Recommendation Engine (Using graphs)

DST project

Efforts :
Aashish SIngh (2K19/IT/001)
Aman Jain (2K19/IT/013)

Problem Statement

- **Recommend users related products based on general user data using graphs**
- A recommendation engine is a system that suggests products, services, information to users based on analysis of data. Notwithstanding, the recommendation can derive from a variety of factors such as the history of the user and the behaviour of similar users

Dataset used

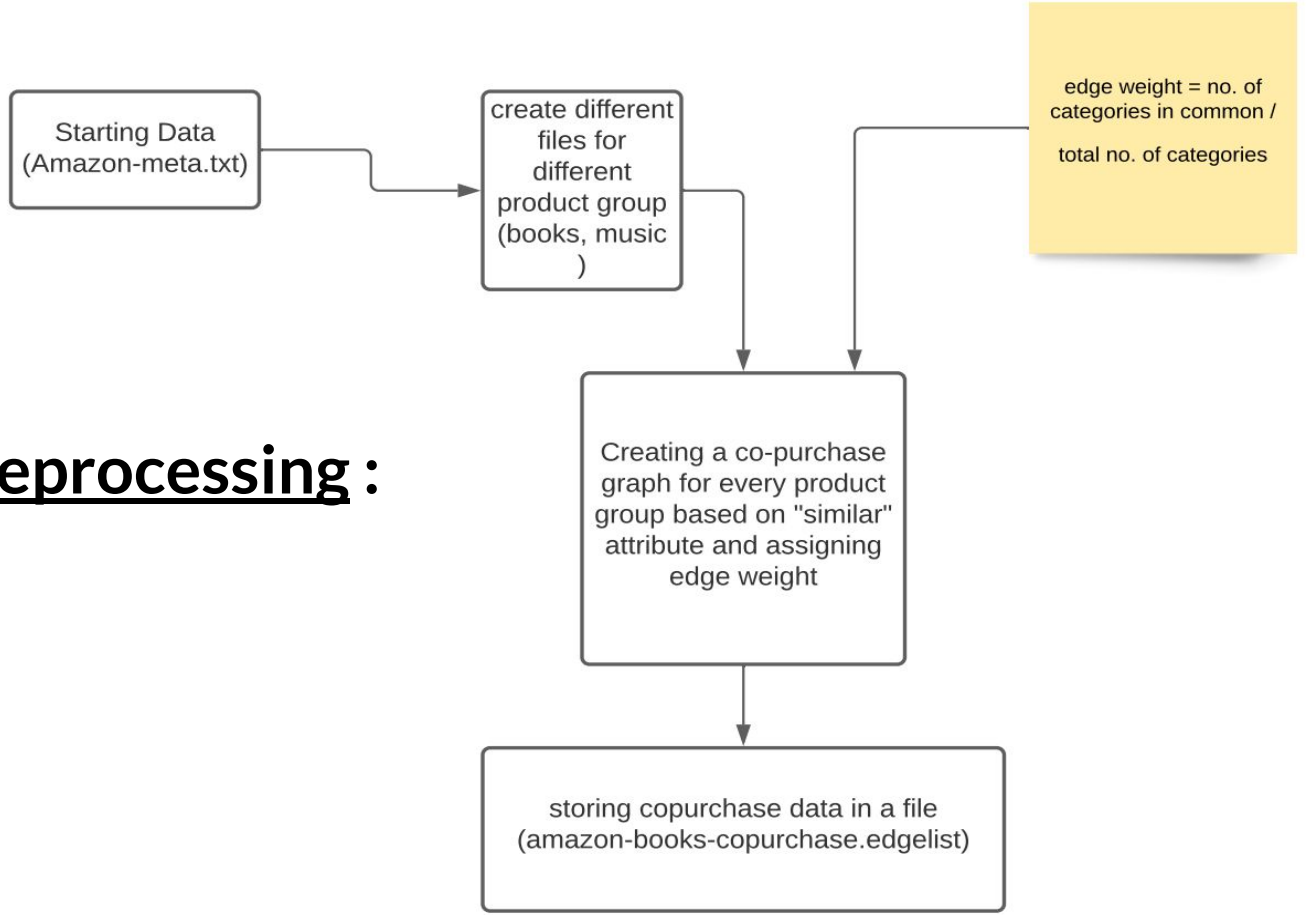
- We have used Amazon's actual dataset for our project
- Dataset consist of product metadata
- Sample dataset-

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

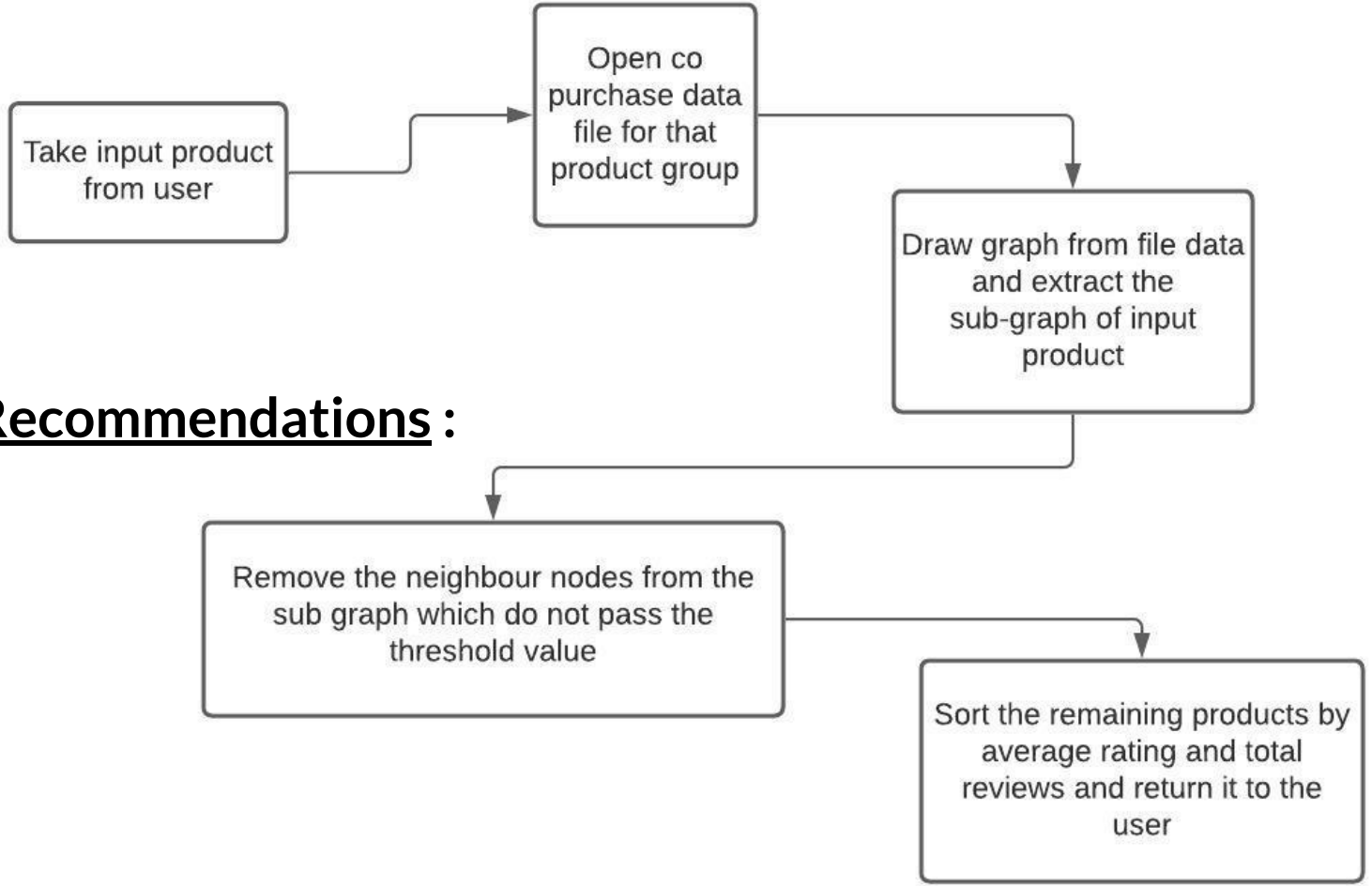
Flow charts

1. Preprocessing
2. Recommendation System

Preprocessing :



Recommendations :



Code snippets

Data formatting

```
# PREPROCESSING
#----------------
fhr = open('amazon-meta.txt', 'r', encoding='utf-8', errors='ignore')# Reading input file
amazonProducts = {} #initialising an empty object

# Read data from amazon meta files and popoulate amazon products nested dictionary
(Id, ASIN, Title, Categories, Group, Copurchased, SalesRank, TotalReviews, AvgRating) = \
    ("", "", "", "", "", "", 0, 0, 0.0) # initialising values
for line in fhr:
    line = line.strip() #removing whitespaces
    # a product block started
    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 = ' '.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()
        # product block end
        # write out fields to amazonProducts dictionary
        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()
```

Data Classification

```
#create specific dictionaries
amazonBooks = {}
amazonMusic = {}
for asin,metadata in amazonProducts.items():
    if (metadata['Group']=='Book'):
        amazonBooks[asin] = amazonProducts[asin]
    elif(metadata['Group']=='Music'):
        amazonMusic[asin] = amazonProducts[asin]
    elif(metadata[asin]=='' )

# Write amazonBooks data to file
fhw = open('amazon-books.txt', 'w', encoding='utf-8', errors='ignore')
fhw.write("Id\t" + "ASIN\t" + "Title\t" + "Categories\t" + "Group\t" + "Copurchased\t" + "SalesRank\t" + "TotalReviews\t" + "AvgRating\n")
for asin, metadata in amazonBooks.items():
    fhw.write(metadata['Id'] + "\t" + \
              asin + "\t" + \
              metadata['Title'] + "\t" + \
              metadata['Categories'] + "\t" + \
              metadata['Group'] + "\t" + \
              metadata['Copurchased'] + "\t" + \
              str(metadata['SalesRank']) + "\t" + \
              str(metadata['TotalReviews']) + "\t" + \
              str(metadata['AvgRating']) + "\n")
fhw.close()
```

Creating co-purchased graph

```
120  # Create a product copurchase graph for analysis where the graph nodes for product ASINs
121  # and graph edge exists if two products were copurchased,
122  # with edge weight being a measure of category similarity between ASINs
123
124  copurchaseGraphBooks = networkx.Graph()
125  for asin,metadata in amazonBooks.items():
126      copurchaseGraphBooks.add_node(asin)
127      for a in metadata['Copurchased'].split():
128          copurchaseGraphBooks.add_node(a.strip())
129          similarity = 0
130          n1 = set((amazonBooks[asin]['Categories']).split())
131          n2 = set((amazonBooks[a]['Categories']).split())
132          n1In2 = n1 & n2     # intersection (Number of words that are common between Categories of connected Nodes)
133          n1Un2 = n1 | n2     # union (Total number of words in both Categories of connected Nodes)
134          if(len(n1Un2)) > 0:
135              similarity = round(len(n1In2)/len(n1Un2), 2)
136          copurchaseGraphBooks.add_edge(asin, a.strip(), weight=similarity)
137
138  fhw = open('amazon-books-copurchase.edgelist', 'wb')
139  networkx.write_weighted_edgelist(copurchaseGraphBooks, fhw)
140  fhw.close()
```

Taking user input

```
16  print("Available groups\n")
17  print("1. Books\n")
18  print("2. Music\n")
19
20  val = input("Choose the group\n")
21  numberToGroupMapping = {
22    "1": "amazon-books",
23    "2": "amazon-music"
24  }
25
26  # key = ASIN; value = MetaData associated with ASIN
27  fhr = open(numberToGroupMapping[val]+".txt", 'r', encoding='utf-8', errors='ignore')
28  amazonProducts = {}
29  display = []
30  fhr.readline() #read first line and skip to second line
31  for line in fhr:
32      cell = line.split('\t')
33      MetaData = {}
34      MetaData['Id'] = cell[0].strip()
35      ASIN = cell[1].strip()
36      MetaData['Title'] = cell[2].strip()
37      MetaData['Categories'] = cell[3].strip()
38      MetaData['Group'] = cell[4].strip()
39      MetaData['Copurchased'] = cell[5].strip()
40      MetaData['SalesRank'] = int(cell[6].strip())
41      MetaData['TotalReviews'] = int(cell[7].strip())
42      MetaData['AvgRating'] = float(cell[8].strip())
43      display.append(cell[1].strip())
44      amazonProducts[ASIN] = MetaData
45  fhr.close()
```

Getting sub-graph of product selected by user

```
60
61  purchasedAsin = input('Enter your product id\n')
62
63  print("Looking for Recommendations for this Product:")
64  print("\n-----------------------------------------------------------")
65
66  # Get the depth-1 ego network of purchasedAsin from copurchaseGraph
67  try:
68      n = purchasedAsin
69      ego = networkx.ego_graph(copurchaseGraph, n, radius=1)
70      purchasedAsinEgoGraph = networkx.Graph(ego)
71  except:
72      print("No similar product available")
73      exit()
74
75
76  # only retain edges with Threshold value
77  threshold = 0.1
78  purchasedAsinEgoTrimGraph = networkx.Graph()
79  purchasedAsinEgoTrimGraph.add_node(purchasedAsin)
80
81  for f,t,e in purchasedAsinEgoGraph.edges(data=True):
82      if e['weight'] >= threshold:
83          purchasedAsinEgoTrimGraph.add_edge(f,t, weight=e['weight'])
84
85  networkx.draw_networkx(purchasedAsinEgoTrimGraph)
86  plt.show()
87
```

Returning recommendations to the user

```
# Get the list of nodes connected to the purchasedAsin
purchasedAsinNeighbours = purchasedAsinEgoTrimGraph.neighbors(purchasedAsin)

# Get Top Five book recommendations from among the purchasedAsinNeighbours based on one or more of the following data of the
# neighbouring nodes: SalesRank, AvgRating, TotalReviews

# Accessing metadata with ASIN in purchasedAsinNeighbours
AsMeta = []
for asin in purchasedAsinNeighbours:
    ASIN = asin
    Title = amazonProducts[asin]['Title']
    SalesRank = amazonProducts[asin]['SalesRank']
    TotalReviews = amazonProducts[asin]['TotalReviews']
    AvgRating = amazonProducts[asin]['AvgRating']
    Similarity = str(amazonProducts[asin]['Similarity']*100)+"%"
    AsMeta.append([ASIN, Title, AvgRating, TotalReviews, Similarity])

# Sorting the top five nodes in purchasedAsinNeighbour by Average Rating then by TotalReviews
T5_byAvgRating_then_byTotalReviews = sorted(AsMeta, key=lambda x: (x[3], x[2]), reverse=True)

# Print Top Recommendations
print('\nTop Recommendations by AvgRating then by TotalReviews for this Product:')
print('\n--------------------------------------------------------------------------------')
t = PrettyTable(['Product ID', 'Title', 'Average Ratimg', 'Total Reviews', 'Similarity'])
for asin in T5_byAvgRating_then_byTotalReviews:
    t.add_row(asin)

print(t)
```

Result

```
Available groups

1. Books

2. Music

Choose the group
1
+------------+-----------------------------------------------------------------------------------------------+----------------+--------------+
| Product ID |                                             Title                                             | Average Rating | Total Reviews |
+------------+-----------------------------------------------------------------------------------------------+----------------+--------------+
| 0028638506 |                  The Complete Idiot's Guide to American History, Second Edition               |      3.0       |      14      |
| 1586211943 |                                    Swimming Across : A Memoir                                 |      4.5       |      26      |
| 1570194424 |                                        War of the Worlds                                      |      4.5       |      5       |
| 9626341017 |            Treasure Island (Classic Literature With Classical Music. Junior Classics)        |      4.0       |      197     |
| 0913321079 |                                      Muhammad the Prophet                                     |      0.0       |      0       |
| 0252067282 |   Pistol Packin' Mama: Aunt Molly Jackson and the Politics of Folksong (Music in American Life)   |      5.0       |      2       |
| 0875885225 |                                    Crissy Family Encylopedia                                  |      4.5       |      7       |
| 0595193501 |                    The Claims of Christ: What Jesus Had to Say About Himself                  |      5.0       |      6       |
| 0670877697 |                                      The Roald Dahl Treasury                                  |      4.5       |      16      |
| 1893342026 |                  Guide to Maryland Trout Fishing: The Catch and Release Streams               |      5.0       |      1       |
| 0486264815 |      Chinese Brushwork in Calligraphy and Painting : Its History, Aesthetics, and Techniques      |      4.0       |      1       |
| 0140388346 |                                          The Thief                                            |      4.5       |      87      |
| 0805039392 |                        The Evidence of Things Not Seen : Reissued Edition                     |      4.0       |      3       |
| 0764120611 |                           Japanese Grammar (Barron's Grammar Series)                          |      4.0       |      8       |
| 1590790154 | Mastering the Rockefeller Habits: What You Must Do to Increase the Value of Your Fast-Growth Firm |      5.0       |      14      |
+------------+-----------------------------------------------------------------------------------------------+----------------+--------------+
Enter your product id
```

Product ID	Title	Average Rating	Total Reviews
0028638506	The Complete Idiot's Guide to American History, Second Edition	3.0	14
1586211943	Swimming Across : A Memoir	4.5	26
1570194424	War of the Worlds	4.5	5
9626341017	Treasure Island (Classic Literature With Classical Music. Junior Classics)	4.0	197
0913321079	Muhammad the Prophet	0.0	0
0252067282	Pistol Packin' Mama: Aunt Molly Jackson and the Politics of Folksong (Music in American Life)	5.0	2
0875885225	Crissy Family Encylopedia	4.5	7
0595193501	The Claims of Christ: What Jesus Had to Say About Himself	5.0	6
0670877697	The Roald Dahl Treasury	4.5	16
1893342026	Guide to Maryland Trout Fishing: The Catch and Release Streams	5.0	1
0486264815	Chinese Brushwork in Calligraphy and Painting : Its History, Aesthetics, and Techniques	4.0	1
0140388346	The Thief	4.5	87
0805039392	The Evidence of Things Not Seen : Reissued Edition	4.0	3
0764120611	Japanese Grammar (Barron's Grammar Series)	4.0	8
1590790154	Mastering the Rockefeller Habits: What You Must Do to Increase the Value of Your Fast-Growth Firm	5.0	14

Enter your product id
0140388346
Looking for Recommendations for this Product:

Top Recommendations by AvgRating then by TotalReviews for this Product:

Product ID	Title	Average Ratimg	Total Reviews	Similarity
0140376410	The Ear, the Eye, and the Arm	4.5	265	65.0%
0140386351	A Girl Named Disaster	4.0	83	47.0%
0531095398	A Girl Named Disaster	4.0	83	41.0%
068982033X	The Moorchild (Aladdin Fantasy)	4.5	54	43.0%
068817423X	The Queen of Attolia	4.5	31	31.0%
0380733048	The Queen of Attolia (rpkg)	4.5	31	32.0%
0374410828	Celine (Sunburst Book)	4.5	12	24.0%
0446522481	A Kind of Grace : The Autobiography of the World's Greatest Female Athlete	5.0	9	10.0%

Example of a sub-graph