CS 765 Data Visualization
Design Challenge 3

Arpit Jain, Srujana
{ajain74, srujana}@wisc.edu

This design challenge aims at having a graph layout for the dataset provided by showing some meaningful and interesting insights using node-link diagrams (graph layouts).

Motivation:

Since we worked on building a visualization pipeline in DC2, we are aiming at building a better visualization using graph layout for the given large dataset based on continuous evaluation.

Approach:

We have built an interactive node-link web application which helps us to visualize the large dataset as a node-link graph. We have tried to show the path we followed to come up with better and scalable visualization.

Libraries:

Python 3.7, pandas, NumPy, plotly, graphviz, pygraphviz, dash

Dataset:

books.csv
pet_supplies.csv
musical_instruments.csv

Source Code:

https://github.com/calvincodes/cs765-dc3
https://github.com/SrujanaN/DC3
We have different repo for web applications and for static analysis.

Website Link:

https://cs765-dc3.herokuapp.com/

How to Run?

Local Env

1. Clone the repository

git clone https://github.com/calvincodes/cs765-dc3.git

2. Change directory to the cloned repository.

cd cs765-dc3

3. Execute the following commands.

Setup

ONE TIME setup to create a venv and install required dependencies.

source setup.sh

Running application

Once setup has been done, use this command to start the application. Note that you do **NOT** have to run setup again.

python3 app.py

Introduction

In this project, we were provided with a huge dataset of categories linked to each other. We followed an iterative approach, and developed 3 designs in total to visualize this dataset. Each design is an attempt to address some of the challenges faced in the previous design.

In the following sections, we will explain all our designs in detail.

Design 1 & 2: Part-whole relationship using Sunburst diagram and TreeMap

Critique

The intent of this design is to show part-whole/ hierarchical relationship between categories.
We have chosen sunburst diagram and treemap for the same.
Sunburst Diagram and TreeMap are almost the same since both of them exhibit part-whole / hierarchical relationship between the entities. They vary with each other with respect to visuals where sunburst diagrams are radial and TreeMaps are rectangular. In case of Sunburst diagram, the root node is at the center and the hierarchy grows far apart from the center showing the levels. But in case of TreeMap the root node is outermost rectangle and hierarchy grow deep down inside as shown.

There are a few **advantages** to using the **sunburst** chart over the treemap. It's easier to see multiple layers of data with the **sunburst**, while the treemap is better for comparing categories within the same hierarchical layer.

Sunburst Diagram:

It follows self-adaptive sunburst algorithm where nodes are allocated their areas according to their attribute value, and siblings of the same parents are made in ascending order according to the size of areas, adjusting the position of sectors.

We created interactive sunburst for this dataset, where the user can click on any of the nodes and the sunburst expands with the clicked node as the new center. This allows users to navigate through the visualization. But given the amount of data we have, the scalability of sunburst diagrams becomes a bottleneck in our case.

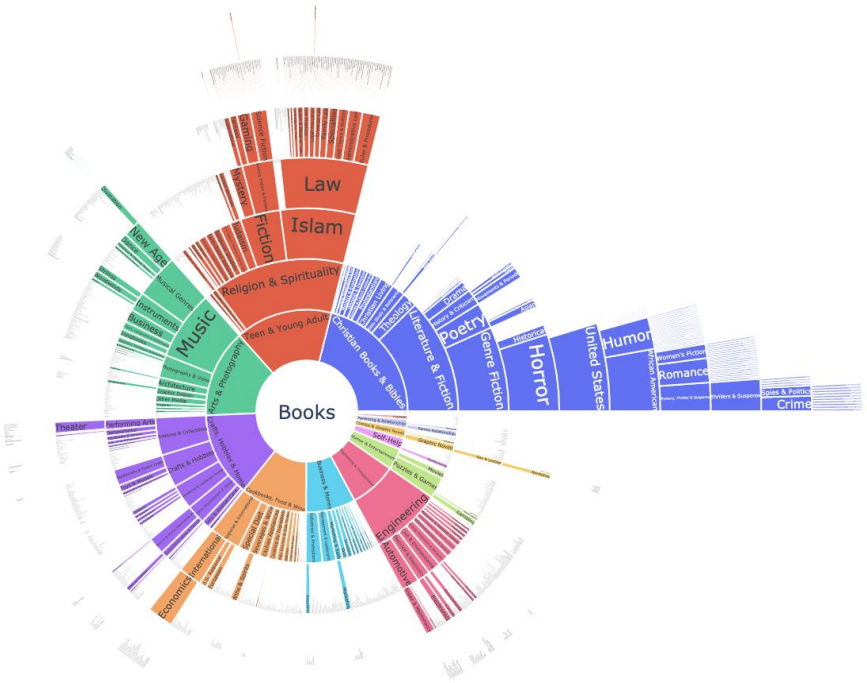
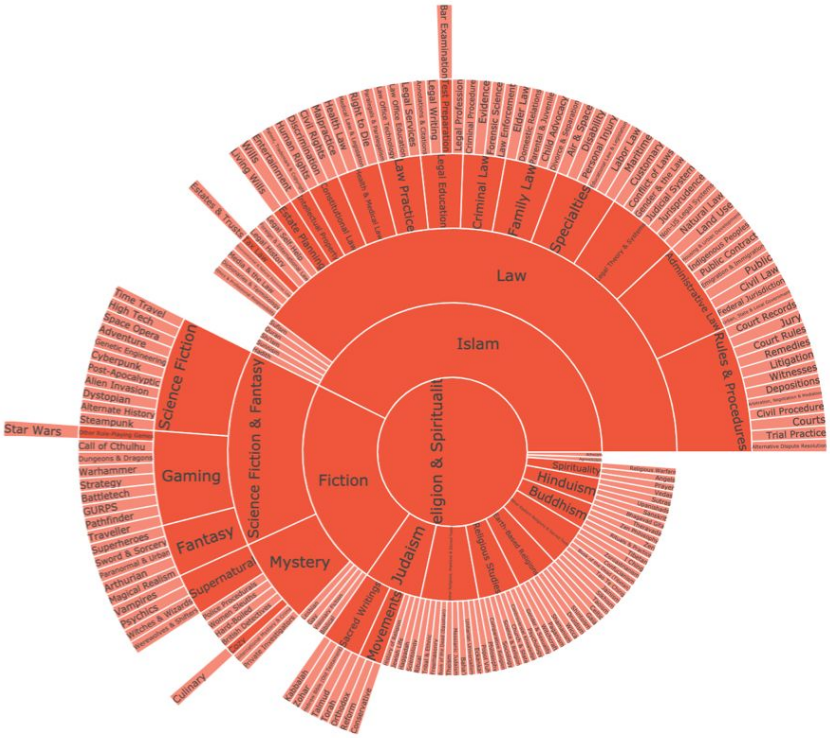


Figure 1: Sunburst Diagram for Books dataset.

The below figure is obtained when clicked on Religion & Spirituality segment, here 'Religion & Spirituality' becomes the parent node and adapts itself as shown below. And also we can navigate back to the root node by clicking on current root node until we reach the actual root node.



TreeMap:

This is a Squarified treemap. In the standard treemap, it is a thin, elongated rectangle. As a result, rectangles are difficult to compare and to select. In Squarified treemap, the layout is in the form of rectangles with approximate squares. To strengthen the visualization of the structure, shaded frames are used around groups of related nodes.

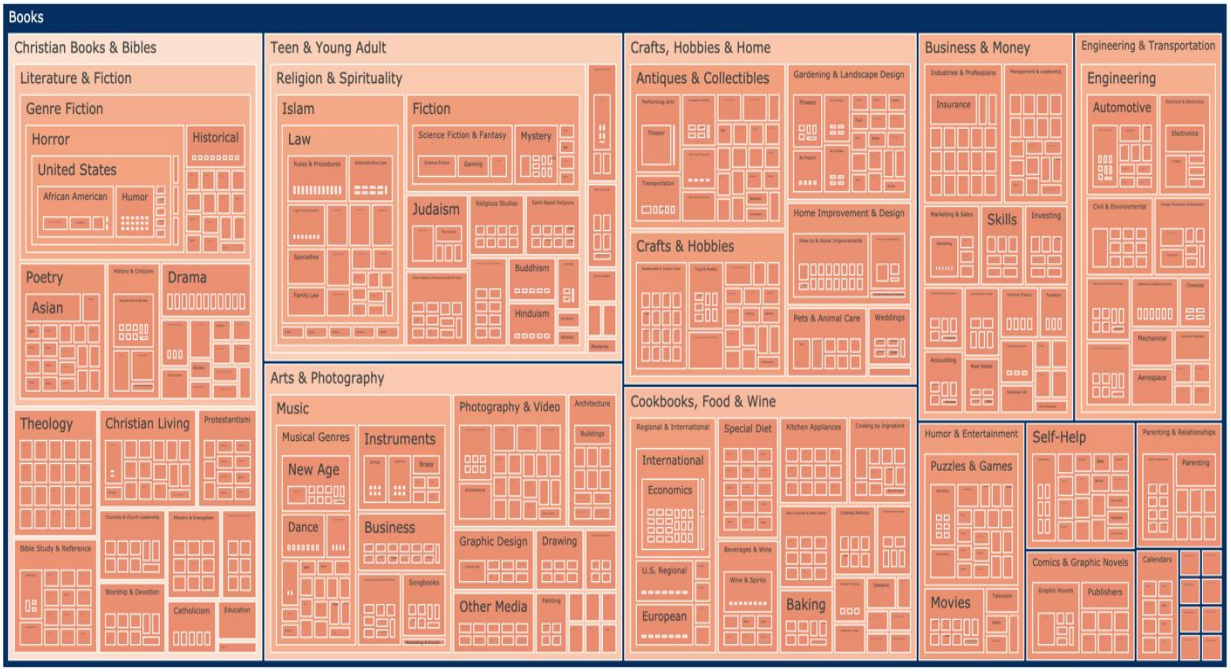
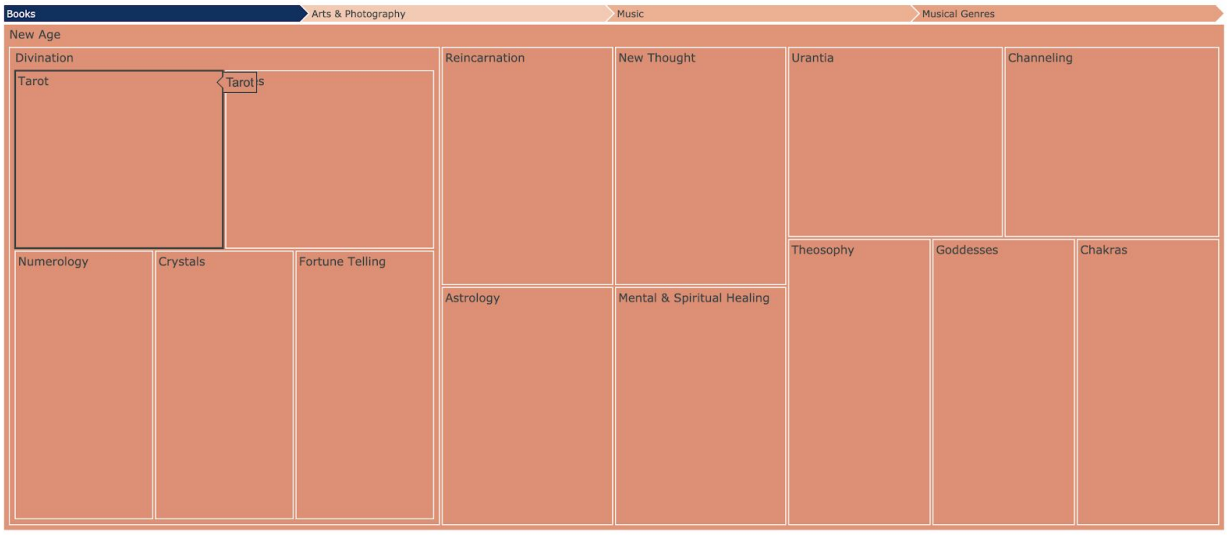


Figure 2: TreeMap for Books dataset.

We have created an interactive treemap which allows us to navigate between categories and also to an extent displays the part-whole relationship even after having shared by multiple categories. The below treemap has a navigation bar which lists the hierarchy and can be used to navigate back to the root node.



Both Sunburst and TreeMap have color encoding and data is abstracted as category by showing a hierarchical relationship.

The book dataset provided doesn't show the exact part-whole relationship since it contains an overlap of categories and subcategories. To organize such dataset, we have to decide on how the relationship has to be established, either we have to choose by placing a category based on granularity or relevance. Coming up with such a design proposal is complex and might leave out some meaningful information. Hence proceeding with TreeMap or Sunburst is not a good idea. To analyze such dataset with overlaps we have come up with an application which help us to look into the data based on request.

Limitation & Challenges:

1. With the help of open source visualization techniques like sunburst diagram and treemap we were able to establish part-whole relation to an extent but we were unable to achieve to the fullest.

2. We have ***developed a custom algorithm of our own*** to show the hierarchical relationship on top of NetworkX which was quite challenging and time-consuming. Demonstrated in the next parts of the report.

Design 3 & 4: Interactive Node-Link Web App

We have built an interactive node-link web application which helps us to visualize the large dataset as a node-link graph.

Step by Step Walkthrough

The application has a pre-built sample dataset and allows users to upload custom data as well (more on this later). The application has various tabs for specifics tasks using network graphs, and we will go through them step by step, trying to adhere to our intended goal. Network graph helps us to find the dependencies and meaning connection within the data.

Landing Page Tab

Description

The landing page provides an overview of the uploaded data as a Network (node-link) graph with category and level filters alongside.

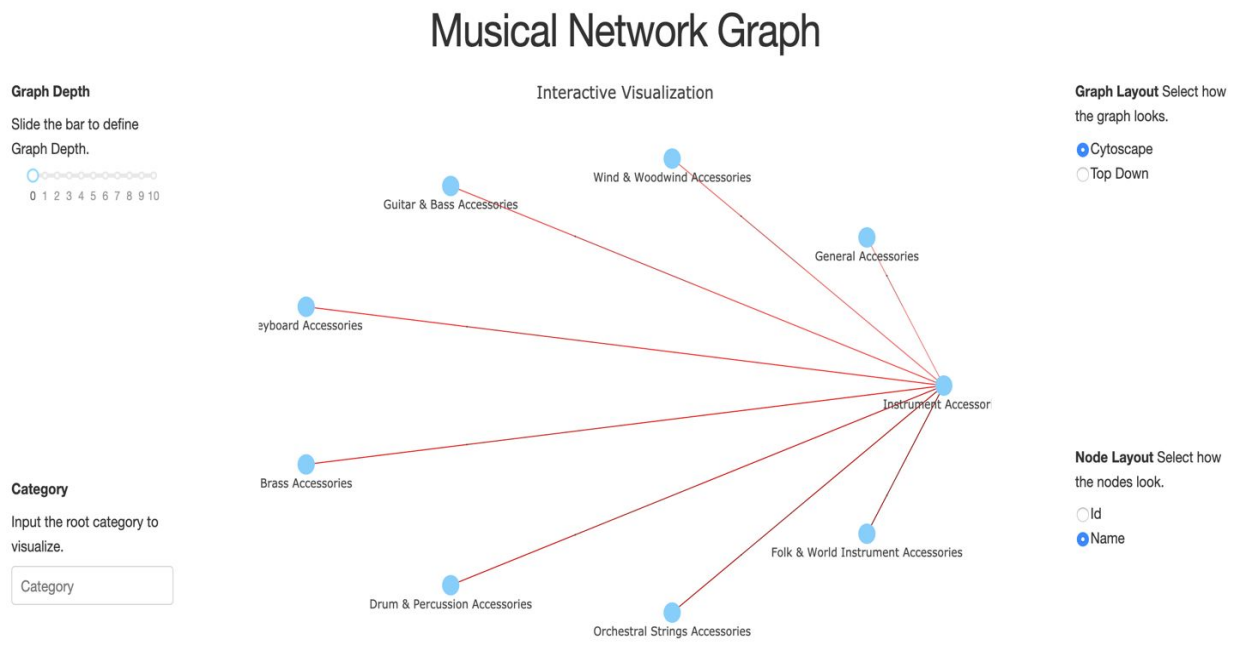


Figure 3: Landing page of the application.

When the user makes changes to the RangeSlider or the Input box, the Plotly figure will change accordingly. When the user hovers or clicks on the node or edge in the Plotly figure, the Graph Layout and Node Layout radio buttons are used for better look and feel of the node-link diagram.

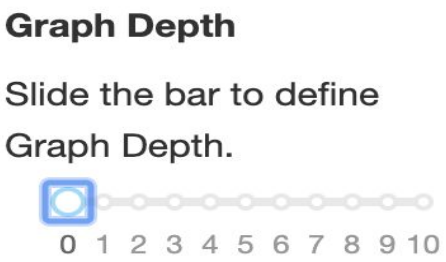


Figure 4: Range Slider used to set the depth of the graph

Figure 4 shows the slider which is a filter used to visualize the network graph based on selected (requested) graph depth.

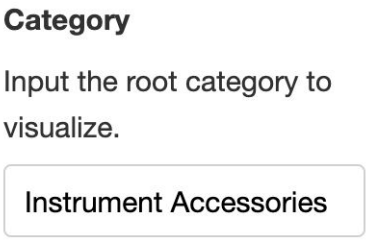


Figure 5: Input Box used to check the hierarchy based on the input root node.

Figure 5 shows the search box, which is also a filter used to visualize the network graph based on the requested root category.

Designs and Intents

The intent is for the user to get a high-level overview of the uploaded data. The design should present these attributes and consider the scale of each attribute as they can significantly differ from each other. The category filter helps us to visualize the specific category and its associated sub-category as a node-link diagram. Slider helps to visualize the category based on the chosen granularity level.

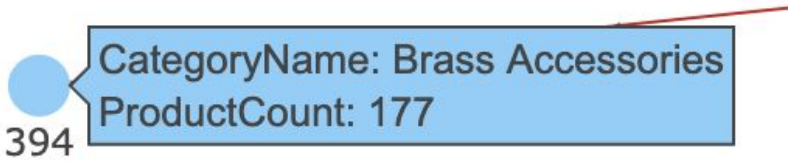


Figure 6: Node with node ID, node name.

When you hover/click a node, it displays node details like node name and number of products in that category as shown in figure 6. The first node is represented using node name and the second node is represented using node ID. Second approach can be used when we have space constraint to avoid overlap between node name which requires some extra effort but works feasible.

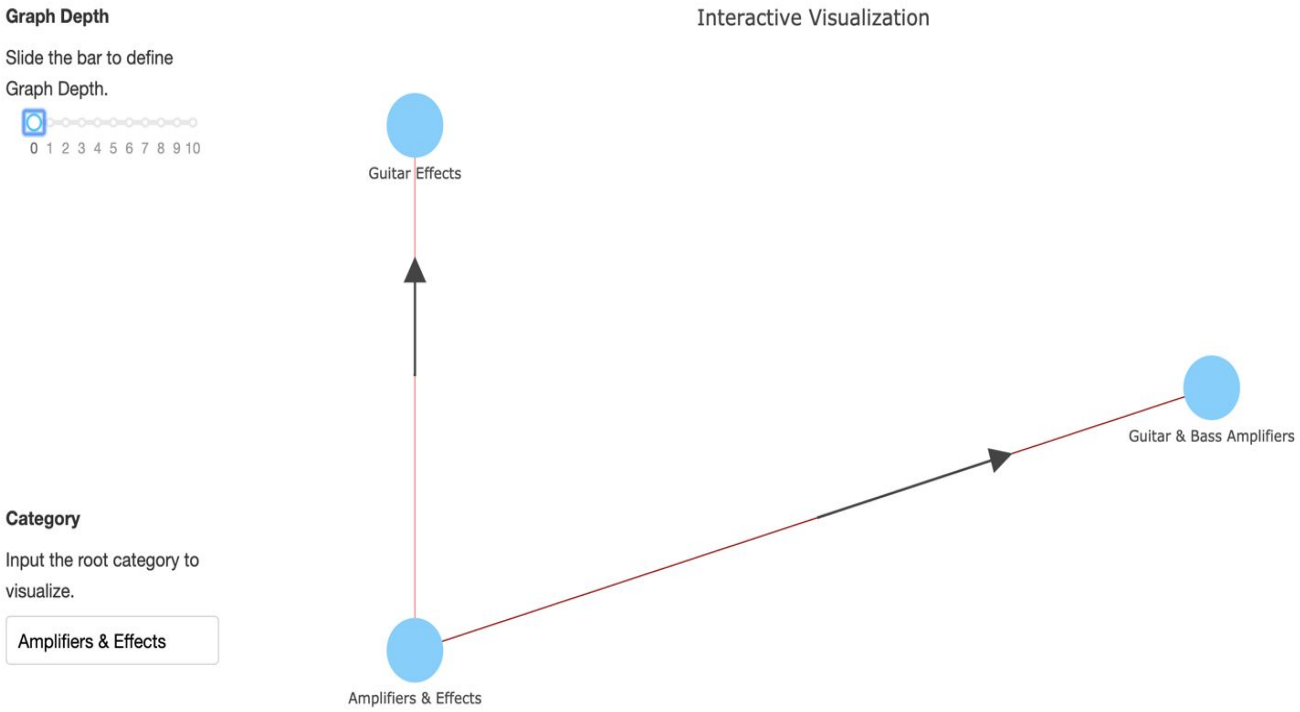


Figure 7: Node-Link Diagram for "Amplifiers & Effects" with depth 0.

Figure 7, we have generated a node-link diagram with "Amplifiers & Effects" as the root node with graph depth '0', it has two child nodes "Guitar Effects" and "Guitar & Bass Amplifiers". A node with an outgoing edge indicates the parent node and node with an incoming edge is the child node.

Figure 8, we have generated a node-link diagram with “Amplifiers & Effects” as the root node with graph depth ‘1’, Here “Guitar Effects” and “Guitar & Bass Amplifiers” each have 3 child nodes. Here we have to scan through the node-link diagram to understand the hierarchy.

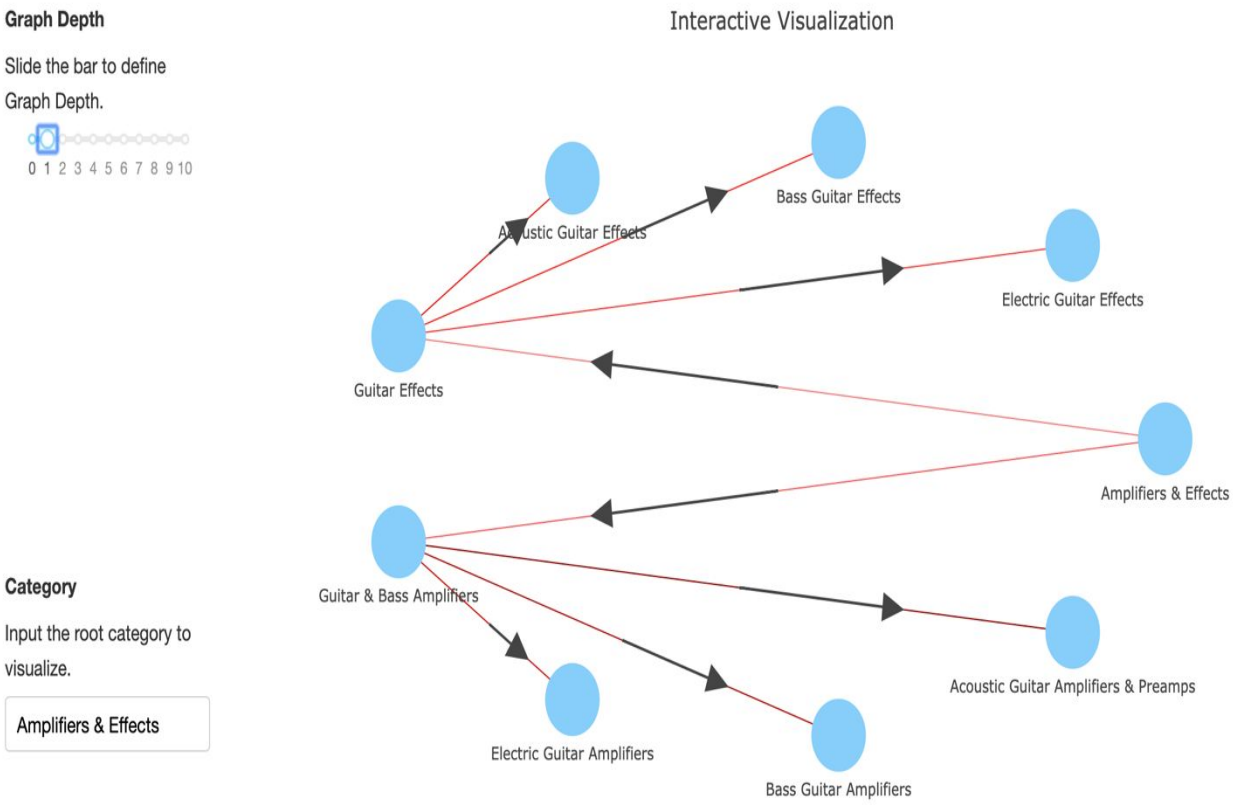


Figure 8: Node-Link Diagram for "Amplifiers & Effects" with depth 1.

But from Figure 9, we can observe that as the depth of the graph increases, nodes and edges overlap and becomes hard to analyze the data based on hierarchy. In this case, we have decided to go with a top-down approach for node-link diagram for readability purpose as shown in Figure 10.

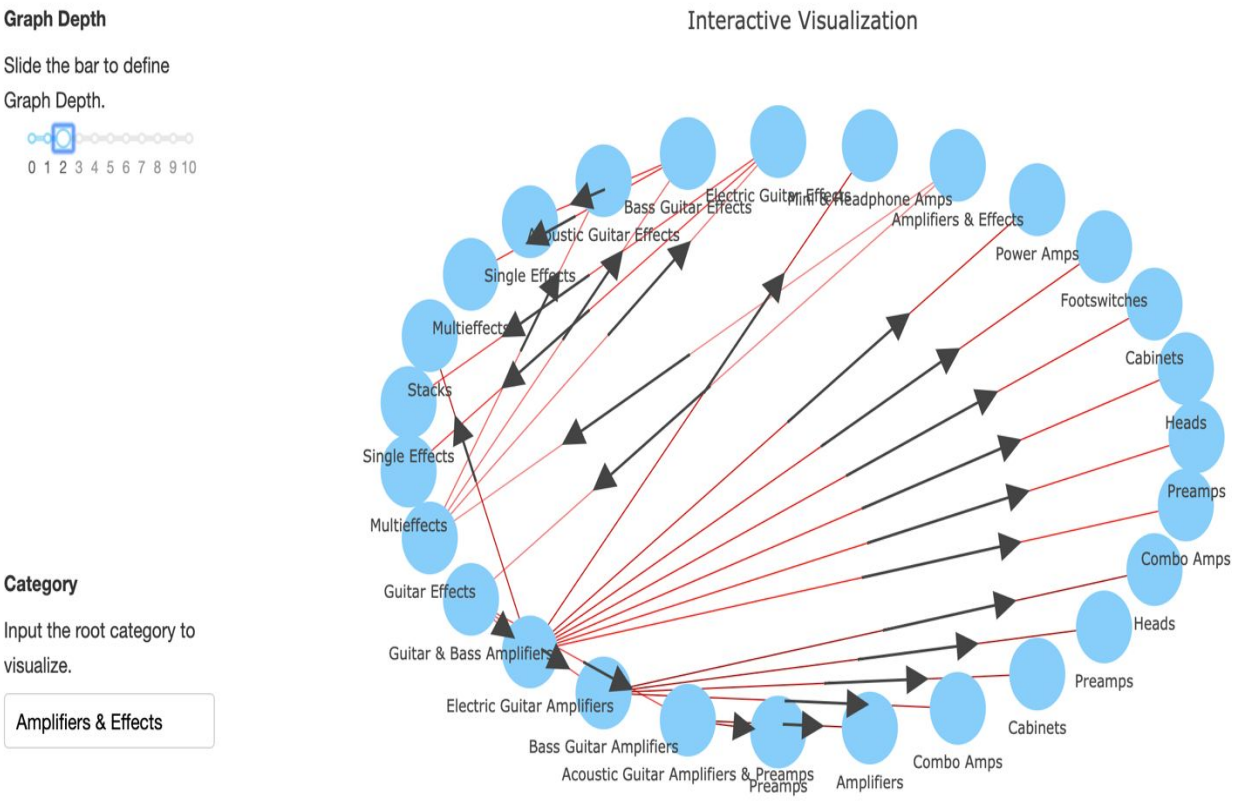


Figure 9: Node-Link Diagram for "Amplifiers & Effects" with depth 2.

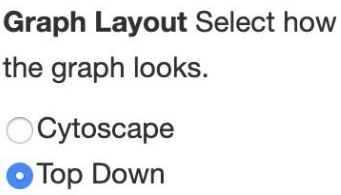


Figure 10: Radio buttons to choose between Cytoscape and Top-Down graph layout.

Graph layout radio buttons allow the user to choose between concentric circular graph layout (Cytoscape) or regular top-down graph layout as shown in figure 11.

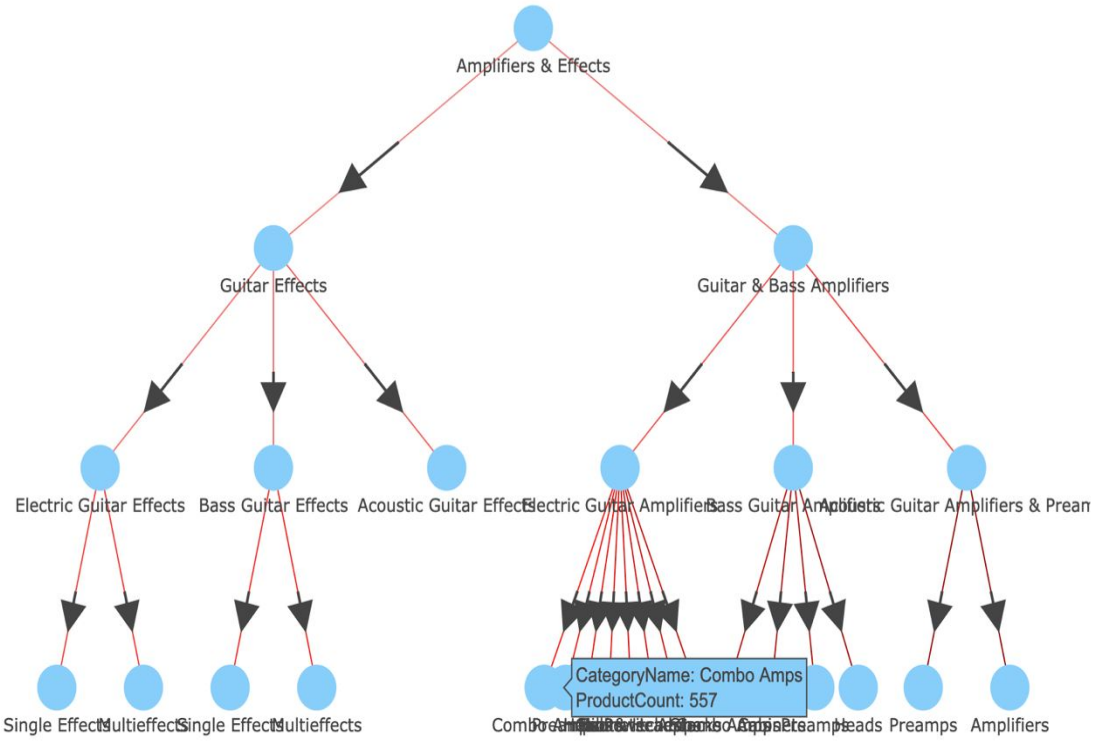


Figure 11: Node-Link Diagram for "Amplifiers & Effects" with depth 2 (top-down).

Figure 11, the top-down layout of the node-link diagram is more organized, but this design has its flaws. Since the nodes are not spread across space, node names get overlapped with each other leading to poor readability.

Node Layout Select how the nodes look.

(•) Id

( ) Name

Figure 12: Radio buttons to choose between node ID and node name.

Figure 12 Node layout radio buttons allow the user to choose between node ID and node name to overcome the issue faced with previous design.

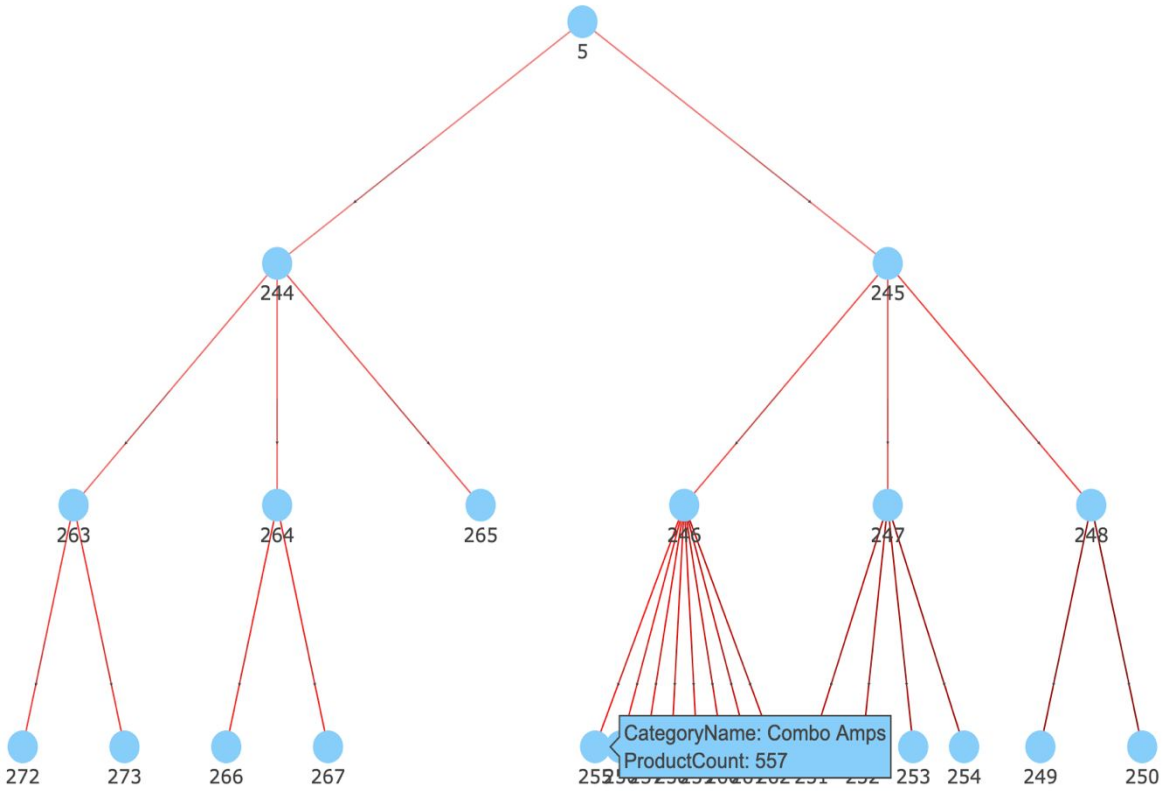


Figure 13: Node-Link Diagram for "Amplifiers & Effects" with depth 2 (top-down) with node ID.

Figure 13 overcomes the issue faced due to node name overlap by having node ID instead of the node name. This can be achieved by just switching the node look radio buttons based on the requirement.

Even after introducing various node-link graph layouts and visualization techniques scalability is still an issue which is to be resolved. After thorough investigation and analysis, we have come up with adjacency matrix also known as heat map used to address the scalability issue.

The specialty of the heatmap is that it takes a non-contiguous dataset and displays them as contiguous. It also shows the intensity of the data connection when placed together. The most important issue solved by heatmaps is the overlap. We often miss the information due to overlaps but with the help of heatmaps the intensity of the data is encoded with color and it shows the intensity of similar data points without overlap by highlighting the intensity of the color.

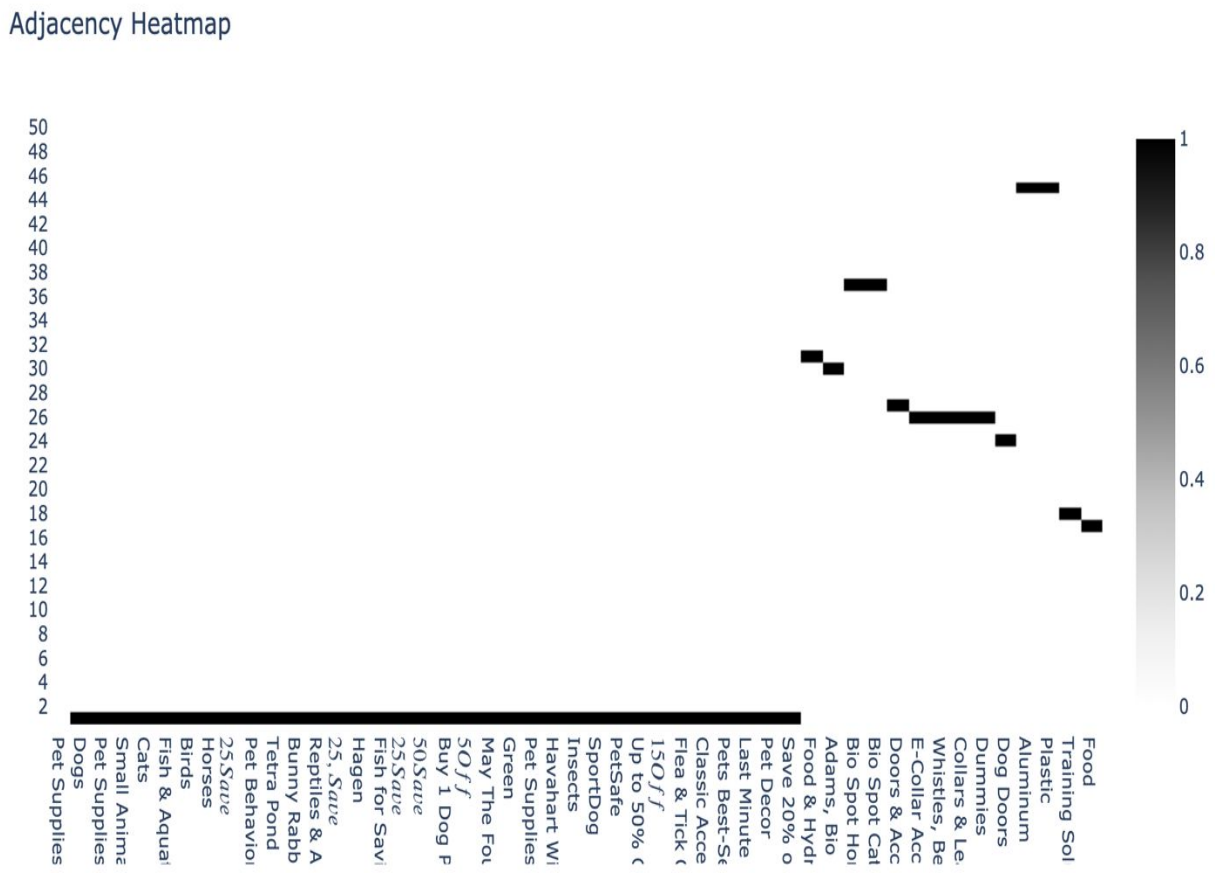


Figure 14: Adjacency Heat Map for first fifty categories to address the scalability issue.

Figure 14 shows the adjacency heat map for first 50 categories of musical instrument dataset and we can see that it scales perfectly well without overlap. With the help of hover over the facility, it becomes even more easy to get to the inner details of the node and the relation.

Scalability

Scalability of an algorithm and computer hardware to compute node-link diagram is a major issue. Since it is difficult to read 'large' and 'dense' graphs, interaction techniques are added to navigate within the network graph. In our case, we are using interaction provided by plotly framework plus customized filters like search box (leftmost bottom-corner) which can be used to search for particular category based on parent category and also displaying the depth of the graph based on request using range slider (leftmost top-corner).

The major scalability is finally solved with the help of the adjacency matrix also known as heatmaps. They scale well with large dataset without overlaps.

Limitations:

1. Scalability is the major issue with ‘large’, ‘dense’ dataset.
2. As the number of nodes increases, nodes overlap with each other invalidating the purpose of readability.

Challenges & Learning:

1. We focused on creating a node-link diagram similar to what Prof. Mike showed us in the class, which was very well structured to scale with large dataset, but we were unable to achieve that because of the time and space constraints.
2. Working with large dataset was quite challenging since it was hard to display everything by making a note on space constraint.
3. Added numerous filters like a range slider, search box to overcome scalability issue.
4. Came up with multiple graph layouts which work best based on the situation.
5. Changed the look and feel of the nodes to avoid node overlaps.
6. Experimented with various visualization techniques to decide which works best for the dataset to show the hierarchical relationship.
7. We are not a regular Python developer and hence spent a good amount of time getting accustomed to Pandas Dataframe APIs.
8. We spent over a day researching what all Python libraries which can be use for this project. We shortlisted Plotly, Bokeh, Dash, graphviz etc.
9. Finally, we went with plotly, dash, and networkx and hosted the webapp using Heroku.
10. We spent a lot of time on investigating on the kind of visualization technique since dataset provided was not as easy as we expected.
11. We tried with TreeMap, Sunburst Diagrams, Regular node-link diagrams and adjacency matrix.
12. Scalability was the major issue we faced as we started working on large dataset with thousands of categories.
13. Issue is finally resolved to an extent with the help of adjacency matrix.
14. We went through a number of visualization papers [2] which finally led us towards adjacency matrix.
15. Networkx does not provide any default hierarchical visualization of trees. Hence, we wrote custom code to position the nodes in order to create a hierarchical top-down view of the graph.

Future Work:

1. We still aim to create a scalable node-link diagram like Prof. Mike for large dataset.

Reference:

1. https://towardsdatascience.com/python-interactive-network-visualization-using-networkx-plotly-and-dash-e44749161ed7
2. http://iihm.imag.fr/blanch/teaching/infovis/readings/2004-Ghoniem-GraphReadability.pdf
3. https://www.sciencedirect.com/science/article/pii/S2468502X18300585
4. https://www2.cs.arizona.edu/~kobourov/NL-AM-TVCG18.pdf