CS765: Design Challenge 3

Nomin Khishigsuren

1 Introduction

For Design Challenge 3, I implemented Force-Directed Graph using D3 to illustrate the relationship between Amazon product categories that have the most products in common. The graph was implemented and published on Observable and can be accessed through this link. Figure 1 shows what the graph looks like.

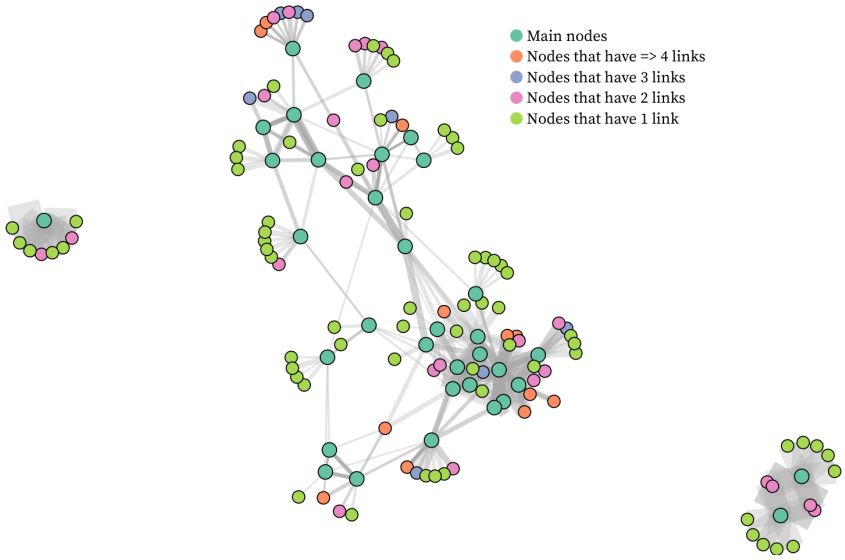

Figure 1: Force-directed graph at a glance

Because the aim of my visualization was to show the Amazon product categories that have common products, and explore what the relationship is between these categories, implementing a network graph was the obvious choice to me. While I was looking at different types of network graphs to implement, Force-Directed graph especially appealed to me because it seemed a more intuitive way to see a network. D3 uses 'force' (or more specifically, 'link') to connect more related nodes together while separating unrelated ones with 'charge'.

In this way, you can identify clumps of nodes where 'connectedness' or 'relatedness' is high and outliers that are unrelated to the other nodes. While network graphs can handle large amount of data, I wanted to select enough data so that the network was complex and not too much so that the details and the paths of the network were not lost.

2 Data

The dataset that was used to create this network graph consisted of selecting 50 categories that had the highest 'also' count or the highest number of nodes/other categories for which

its products were also in. These are what I call 'Main nodes' in my visualization.

From these main nodes, I extracted their top 10 'also' categories ranked by the number of products these categories had in common. I was able to transform this data into a format of node and links that D3 recognized. The code that I implemented to do this is attached in the submission. Figure 2 shows the input dataset types.

Dataset Type: Network format containing nodes, links and attributes

Attribute	Type of Attribute
Links.source	Categorical
Links.target	Categorical
Links.value	Ordered Quantitative
Node.name	Categorical
Node.group	Categorical
Node.nodesize	Ordered Quantitative

Figure 2: Dataset type

The link value corresponds to the number of products that two categories have in common. The node group corresponds to the number of links the node has. The node size was used to distinguish the main nodes from the other nodes.

3 Tasks

The main task of this visualization is to show the relationship between the categories that have the most products in common by helping viewers explore the network through interaction. The topology of the network i.e. clusters and outliers are important for the viewer to see and understand.

Secondary tasks include helping viewers browse and locate the paths in the network to see the relationship between specific categories, and identifying the categories that are associated most to other categories.

4 Design choices

Using Force-Directed graph to implement the network was a design choice to optimize the clustering and showing the outliers. The graph is implemented to allow the user to select a node and when once selected, the node's name and its direct neighbors (and their names) are highlighted. Figure 3 shows an example of a node being selected.

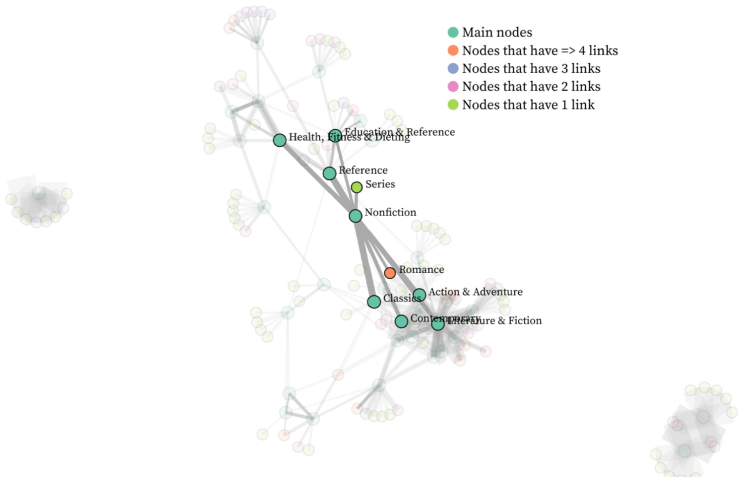

Figure 3: An example of a node ('Nonfiction') being selected

This 'select' interactive element helps the user to focus on the selected node/category and its links to other categories by highlighting it. It also helps the user to visualize the relationship between the categories more clearly. Allowing the user to select also helps the task of explore, browse and locate certain categories and its neighbors.

The graph also has a dynamic layout and an interactive element of letting the user drag a node or cluster around to avoid congestion. This helps the user to manipulate and interact with the network to see what is being shown better. Figure 4 shows this drag simulation in action.

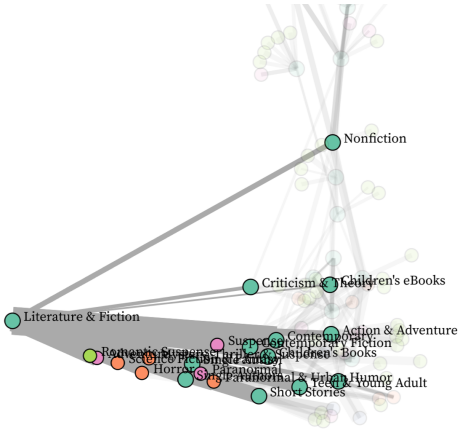

Figure 4: An example of a node being dragged

One of the tasks of this visualization is to identify the categories that have the most neighbors/links, not accounting for the main nodes. To address this task, I grouped the nodes into bins depending on the number of links they had and encoded the group attribute as color. Because the data had many overlaps, it did not translate well to the graph but for the task, the color helps the viewer to easily locate and identify the categories with

the highest number of links. Similarly, the node size attribute was encoded as size to distinguish the main nodes from the rest. Figure 5 shows an example of a node with 4 links being selected.

Figure 5: A node with 4 links ('Historical') being selected

The link value corresponds to the number of products any two categories share and is encoded as the link thickness. Figure 6 shows an example of two categories having many products in common.

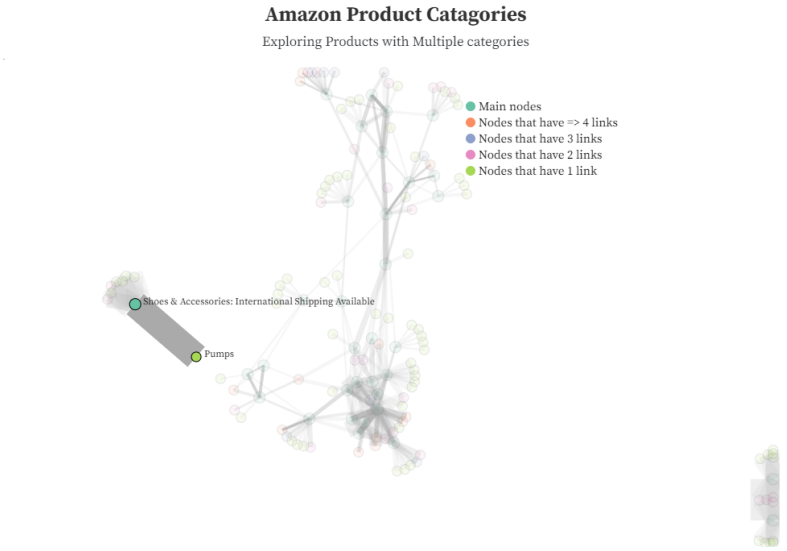

Figure 6: An example to show that many products have 'Pump' and 'Shoes & Accessories' as their categories

I think all the encodings and interactive elements that this graph showcases help the viewer to see and explore the data, and visualize and understand the relationships between categories.

5 Cases

I have 2 insights that I gained through this visualization that I would like to share. The first one is that the big cluster in the graph where the 'connectedness' is high, are mostly Book categories. Figure 7 shows the cluster.

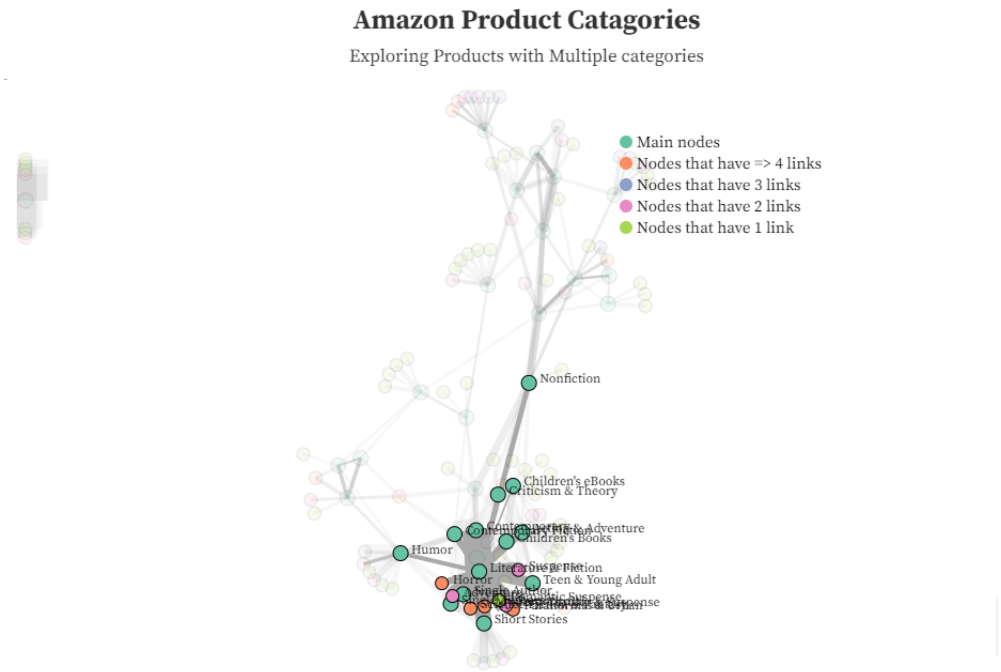

Figure 7: Book categories had the most associated categories

The second insight I had was that Music and Shoe categories had lot of common products within itself, resulting in small bubbles that were unlinked from the main network. Figure 8 shows Music categories as a small network of its own.

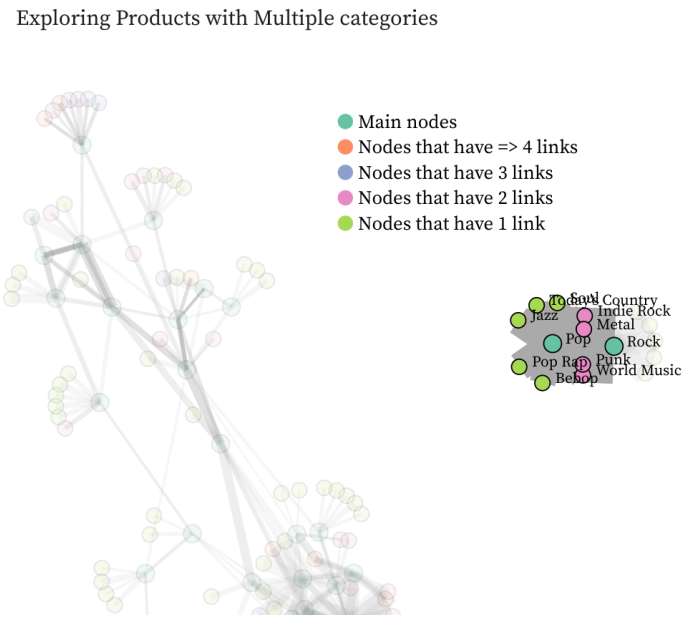

Figure 8: Music categories

6 Self-Assessment

I definitely had bigger plans with this visualization, but having to use D3 with javascript was really difficult as I had almost no experience with it. It made everything so pretty though, so I am glad that I stuck it out and did what I could. I wanted to add buttons so you could choose a 'subcategory' and see its network, for example, selecting 'Books' and seeing its network. Or even, allowing the user to select the number of nodes and links they want to see.

And I was so close to making the zoom function work on this visualization but alas, it kept throwing me errors that I just had to take it out. Having a platform like Observable was really helpful in trying to grasp what was going on, and snip bits and pieces of others code to make mine work. It also helped me showcase what I had without having to turn to Github.

Having to turn the data from the .csv format to .json format that D3 required was also difficult on its own and annoying but it worked out. The Python file I attached takes the .csv file, filters it and outputs a .json data.

In conclusion, despite having to learn things from scratch and constantly running into issues, I am satisfied with what I have turned in. I learned a lot from it and to be honest, that's all I can ask.