Yuheng Chen, Jack Yang

Professor Michael Gleicher

CS 765

DC 3 - Amazon Product Tree

Link to demo video: https://youtu.be/0Uk2PDgQyUs
Link to repository: https://github.com/jackyangzzh/Amazon-Product-Tree
Link to live demo: https://jackyangzzh.github.io/Amazon-Product-Tree/

TASK

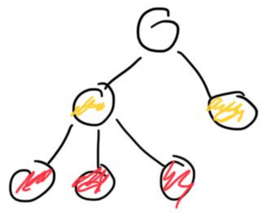
For this project, we have four main tasks in mind: first, given a large amount of data and its hierarchical structure, we want to see the relationship between each category in its authentic tree form. Second, we want to visualize the quantities of subcategories of each node (or whether there is any). Third, we aim to quickly identify which category has the most products without clicking through all the subcategories. Fourth, we want our visualization to reflect the degree of categorization of a particular node. Fifth, we want to create a rich visual representation to provide freedom for the users to explore the data with their own interest in mind. Our tasks can be summarized into four phrases: hierarchical relationship, subcategory representation, subcategory product encoding, categorization, and user interaction.

DATASET

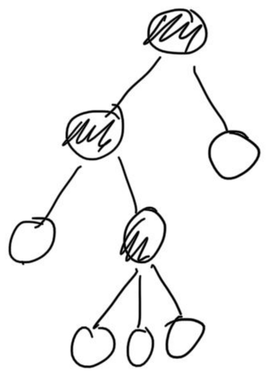
The dataset is a subset of categorical product data by Amazon. There are a total of 4 CSV files contains those data (books, musical instruments, pet supplies, and all). Each dataset contains the following keys: id, name, product count, subtree product count, parent, number of children, a list of children etc. Given a CSV or a JSON file that contains keys listed as such and has a hierarchical relationship, our program will be able to generate a visualization tree accordingly. Our program works for all four files. For demonstration purposes, we will be using the musical instrumental data, which contains 666 entries.

SKETCHES

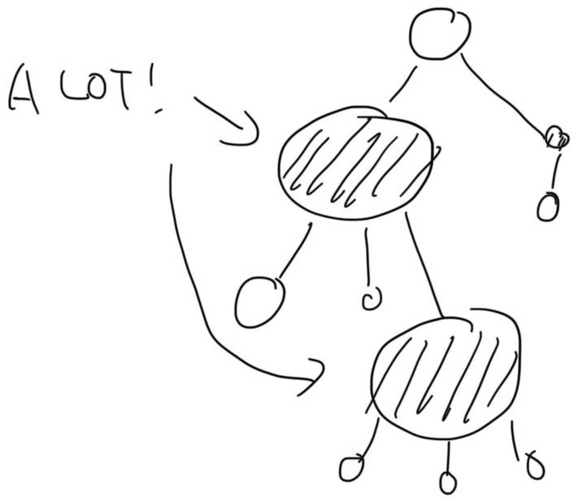
For the first task of representing hierarchical relationships, we planned to use a tree to represent the overall structure and color to represent the encodings.



For the second task of representing which category has subcategories, if so, how many compared to others. We haven't decided on a way to represent the number of subcategories, but so far we decided to fill in the circles to represent if the node has any children.

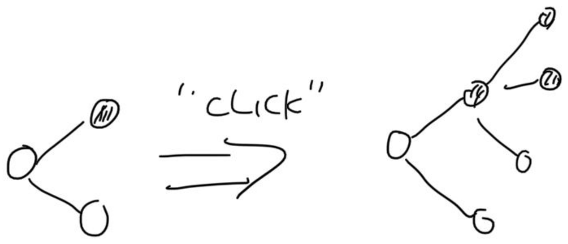


For our third task, we decided to use the size of the circles to represent the amount of items in the subtrees. This is intuitive such that users can visually distinguish the ones with many products versus the ones that do not.



The fourth task can be achieved using the size and color encodings in conjunctions.

For our fifth task of interaction, we decide to have the users interact with the data by mouse clicking the nodes. This way, users are able to select details on demand.

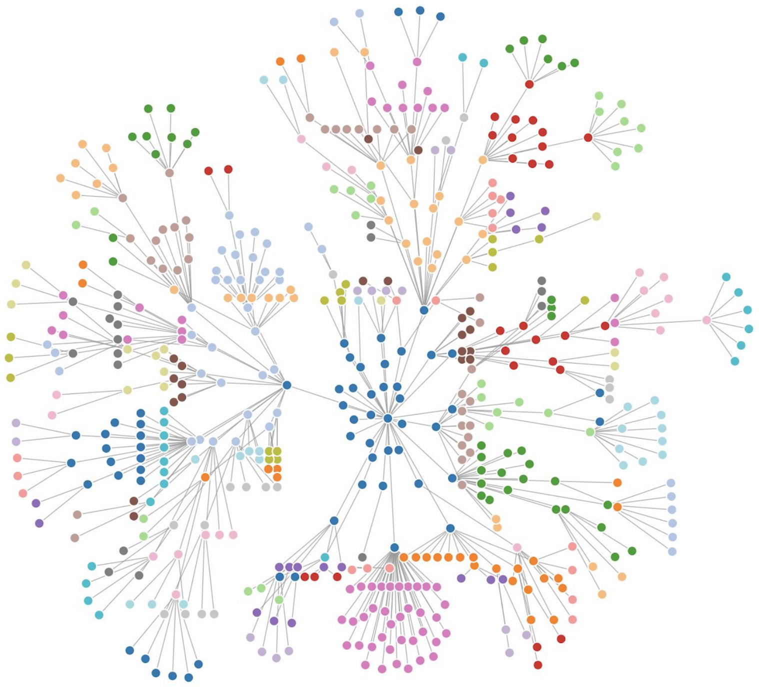


TOOLS

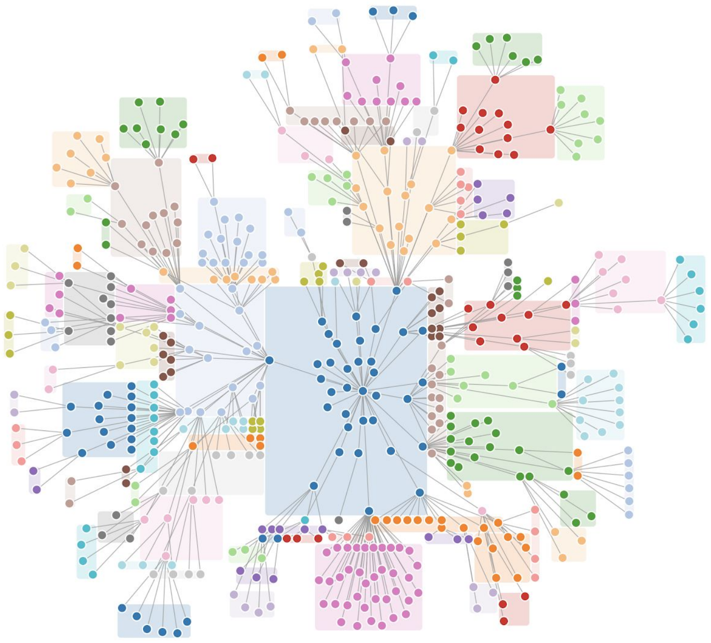
The program is written in JavaScript and HTML using D3.js (https://d3js.org/) and Cola.js (https://ialab.it.monash.edu/webcola/). Instead of including the libraries in the repository, we used Content Delivery Network (CDN) to refer to the libraries. This reduces the size of the file by putting dependencies on a working Internet. The editor we used is Visual Studio Code with Live Server extension for launching a local server.

PROTOTYPE / ATTEMPTS

Our first prototype created with Cola.js was as follows:

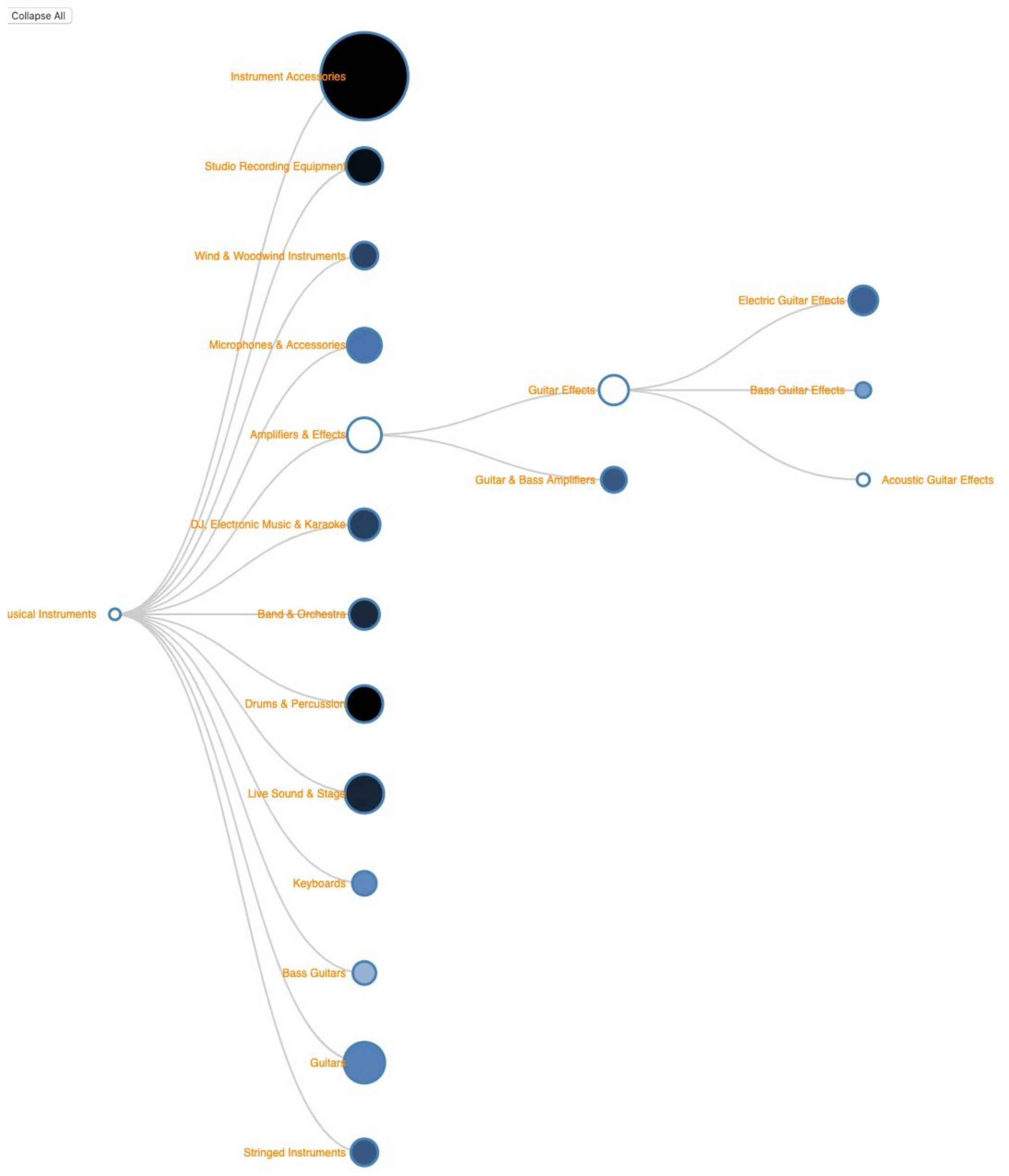


For this implementation, we represented the hierarchical relationship of the categories and this did a moderate job doing it by using color to encode the same level categories. However, because we tried to avoid occlusion, the dots were very spread out and the hierarchy was not obvious. This plot did a relatively decent job at representing the categories, but was not good at representing the quantities of subcategories when compared to others. Therefore, we decided to use areas as encoding and produced the following.



Even though this iteration looks visually appealing, it failed to truthfully represent the data because the colored rectangles were used as containments of the scattered dots, rather than using their sizes to reflect the number of subcategories. Therefore it could be misleading to the viewers. We decided to start from scratch and approach from a completely different angle by using a more traditional tree.

IMPLEMENTATION



Design Decisions:

For our first task, a tree is a great way to represent the hierarchical relationship. With our previous prototype, we realized we can use positions to show the hierarchy. For instance, by

aligning the categories of the same height, it automatically shows which categories have the same depth.
For our second task, we used fill to represent the number of children a node has. We further expanded our idea by using the intensity/illuminance of the color to represent the amount of children it has.
For our third task, we revisited our original plan of using node size to represent the amount of subtree products a node has. Since the differences between the subtree product counts are very significant, we normalized them using cubic root and scaling factor.

Code:
D3.js library provided us with useful abstractions for basic tree implementation. However we had to process CSV files into nested javascript objects using a recursive algorithm so that the data are compatible with D3’s tree data structure. Within the same recursive function, we calculated new fields such as node depth and number of subcategories to be encoded by size and color.

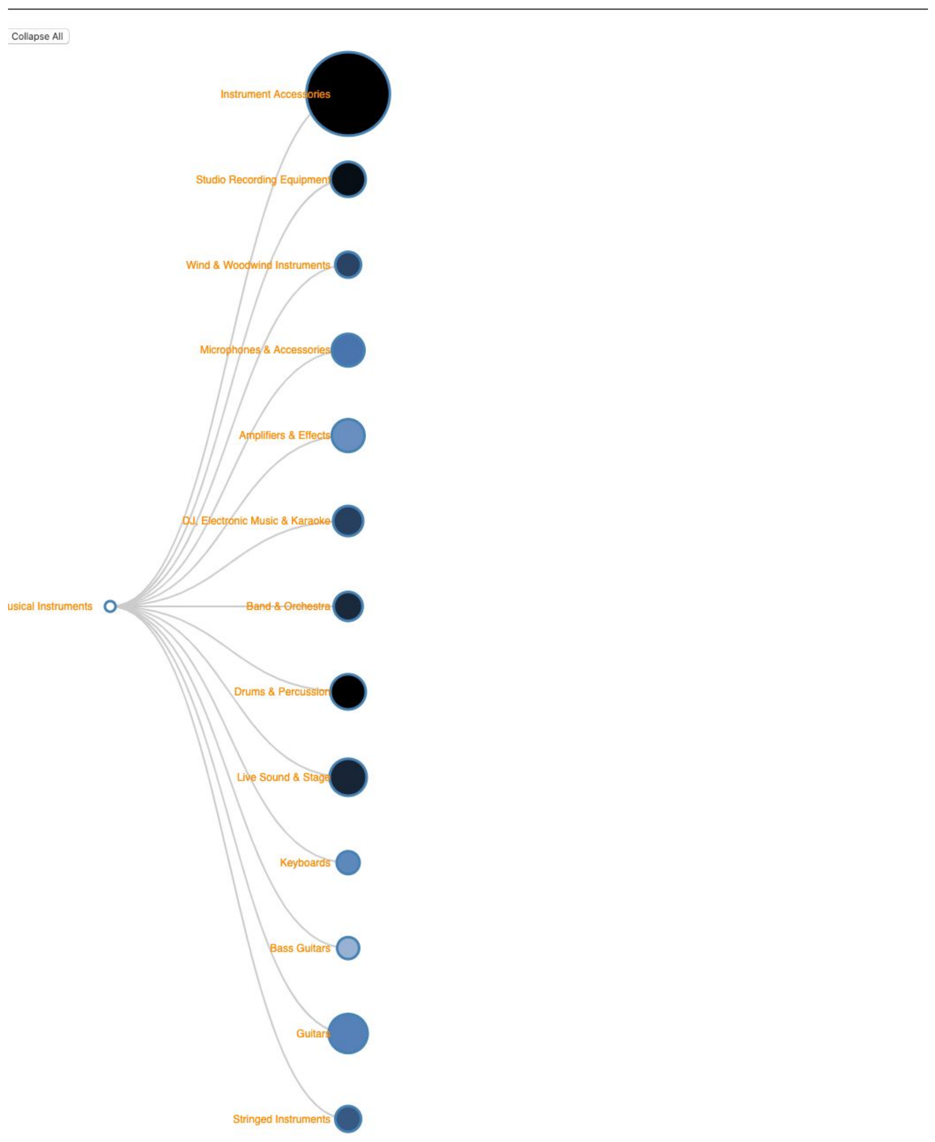
For canvas setup, the program uses an adaptive approach to determine the height of the svg plane with the number of direct relatives the root node has. This way, the vertical space between nodes are consistent throughout different datasets.

The data will then be propagated to form our visual representation. The code maps the square root of subtree product count to node radius so that the area is directly proportional to the subtree product count. It also maps the the number of subcategories to the luminance of the node with a deeper tint representing a larger number. Both size and color are normalized so that occlusions are minimized and each value is distinct in the color range.

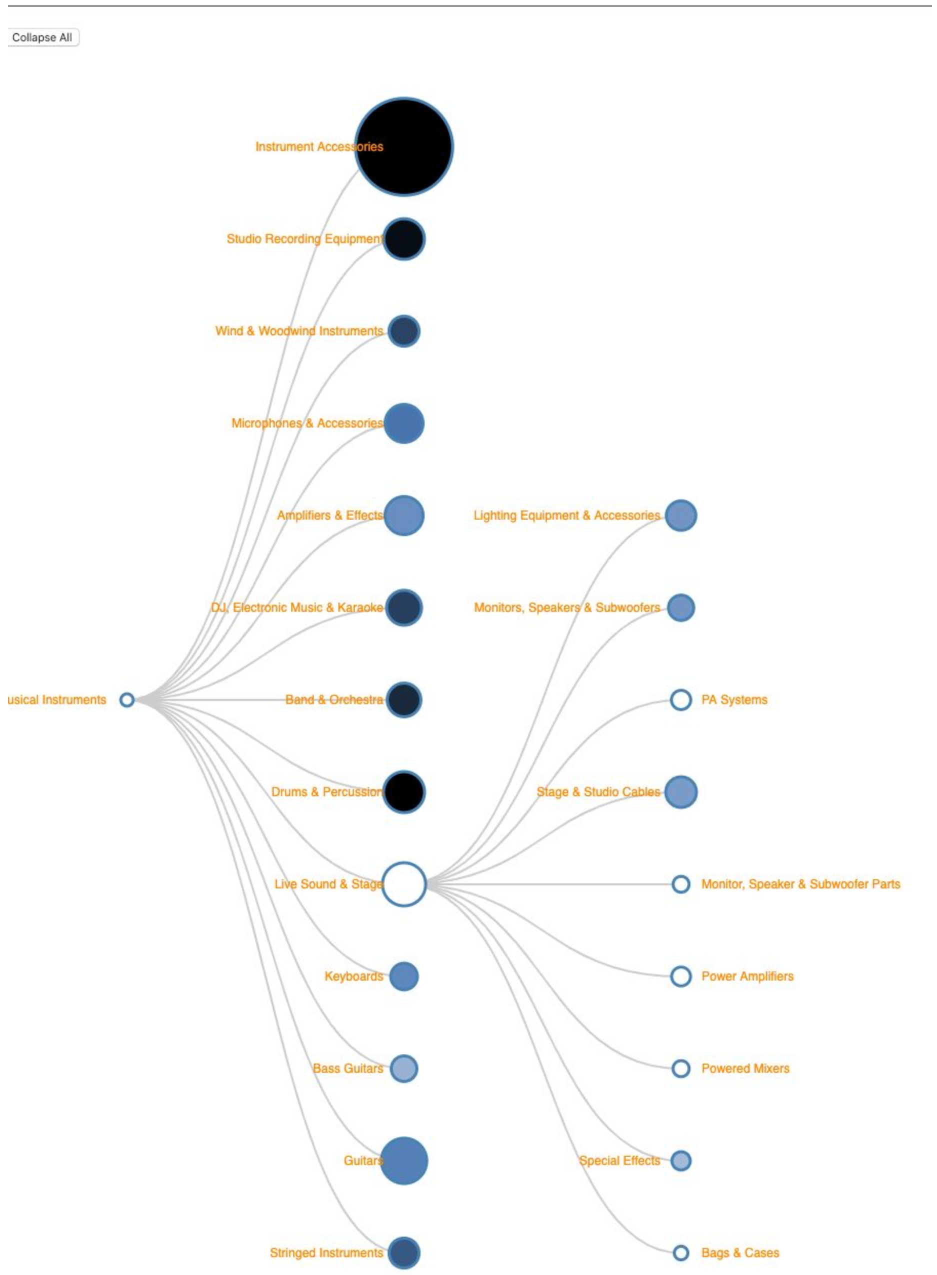
DEMO + EXPLANATION

Link to demo video: https://youtu.be/0Uk2PDgQyUs

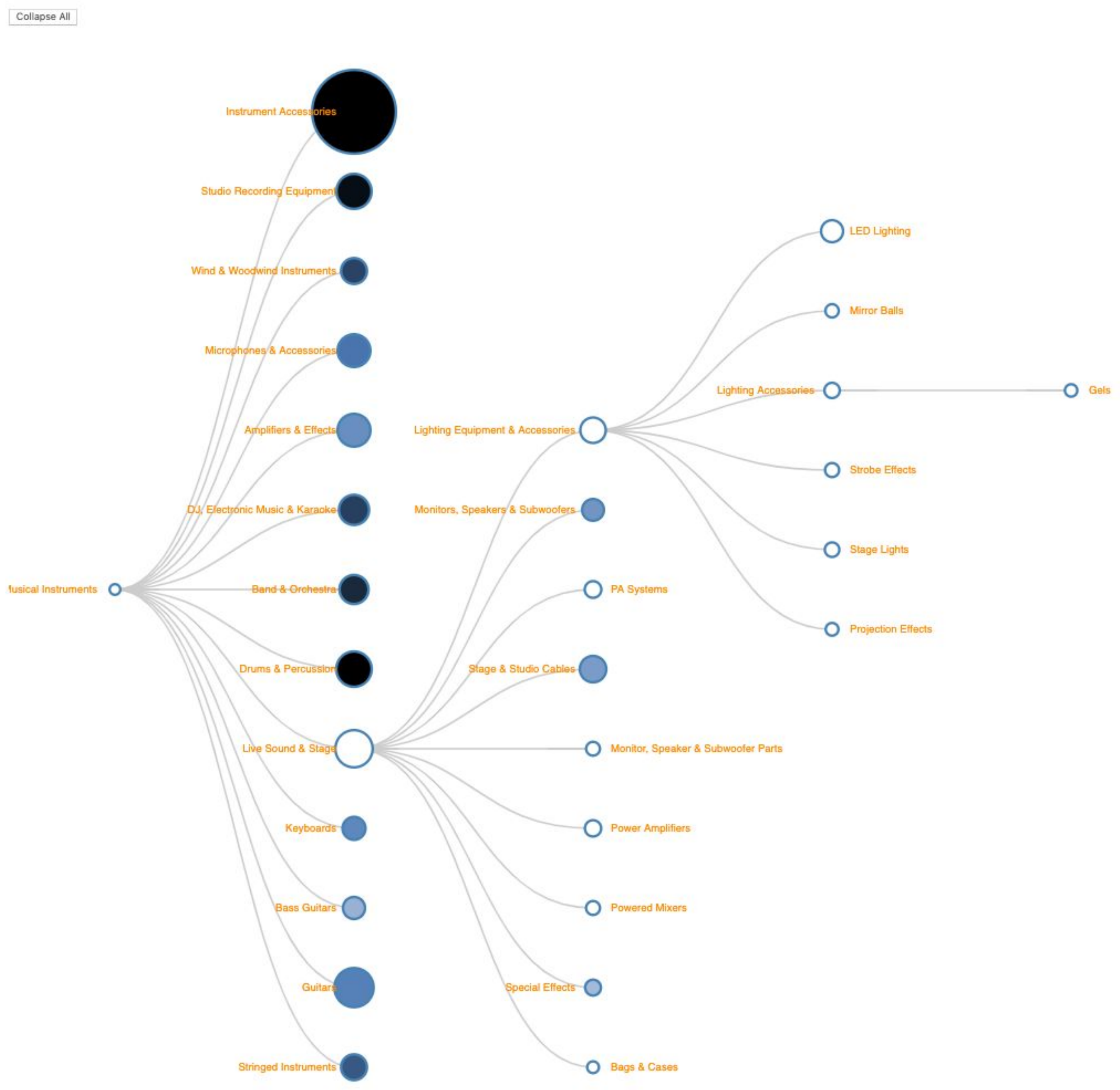
The first page of our program. As mentioned previously, the luminance of node color represents the number of subcategories and the size of a node represents its subtree product counts. When a node is filled, the node has at least one child and user may only expand the ones that have children.



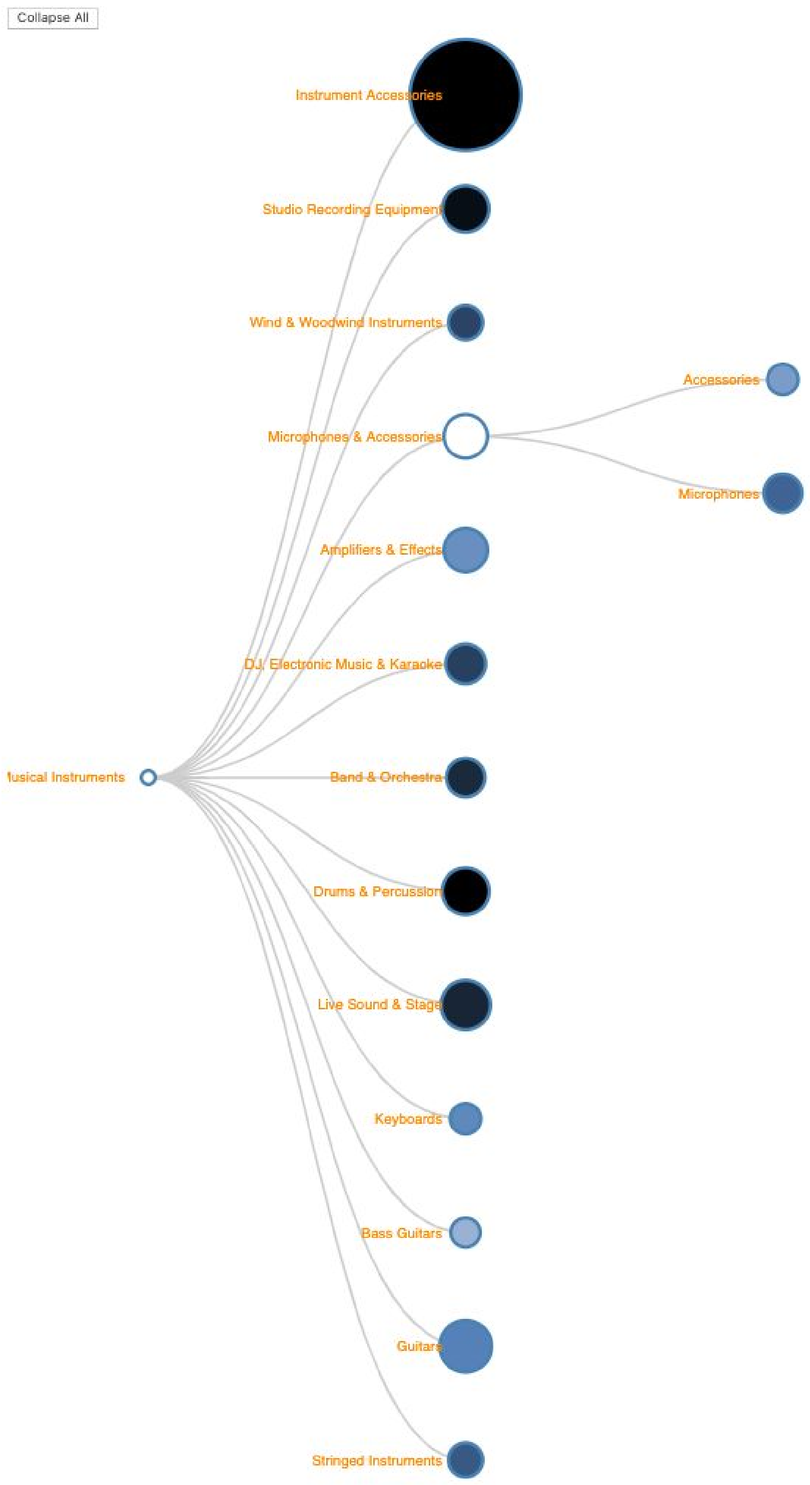
Once a user clicks on a particular node, the selected node expands to the next level while the other nodes re-adjust to fit the content on the screen as the following. In addition, if a node does not have a child, the title appears on the right of the node while the ones that have a child appear on the left for users to select.



The user may continue to further investigate the categorise by clicking and the nodes re-renders every time a new node is selected to make sure the user interface is optimized.



If the user wishes to investigate another node, to avoid occlusion, the current branch of nodes automatically unfolds and the node is expanded. The animation can be found in the link provided in the beginning of the report.



A more complicated task the visualization tackles is to provide insights of how well the products are categorized. Two extremes might be found with certain datasets: large, light nodes and small, dark nodes. When there is a large, light node, it contains many products but few subcategories, which could be an indicator suggesting a more extensive categorization of the data. A small, dark node means the opposite and may result in too few items within certain categories for the customer to view and compare or for the vendor’s suggestion algorithm to make recommendations of related products.

CHALLENGES

In order to generate a tree, we researched several libraries and decided to write our program in javascript using D3.js. Neither one of us has any experience with the library and we had to learn from scratch by watching tutorials and reading through documentation. The first challenge was to learn the library and program visualizations in general. Once we got something on the screen, the next challenge was to refactor the data so that it would be compatible with the tree data structure provided by D3. We designed several different recursive functions to do so. The next challenge was scale. Displaying every single node at once was ineffective since the large amount of categories caused severe occlusion. As a result, we needed to limit the number of nodes being displayed and provide a way for the user to acquire more data on the categories of interest. Also, wanted users to be able to interact with each node and explore on their own. Understanding the trigger functions and updating the data accordingly was also something we put a lot of work into.

LIMITATIONS

As mentioned previously, we used CDN (Content Delivery Network) to implement the D3.js and Cola.js libraries. We did so to reduce the file size. As a result, this program requires an internet connection. Another limitation is the format of the data provided. The program only accepts data with certain keys contained: id, name, product count, subtree product count, parent, number of children, a list of children. Without the necessary keys or false information, the tree may not display as expected. Another potential limitation is its scalability, which will be discussed in the following section.

SCALABILITY

The visualization works well with large files in general but it might not perform as expected when encountering significantly large data. For the data that we have, *Books.csv* (4,300 entries), *Musical Instruments.csv* (666 entries), *PetSupplies.csv* (554 entries) and *all-nodes.csv* (29,240 entries), the program is able to handle the first three with ease. For the 29,240 entries, the program took a few seconds to process and the nodes were slightly overlapping, but the user would still be able to interact with the data. We assume interaction could potentially be difficult with more than 50,000 entries, or if specific nodes have too many children at the same level. In brief, computationally, this program is able to handle a very large amount of data, and practically, this program is can handle somewhere between 0 - 25,000 entries without affecting much user experience.

CONCLUSION + SELF ASSESSMENT

To summarize, we successfully achieved the tasks we set in the very beginning. We did so with the help of D3.js and Cola.js. Even though we did not have any experience in either library, we were able to get comfortable using them and eventually created an interactive visualization. We solved the first task of hierarchical relationships with tree structures. We solved the second task of subcategory representation using the color of the nodes. We solved the third task of subcategory product encoding with the size of the nodes. And we solved the fourth task of user interaction by adding buttons, clickable nodes and more. We put a significant amount of time and effort into this project and we are pleased with the result.