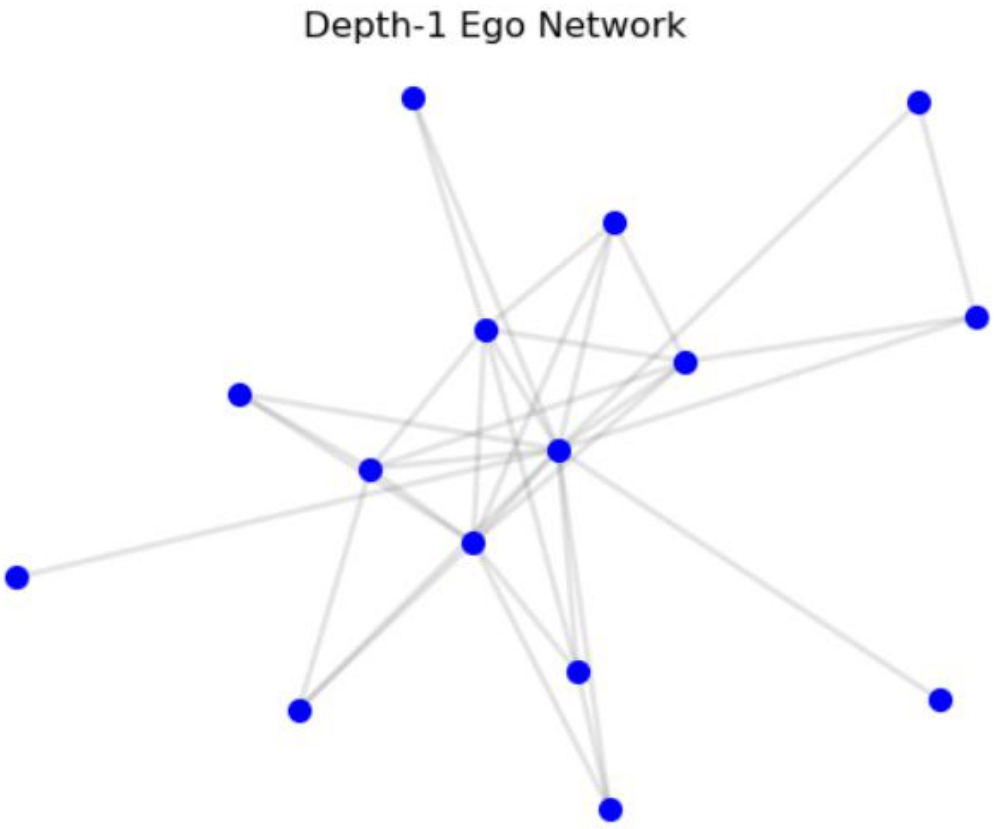
## Results

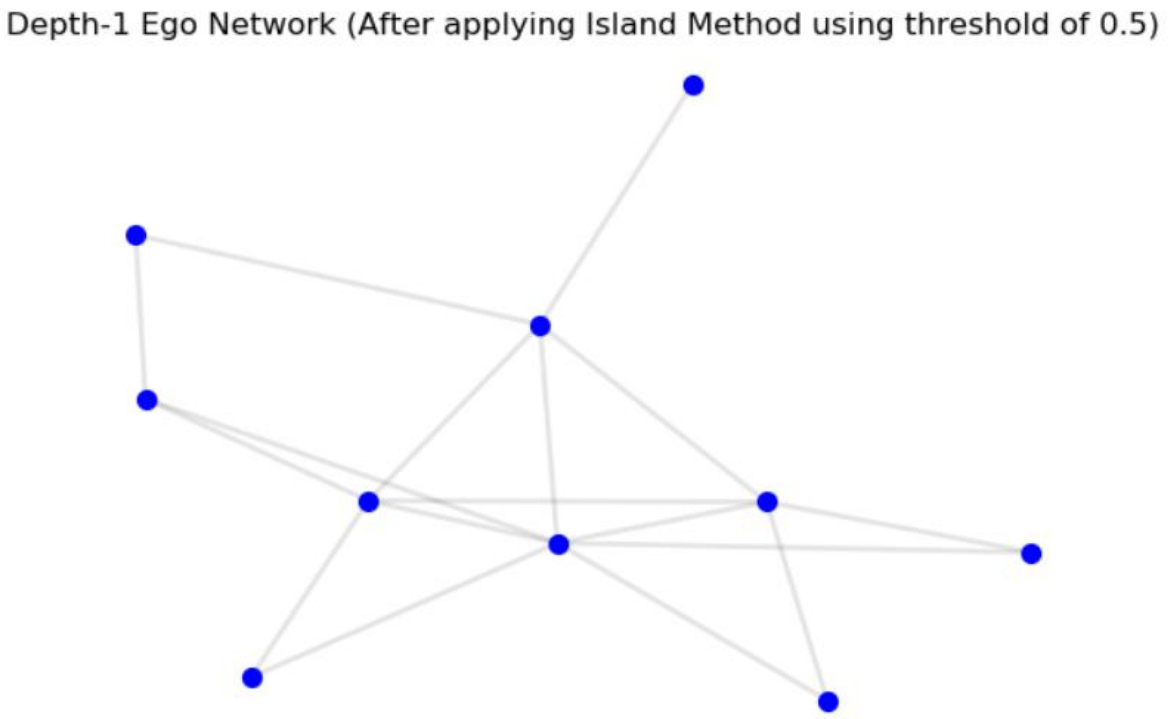
To make the Top Five book recommendations based on the measures associated with neighbours in pAsinNeighbours: SalesRank, AvgRating, TotalReviews, Degree Centrality and Clustering Coefficient the most appropriate will be the combination of AvgRating and TotalReviews which will provide the best top five books for recommendations to the User.

```
AsMeta = []
for asin in pAsinNeighbours:
    ASIN = asin
    Title = amazonBooks[asin]['Title']
    SalesRank = amazonBooks[asin]['SalesRank']
    TotalReviews = amazonBooks[asin]['TotalReviews']
    AvgRating = amazonBooks[asin]['AvgRating']
    DegreeCentrality = amazonBooks[asin]['DegreeCentrality']
    ClusteringCoeff = amazonBooks[asin]['ClusteringCoeff']
    AsMeta.append((ASIN, Title, SalesRank, TotalReviews, AvgRating, DegreeCentrality, ClusteringCoeff))


Top5_byAvgRating_then_byTotalReviews = sorted(AsMeta, key=lambda x: (x[4], x[3]), reverse=True)[:5]

f=open("o.csv","w+")
csv_f=csv.writer(f)
csv_f.writerow(['ASIN','Title','SalesRank','TotalReviews','AvgRating','DegreeCentrality','ClusteringCoeff'])
for asin in Top5_byAvgRating_then_byTotalReviews:
    csv_f.writerow(asin)
```

**Output: -**

Looking for Recommendations for Customer purchasing this Book:
ASIN = 0875421210
Title = Earth Power: Techniques of Natural Magic (Llewellyn's Practical Magick)
SalesRank = 100261
TotalReviews = 47
AvgRating = 4.5
DegreeCentrality = 14
ClusteringCoeff = 0.67
Top 5 Recommendations by AvgRating then by TotalReviews for Users Purchased the book:
---
ASIN Title SalesRank TotalReviews AvgRating DegreeCentrality ClusteringCoeff
('0875421229', "Cunningham's Encyclopedia of Magical Herbs (Llewellyn's Sourcebook Series)", 6565, 96, 4.5, 68, 0.65)
('0875421849', "Living Wicca: A Further Guide for the Solitary Practitioner (Llewellyn's Practical Magick)", 6003, 95, 4.5, 30, 0.81)
('0875421318', "Earth, Air, Fire, and Water: More Techniques of Natural Magic (Llewellyn's Practical Magick Series)", 7286, 57, 4.5, 10, 0.73)
('0875421261', "Cunningham's Encyclopedia of Crystal, Gem, and Metal Magic", 14867, 39, 4.0, 17, 0.54)
('0875421245', "The Magical Household: Spells & Rituals for the Home (Llewellyn's Practical Magick Series)", 111836, 21, 4.0, 8, 0.7)


Depth-1 Ego Network




| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| ASIN | Title | SalesRank | TotalReview | AvgRating | DegreeCentrality | ClusteringCoeff |
| 152010661 | Time for Bed | 3122 | 87 | 5 | 60 | 0.57 |
| 694006246 | Big Red Barn Board B | 4457 | 40 | 5 | 27 | 0.57 |
| 1581170769 | What Makes a Rainbc | 40821 | 29 | 5 | 7 | 0.8 |
| 60235152 | From Head to Toe | 43438 | 22 | 5 | 5 | 0.93 |
| 694013013 | From Head to Toe Bo | 6026 | 22 | 5 | 48 | 0.58 |

Top Five Books Recommendations

## Conclusion

Recommendation systems help users discover items they might not have found by themselves and promote sales to potential customers, which provide an effective form of targeted marketing by creating a personalized shopping experience for each customer. Lots of companies have such kind of systems, especially for e-commerce companies like Amazon.com, an effective product recommendation system is very essential to their businesses.In this project by using the concept of Ego networks which is the most appropriate technique for our given dataset, to understand and analyze the copurchased data will lead to the best recommendations.This method is effective as well as computationally cheap that will provide accurate recommendations to users.

## Future Work

We also would like to study how we could control or tweak the outputs of recommendation systems based on application-specific requirements. For example, a company might want to avoid recommending some very popular items to distribute the traffic to other products, or the company would like to promote some newly listed products. In general, it is a promising direction to build recommendation systems that can adapt to more granular and flexible application-specific requirements.

# References

[1] Amazon product co-purchasing network metadata (http://snap.stanford.edu/data/amazon-meta.html)

[2] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. Proceeding of WSDM 2011, pages 635–644, 2011.

[3] Jure Leskovec, Lada A.Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. ACM Transactions on the Web (ACMTWEB), 1(1), 2007.

[4] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-toitem collaborative filtering. IEEE Internet Computing, 7(1):76–80, 2003.

[5] Tao Zhou, Jie Ren, Matus Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. Physical Review E, page 76, 2007.