Amazon Product Co-Purchasing Network Analysis

Mu Niu, Kai Wa Ho, Jingtang

2024-06-06

Introduction

In this project, we conduct a comprehensive social network analysis on a dataset derived from Amazon's product co-purchasing records. The raw edge data, which was collected on June 1, 2003, from the Amazon website, includes 403,394 nodes representing individual products and 3,387,388 unweighted directed edges indicating frequently co-purchased product pairs. The direction of an edge signifies the order of purchase, with an outward edge from product A to product B indicating that product B is frequently purchased after product A. This dataset is available at **Amazon Product Co-Purchasing Network.**

Additionally, the raw nodes data was collected by crawling Amazon's website and encompasses metadata and review information for 548,552 different products, including books, music, DVDs, and video tapes. This dataset provides detailed information such as the product title, sales rank, list of similar products, detailed product categorization, and reviews. This data was collected in the summer of 2006, and it can be accessed at **Amazon Product Metadata.**

After a thorough data cleaning process, we integrated the edge data and node information data, resulting in an igraph object containing 398,688 nodes. Each node has four attributes: ID (unique product ID), Title (product name), Group (class the product belongs to), and Category (subcategory of the class).

The primary objectives of this analysis are to understand the relationships between products, identify co-purchasing patterns and customer shopping behaviors through network data visualization, and explore various network, node, and edge metrics. We aim to interpret these metrics within the network context, apply community detection algorithms, and analyze the resulting communities. Additionally, we will examine the network's adjacency matrix, reorder vertices to highlight patterns, and observe changes in the matrix to identify clearer patterns based on community or connected components.

Methodology

In this project, we followed a structured approach to process the data, analyze the network, and visualize the results.

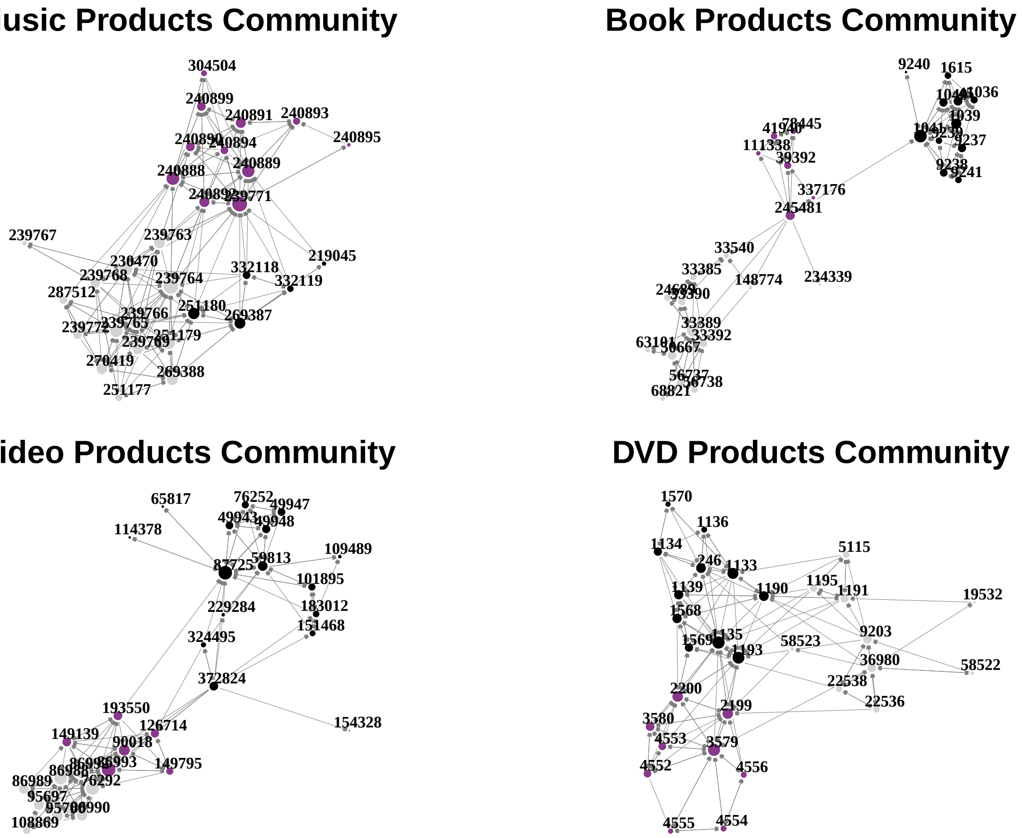
The data processing procedures began with mapping the nodes information dataset, which was a text file with issues of missing data and inconsistent formats, into a dataframe with four columns: ID, Title, Group, and Category. Subsequently, we filtered out all edges in the edge dataset that connected nodes with missing IDs, product titles, or product groups. This ensured that only valid and complete data was included in the analysis. The mapped nodes information data and filtered edge data was then integrated into an igraph object containing 398,688 nodes with four vertex attributes: ID, Title, Group, and Category. The data cleaning process employed packages such as `dplyr` for data manipulation and `igraph` for constructing and managing the igraph object in R.

For sampling methodology, we generated 4 induced subgraphs by using a node-centric approach. For each sub-network, we randomly selected one node as the starting point and retrieved all nodes connected to it. We then iteratively expanded our sample by including nodes connected to the current set of nodes, continuing this process until the specified number of nodes was reached. This sampling method was designed to capture densely connected nodes, facilitating better visualization and understanding of the relationships between products.

This comprehensive methodology enabled us to effectively analyze and interpret the Amazon product co-purchasing network, uncovering valuable insights into customer shopping behaviors and product relationships.

Analysis

After sampling four sub-networks, each formed by using our previously described sampling method and starting with nodes from the book, music, video, and DVD groups respectively, we visualized the subgraphs. This visualization was performed using the base plot and Plotly package in R to generate both static and interactive plots.


Music Products Community

Book Products Community

Video Products Community

DVD Products Community

The visualizations revealed several interesting patterns in co-purchasing behavior. For instance, we observed that individuals who purchase education books also tend to listen to R&B, rap, and hip-hop music. This correlation might suggest a demographic overlap between students or educators and fans of these music genres. Additionally, we found that people who buy nonfiction books also favor history books, which is logical as both are grounded in factual content. Travel book enthusiasts were also seen to purchase audiobooks, likely due to the convenience of not carrying physical copies while traveling. Similarly, customers who buy children's books often buy audiobooks, possibly because audiobooks provide a convenient way for babysitters to engage children. In the DVD group, those who enjoy comedy DVDs also show a diverse taste in music, purchasing DJ music, pop, rock, and metal. An intriguing finding was that several memoir nodes were linked to R&B and DJ music, which might suggest that people prefer listening to these genres while reading memoirs. Additionally, we found that individuals interested in business and investment books also showed an interest in books on arts and photography.

These visualizations enhanced our understanding of the relationships between products and allowed us to identify distinct co-purchasing patterns. The interactive plots, containing more detailed information, can be accessed **here.** By analyzing these subgraphs, we gained valuable insights into customer behavior and product affinities, which can be leveraged for better marketing strategies and product recommendations.

Results

Conclusion

References

Appendices

```r
# Loading Library
library(readr)
library(igraph)
library(rsample)
library(dplyr)
library(stringr)
library(plotly)

# read data
meta <- readLines('../data/amazon-meta.txt')

# map into df with 5 columns: Id, Title, Group, Categories, Subcategory
meta <- data.frame(line = meta, stringsAsFactors = FALSE)

filtered_meta <- meta %>%
  mutate(keep = grepl("Id:|title:|group:|categories:", line) |
               lag(grepl("categories:", line), default = FALSE)) %>%
  filter(keep) %>%
  select(-keep)

# map to df
result <- data.frame(Id = character(),
                     Title = character(),
                     Group = character(),
                     Categories = character(),
                     Subcategory = character(),
                     stringsAsFactors = FALSE)


# Process the data
i <- 1
while (i <= nrow(filtered_meta)) {
  current_row <- filtered_meta$line[i]
  if (grepl("Id:  ", current_row)) {
    # Check if there are at least three more rows and they match the expected titles
    if (i + 4 <= nrow(filtered_meta) &&
        grepl("  title: ", filtered_meta$line[i + 1]) &&
        grepl("  group: ", filtered_meta$line[i + 2]) &&
        grepl("  categories: ", filtered_meta$line[i + 3])) {
```

```
      # Extract and clean the data
      id <- as.integer(sub("Id:  ", "", filtered_meta$line[i]))
      title <- sub("  title: ", "", filtered_meta$line[i + 1])
      group <- sub("  group: ", "", filtered_meta$line[i + 2])
      categories <- as.integer(sub("  categories: ", "", filtered_meta$line[i + 3]))
      if (categories == 0) {
        subcategory <- NA  # Return NA if categories are 0
      }
      else if (categories > 0){
        if (group %in% c("Music", "Book", "Video")){
          sub <- filtered_meta$line[i + 4]
          subcategory <- str_split(sub, "\\|")[[1]][4] %>% str_replace_all("\\[.*?\\]", "")
          if (subcategory %in% c('Reference', 'Genres')){
            subcategory <- str_split(sub, "\\|")[[1]][5] %>% str_replace_all("\\[.*?\\]", "")
          }
        }
        else if (group == "DVD"){
          sub <- filtered_meta$line[i + 4]
          subcategory <- str_split(sub, "\\|")[[1]][5] %>% str_replace_all("\\[.*?\\]", "")
        }
      }
      # Append to result dataframe
      result <- rbind(result, data.frame(Id = id,
                                         Title = title,
                                         Group = group,
                                         Categories = categories,
                                         Subcategory = subcategory,
                                         stringsAsFactors = FALSE))
      # Move index forward by 4
      i <- i + 5
    }
    else {
      # Skip the current Id and all following rows until next Id or end of data
      i <- i + 1
      while(i <= nrow(filtered_meta) && !grepl("Id:  ", filtered_meta$line[i])) {
        i <- i + 1
      }
    }
  }
}

# write to csv
result %>% write.csv('../data/meta_data.csv')

# read edges data
meta = read.csv('../data/meta_data.csv')
data = read.table("../data/amazon0601.txt")
colnames(data) = c("From", "To")

# keep edges that link nodes we have info on
filtered_data = data[(data$From %in% meta$Id) & (data$To %in% meta$Id), ]

# write to csv
```

```r
filtered_data %>% write.csv('../data/filtered_data.csv')

# convert to igraph
amz <- graph_from_data_frame(filtered_data, directed = TRUE)

# create attributes for igraph object from meta data
title = c()
group = c()
sub = c()

for (i in V(amz)$name){
  if (as.integer(i) %in% meta$Id){
    title = append(title, meta[meta$Id == as.integer(i), ]$Title)
    group = append(group, meta[meta$Id == as.integer(i), ]$Group)
    sub = append(sub, meta[meta$Id == as.integer(i), ]$Subcategory)
  }
}

# handle missing value
sub[is.na(sub)] <- "missing"

V(amz)$title <- title
V(amz)$group <- group
V(amz)$sub <- sub

# save igraph object
saveRDS(amz, file = '../data/amz_igraph.rds')

amz <- readRDS(file = '../data/amz_igraph.rds')

# Function to retrieve connected nodes up to a given count
retrieve_connected_nodes <- function(graph, start_node, count = 30) {
  # Initialize the list with the start node
  nodes_to_explore <- list(start_node)
  connected_nodes <- c(start_node)

  # Keep a list to avoid revisiting nodes
  visited_nodes <- numeric(0)

  # Explore the graph until we reach the desired number of nodes
  while (length(nodes_to_explore) > 0 && length(connected_nodes) < count) {
    current_node <- nodes_to_explore[[1]]
    nodes_to_explore <- nodes_to_explore[-1]  # Remove the explored node

    # Skip if already visited
    if (current_node %in% visited_nodes) next

    # Mark as visited
    visited_nodes <- c(visited_nodes, current_node)

    # Get neighbors and add to nodes to explore
    neighbors <- neighbors(graph, current_node)
    new_neighbors <- neighbors[!neighbors %in% connected_nodes]
    nodes_to_explore <- c(nodes_to_explore, as.list(new_neighbors))
    connected_nodes <- c(connected_nodes, new_neighbors)

    # Limit the collection if it exceeds the desired count
    if (length(connected_nodes) > count) {
      connected_nodes <- connected_nodes[1:count]
      break
    }
  }

  # Return the vertex sequence of connected nodes
  return(connected_nodes)
}

# Define a function to generate and plot the community for a given product group
plot_product_community <- function(group_name, main_title) {
  set.seed(194)
  # Randomly select one node from the specified group
  random_node <- sample(V(amz)[V(amz)$group == group_name], 1)

  # Apply function to get 200 related nodes
  nodes <- retrieve_connected_nodes(amz, random_node)
  network <- induced_subgraph(amz, nodes)

  # Choose a layout that spreads out the nodes more effectively
  layout <- layout_with_fr(network)

  # Set graph margins to zero
  par(mar = c(0, 0, 2, 0))

  # Base plot
  plot(network, layout = layout,
       vertex.color = "#88398A",
       vertex.frame.color = "#FFFFFF",
       vertex.size = 5,
       vertex.label = V(network)$name,
       vertex.label.dist = 1,
       vertex.label.cex = 0.8,
       vertex.label.color = "black",
       vertex.label.font = 2,
       edge.color = "gray50",
       edge.width = 0.2,
       edge.arrow.size = 0.1,
       main = paste(main_title, "Base Plot"),
       bg = "white"
  )

  # Community plot
  cluster <- cluster_optimal(network)
  mycomcols <- c("black", "#D3D3D3", "#88398A")

  plot(network, layout = layout,
       vertex.color = mycomcols[cluster$membership],
       vertex.frame.color = "#FFFFFF",
       vertex.size = sqrt(degree(network)) * 2,
       vertex.label = V(network)$name,
       vertex.label.dist = 1,
       vertex.label.cex = 0.6,
       vertex.label.color = "black",
       vertex.label.font = 2,
       edge.color = "gray50",
       edge.width = 0.2,
       edge.arrow.size = 0.1,
       main = paste(main_title, "Community"),
       bg = "white"
  )

  # Get vertex data including the degree for size scaling
  vertex_data <- data.frame(
    Id = V(network)$name,
    x = layout[, 1],
    y = layout[, 2],
    degree = degree(network),
    Title = V(network)$title,
    Group = V(network)$group,
    Category = V(network)$sub
  )

  # Enhance hover info by including all attributes except x, y coordinates
  vertex_data$hoverinfo <- apply(vertex_data[, -c(2, 3)], 1, function(row) {
    paste(names(row), row, sep=": ", collapse="<br>")
  })

  # Get edge data
  edge_data <- get.data.frame(network, what = "edges")

  # Join edge data with vertex data to get coordinates for 'from' and 'to'
  edge_data <- merge(edge_data, vertex_data, by.x = "from", by.y = "Id", all.x = TRUE)
  edge_data <- merge(edge_data, vertex_data, by.x = "to", by.y = "Id", all.x = TRUE, suffixes = c(".fro

  # Prepare data for Plotly plot
  edges <- list(
    x = c(rbind(edge_data$x.from, edge_data$x.to, NA)),
    y = c(rbind(edge_data$y.from, edge_data$y.to, NA)),
    type = "scatter",
    mode = "lines",
    line = list(color = "grey", width = 0.5)
  )

  nodes <- list(
    x = vertex_data$x,
    y = vertex_data$y,
    hovertext = vertex_data$hoverinfo,
    mode = "markers",
    marker = list(size = vertex_data$degree * 2,
                  color = mycomcols[cluster$membership]),
```

```r
    type = "scatter",
    hoverinfo = "text"
  )

  # Create the plot
  plot_ly() %>%
    add_trace(x = edges$x, y = edges$y, mode = edges$mode, type = edges$type, line = edges$line) %>%
    add_trace(x = nodes$x, y = nodes$y, hovertext = nodes$hovertext, mode = nodes$mode, type = nodes$typ
    layout(
      title = paste("Network Visualization of", main_title),
      xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
      yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
      hovermode = 'closest'
    )
}

# Music group
plot_product_community("Music", "Music Products")

# Book group
plot_product_community("Book", "Book Products")

# Video group
plot_product_community("Video", "Video Products")

# DVD group
plot_product_community("DVD", "DVD Products")
```