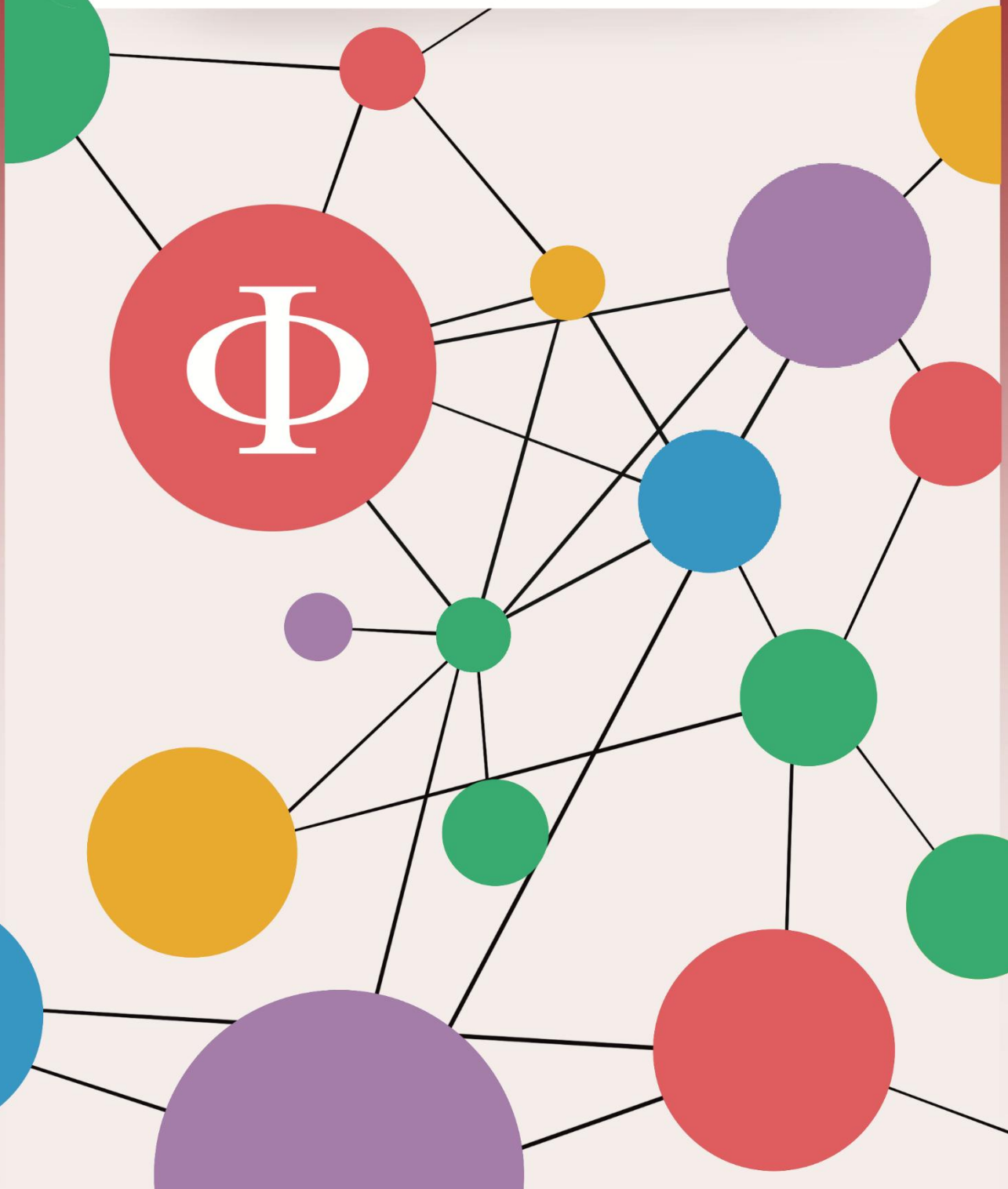
I declare that, except where otherwise indicated, this mini-project is entirely my own work, and that it has not been previously submitted and/or assessed and is not due to be submitted in its entirety or in part for any other course, module or assignment

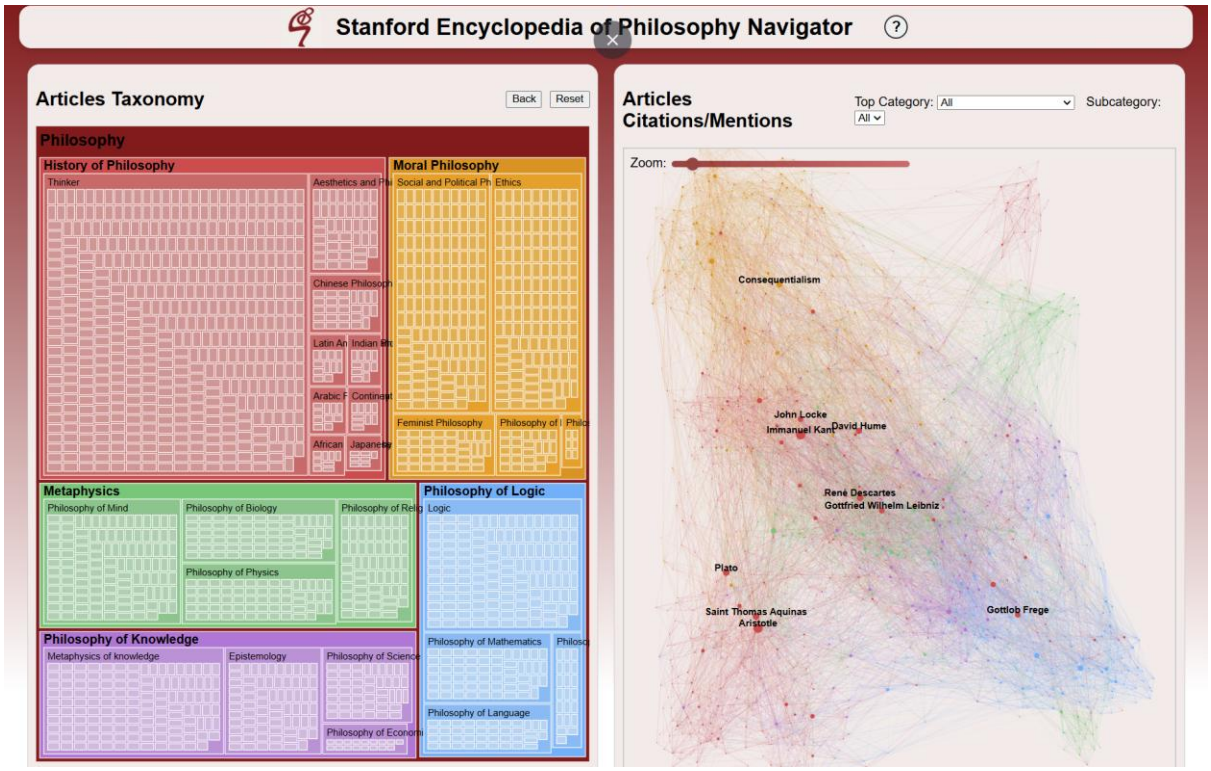
1081049

The Stanford Encyclopedia of Philosophy Navigator



Overview

Teaser:



Stanford Encyclopedia of Philosophy Navigator visualises the taxonomy of the Stanford Encyclopedia of Philosophy articles on three levels of hierarchy, and links between articles- an edge from x to y represents that article x cited y (links to it).

The problem to be solved is the following: there is no clear map of the articles in the SEP to discover at a leisurely pace, and users might want to explore not only the articles by type and topic, but also by how often they are mentioned, or which articles are related.

The visualisations tackles this by offering a treemap of the articles in their hierarchy of categories and sub-categories, as well as a network graph that shows the links between all the articles, both consistently colour coded and can dynamically update, by click or menu selection. Both have tooltips for more information to facilitate exploration.

The intended audience are philosophy enthusiasts who are curious about how the SEP is organised and how the topics of philosophy interconnect. They should know roughly what each category and sub-category means (e.g. epistemology, metaphysics).

Data

URL:
https://github.com/josephdicastro/VisualizingSEP/blob/master/static/sep_network.json

The dataset represents the structure and content of the Stanford Encyclopedia of Philosophy.

In domain language, it has a hierarchy of the categories and subcategories that make up the map of the philosophy discussed in the SEP. Each category and sub-category is named. At the bottom of the hierarchy are the individual articles, with their associated article name, individual ID, word count, primary author name, and URL linking to the article. The dataset also includes every relationship of an article being cited by another article.

In abstract language, it is a JSON file with two roots (dataset type: network). One is called “philosophy” which becomes a hierarchy (tree) by nesting 3 levels: Categories (cardinality 5), sub-categories(cardinality 27), and articles (leaves, cardinality 1,746). All the items from the first root but the leaves have only one attribute, their name. The leaves have the attributes “name” (string), “id” (string), “author”(string),”article_url”(string),”word_count”(Int). The semantics are described above and the answer to which attributes connects to which semantics is self-explanatory. The second root, “links” has one level of depth(cardinality 18,916) and has nodes with “Source” and “target”, showing the source and target of each edge (the relationship between articles) in terms of their “id”. It includes a “target_title” but that is not used. All attributes are categorical with the exception of word_count which is quantitative, which is also not used.

In terms of processing, I took out some of the attributes in the original dataset, and using python I added a level of hierarchy (the highest after the “philosophy” root) using Wikipedia’s map of philosophy I ordered the 27 sub-categories into 5 categories and re-made the dataset in that fashion. All of this can be seen in the “data” folder.

Some processing is done “on the fly” using JavaScript, giving nodes attributes such as how cited a node is (abstract language: the number of links that have the node as the target, or node degree). The range for this is 0-114 (quantitative).

Goals and tasks

-Task 1;

-Domain Task: Explore which articles are linked to each other, by citations.

-Attributes: Link between nodes (directed connection), branch membership (category, subcategory), node name (article title)

-Abstraction Task: Explore the topology of the network by traversing links, and identify connected entities.

-Task 2;

- Domain Task: Compare the number of articles by category and subcategory. Compare which categories in the taxonomy of philosophy the SEP a higher number of articles about.

-Attributes: i) Cardinality of each category or subcategory (number of nodes it contains), ii) hierarchy level (parent, grandparent, root).

-Abstraction Task: Compare values of cardinality in a hierarchical branch at different levels.

-Task 3;

- Domain Task: Identify articles within specific categories and subcategories, and identify which categories and subcategories a given article belongs to.

-Attributes: Branch membership (category, subcategory), node name (article title).

-Abstraction Task: Search and filter nodes by branch membership and identify parent and grandparent information for a given node.

-Task 4;

-Domain Task: Identify which articles are most cited in the SEP, by category.

-Attributes: Inbound links (node degree), hierarchy level (leaf, parent, grandparent).

-Abstraction Task: Locate nodes with the highest values of node degree at each level of the hierarchy.

Visualisation Design

Innovative view

The first view is the Treemap View. This allows users to do Tasks 2 and 3. The users can compare the number of articles in each category by looking at the treemap as a whole and seeing which sections are largest. Each section also clearly shows the names of the sections one level down in the hierarchy. This is to enable the user to quickly see which sub-categories are in each of the broad categories.

The visual encoding for this view is the idiom of treemap. The mark used in this view is area (containment). The channels for each attribute are: size of area for the attribute of cardinality of the categories/subcategories, saturation/brightness for the hierarchical position of a node in the tree, hue for branch membership.

The size of area is proportional to the number of leaves in a branch. Each category area (depth 1) has multiple containments of sub-categories, also proportional to their number of leaves. The container area for the broad categories colours with different hues for different categories, and for each step down the tree, the saturation decreases and brightness increases. To achieve tasks 2 and 3 the user can compare the sizes of the different containments with different hues in order to find out the size of each category in the SEP, then use the saturation/brightness/containment level to see what level a node is at, and find what article is contained within each category and sub-category by clicking on those and expanding, then either reading the titles of the leaves/sub categories or hovering to reveal a tooltip with details.

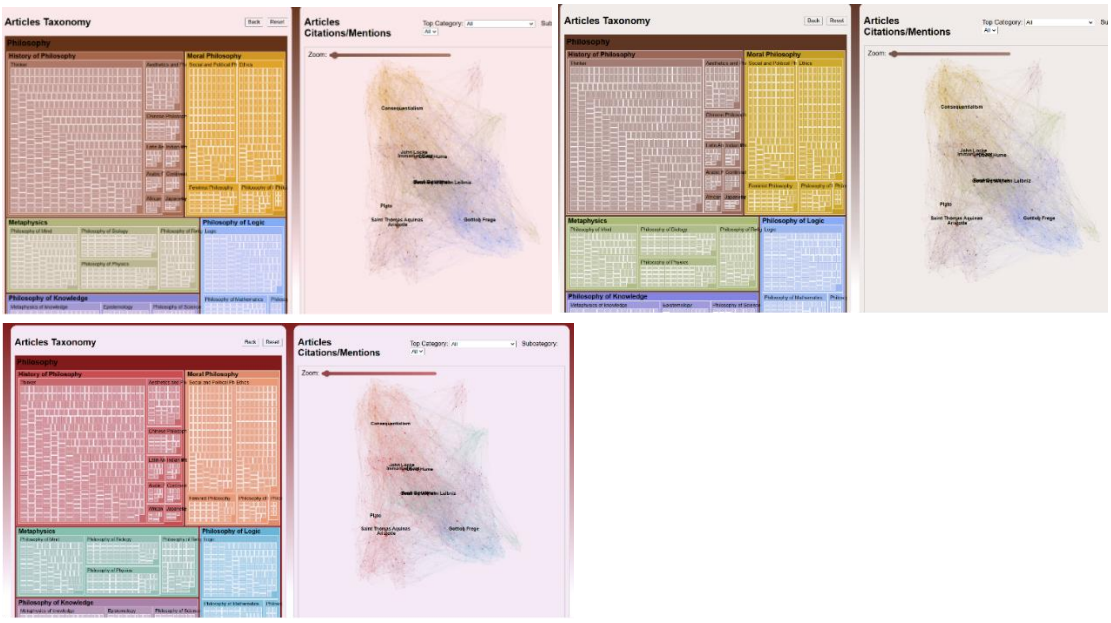
Size of area is easy to compare: in a treemap, the colours are noticeably different and the size of different areas is easily comparable since they are right next to each other or not far off. The only downside is that one cannot compare two different subcategories that are not in the same category as easily as it is not possible to directly position them next to each other manually. However, the aim of the task is to compare broad categories representation in the SEP and then within those categories, the subcategories representation in the SEP. It is less useful to compare the absolute size of two sub-categories in two different categories as they represent different relative sizes compared to their categories, so this is not such a loss. The cost of implementing such a niche comparison use case would be to render the website slower, clunkier, and confusing as to whether it's showing relative size or absolute size.

The colour scale for broad categories is as a "rainbow" type (multiple hues across the spectrum) because it therefore does not imply ordering. On the other hand, depth is coded using brightness and saturation, implying an ordering. The reason why I'm using both saturation and brightness is because both on their own are not enough to be visible enough, and if I were to crank either one more the nodes would become white or

grey, either of which makes them indistinguishable to other article nodes that may not be in the same category. This follows the expressiveness principle from the lectures, which states that channels should only imply an ordering if and only if there is an ordering in the data.

Using the Psychophysics laws in the lectures, for the saturation and brightness changes in the treemap symbolising depth in the tree, I should have marginally increased the saturation channel while increasing the brightness substantially. This is because we perceive changes in saturation at 1.7 times the rate of the actual change, and 0.5 times for the brightness. However, I did a change of 0.1 and then 0.2 for each level of depth for the saturation, and less for the brightness: 0.05 and 0.15. Brightness being lower seems not to follow the psychophysics principles. I did this for two reasons: firstly, the treemap has white borders for each area, naturally increasing the brightness of lower levels since there's a large amount of border per unit area. Secondly, increasing the brightness too much means that when one is in "single node" view (meaning only a leaf fills the entire treemap), if the brightness is too high, the colour is too different from its original: the user might not be able to recognise what it originally belongs to as easily, and might not be able to map it to the equivalent colour in the network graph, which is always the depth 1 colour.

The colours of the Categories were taken using Colorgorical (from the lectures) and I tried to keep a warm palette to fit with the background and the Stanford red, which I used for both some of the background (in the same style as the SEP) and the root colour. These colours are also recognisable and nameable (pink, yellow, purple, green, blue) which increases memory in the user when switching between views and enables them to keep track of what they're looking at more easily to achieve their tasks. They are also differentiable under all anomalous trichromacies:



The leaf node names (labels) are not present unless from the viewpoint of a sub-category or the node itself. This is in order to make the graph free from unnecessary and illegible writing when in root view or category view. This increases the expressiveness of the text position channel as this means that the user only sees the relevant information at any one view without being distracted. The level in the hierarchy determines the text boldness and size.

The interactivity for this view includes a changing of viewpoint, click to access article, and a tooltip. The tree change of viewpoint enable to user to, on click, re-map the treemap so the root is now wherever they clicked. This can be a category or a sub-category. This is a constrained navigation, so this is easier for the user to navigate and the entire domain they selected to fill the space. Another aspect is the click to access the article- if a user clicks on a node directly, it will redirect them to the article page. This is for all tasks: to truly understand the articles at hand for any task, one can read them first-hand. The last interactivity feature is that of the tooltip: this tells you what category/sub-category/article the mouse is hovering on. This website is optimised for chrome on laptop, so hover makes sense here. If a user cannot see the text properly (which happens when in root view or when many articles are in view) the tooltip will give more information on the article + the author. This also helps with effectiveness, as the author is not directly relevant for the user to see unless interested in a particular article, so this makes sense to only include this kind of information in the tooltip.

Secondary view

The second view is the Network View. This enables users to do tasks 1 and 4 mainly, but also task 3. The marks here are points and line (connection). The channels are colour, size, and position (link). Colour maps to the category only (not sub-category or specific article). The size is mostly proportional to the node degree of the node. Position is determined using a force-directed simulation (the edges determined by source_node and target_node). The position of the lines (links) are also determined by this, and are all straight.

The colours that map to categories are the exact same as the ones used for the treemap categories. This choice is hence supported by the reasoning in the previous section, and also serves to preserve the continuity and consistency of the colourings so that the viewer knows what they are looking at. The colour of a link is that of its source node. The colouring of the background makes sure that all the nodes are very visible, and has the same background colour to keep the visual interpretation of the colour consistent via lightness/colour constancy.

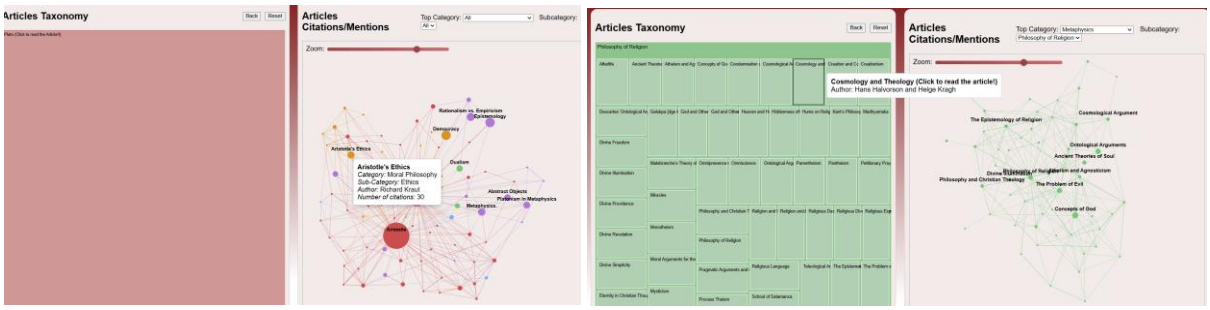
Size is proportional to number of citations (the number of incoming links) unless it is 2 or lower, in which case the size is constant. One would not want an article with 0 citations to be invisible. This helps with task 4, and is accompanied by the label being

positioned for the 10 largest nodes at any time during a view, which is directly helpful to the task.

The position is determined by the links in a force directed simulation. Even though this is very computationally expensive in the “all” view and makes the website a bit slow at times, (as nodes > 1k) this enables the user to quickly see which areas are most interconnected which helps in task 1.

One interaction is the drag in the force simulation, which helps in tasks 1 and 4 as it both enables one to move nodes for clarity (text overlap, hiding). Another is the dropdown menus which use the filter idiom to only show nodes withing a certain category and maybe also only within a certain subcategory, which directly helps with tasks 1 and 4. The program also has a tooltip to display the category in addition to the colour, the subcategory, the author, and number of citations (helps with 4 as comparison of circle area is not precise, especially if far away from each other). Also, if a user clicks on a node, it only shows that node + all nodes linked to or from it. This enables the user to achieve task 1, as when it is coupled with the tooltip on hover, one can look at the names of linked nodes. One can then explore the topology from node to node by clicking on the next one of interest. There is also a zoom bar which enables the user to zoom using the unconstrained navigation idiom (the text is dynamically updated to be of appropriate size, and one can also zoom by double-click or scroll), and a function that zooms automatically using the constrained navigation when clicking on a single node, then allows the user to move, zoom/zoom-out freely.

The views are directly linked: whatever the view is (All, category, subcategory, single node) each view is updated accordingly depending on the interaction in the other view. This enables the user to be able to complete all 4 tasks at any point in their navigation, and keeps the program consistent. This helps handling complexity by keeping the amount of information consistent across both views, complemented by the filtering allowed for both, and the choice of only showing the top 10 sized nodes’ names, using tooltips for extra information on both, and only showing 2 levels of depth worth of names in the treemap. Examples of Single/subcategory view below.



Visualisation Principles

I chose the treemap view to show the hierarchy because it managed to enable comparison without overbearing the user with too much data, and show the hierarchy in a simply digestible way. I could have chosen a node-link tree, an icicle/sunburst diagram, or a grouseflock, since these represent trees (hierarchical data). In fact, the grouseflock would have also allowed to represent the links as well.

I did not use the node-link tree, because this would not have allowed the users to do something only the treemap allows: task 2, or comparing number of articles in each section. This is because a user cannot easily compare number of articles: they'd have to mentally count or approximate the number of nodes rather than just comparing areas like in the tree map. Furthermore, it is hard to colour since the text or the link would have to be coloured, which are both difficult to both see and fully see the colour when coloured.

I did not use an icicle or sunburst diagram for two main reasons: all the leaves in the hierarchy have the same depth, and there is a very high ratio of children per parent node. Since the nodes have the same depth, an icicle diagram which supposedly shows depth becomes redundant. Furthermore, it would emphasise the higher levels too much by giving them a third to each of the levels to the page, whilst they only have information for their name. The sunburst diagram suffers the same problem: with the same level of depth, this gives up the specific use of the sunburst and gives too much space to high level categories. Moreover, the treemap makes use of the extra space is has very efficiently by being a quadrilateral over radial. The radial approach would be a distraction and makes it harder for users to compare areas in outer layers. Treemap is a better idiom than both of those on all fronts (visibility of area, comparison, efficiency of space) for the data I am using.

Finally, I did not use a Grouseflock diagram, even though it would be able to technically represent all of my data into one visualisation, because it would be too visually dense. The links are already very dense (18k links for 1.7k nodes) so in a grouseflock diagram this would likely hide the categories during a high level view. It would further not work for the single view- I would like to show and make clear what node we're viewing and what nodes it's linked to, to help with task 1. However, in the grouseflock idiom, it would be difficult to represent a node and only its connections. If a node in category x had links in some other categories, it would be hard to make the diagram compact. Furthermore, it would increase the saliency of the ones in other categories as they would be very far away (longer links are more salient as per the lectures) and this is introducing unhelpful focus.

Credits

Inspiration from Mike Bostock code for treemaps- I added my own interactivity and animation, but the d3 structure was inspire by how he coded it to make it fit a specific space effectively and change root. https://observablehq.com/@d3/zoomable-treemap

The force directed graph was inspired by https://observablehq.com/@d3/force-directed-graph/2 but majorly changed in almost all aspects except the force-directed equation. I added all the interactivity except drag.

ChatGPT 4o was used for correction of bugs (inputting the code and asking it if it sees a typo/missing brackets etc), and for reminders of things like HTML, CSS and JS/D3.js code meaning (for example, remembering what flex does). No idea, direct implementation, and/or writing was taken from AI generated content.

The rest of the work was entirely done by me/taken from my previous answers during tutorial sheets, such as the tooltip code, the GlobalState component (though that was re-written from scratch) and dropdown menu logic.

For colour generation, Colorgorical was used. http://vrl.cs.brown.edu/color

For colour blindness simulation/checking, https://www.color-blindness.com/coblis-color-blindness-simulator/ was used.

No artificial intelligence was used for writing, nor any other help except the lecture slides.

For idea generation and inspiration, https://www.data-to-viz.com/ , https://datavizproject.com/ and https://datavizcatalogue.com/ were used, as well as, of course, the lectures. VisualizingSEP also provided a starting point for thinking of this dataset and how to visualise it. However, of: the treemap +all related components, all the interaction apart from the dropdown menu for the network (so the tooltip, drag, zoom, click to view single node, force simulation equation etc.), the large network itself, the colours, the layout, none were taken from/inspired by the website. None of the code was used/looked at for inspiration. The only real influence was thinking about the node size as the number of links, and the dropdown menu for selecting subcategories.