Tree Diagrams and Sunburst: Multiple visualizations for a better perspective

Group 10D: Zakaria Ameziane, Jakub Jerzy Chrupek, Jan Jacob Cozijnsen, Robin van Hoorn, Daemon-Keith Kregting, Yeoch

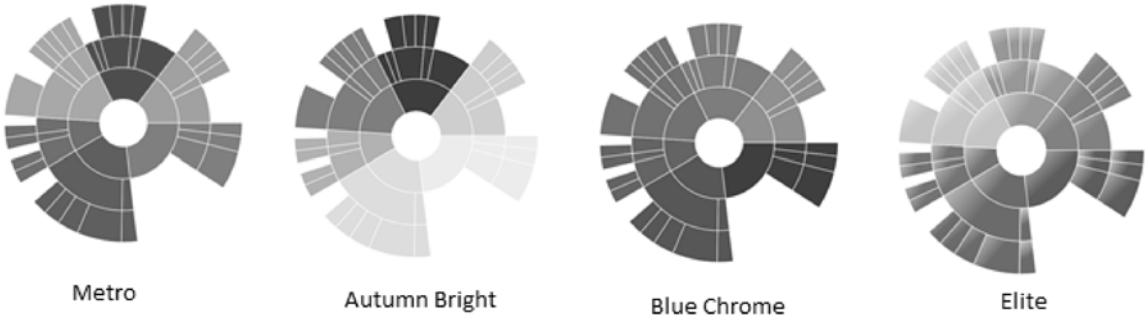


Fig. 1. Multiple sunburst diagrams, each with a different color set, to give the user a part-to-whole view with customizable designs

Abstract—Within this research paper we evaluate and describe two visualization techniques for hierarchical data on which we are currently working on. The Sunburst and Tree diagrams complement eachother as they both give an unique perspective of any hierarchical data set. By combining these techniques we can provide the user multiple views to look into the data sets, which can be helpful in both educational as analytical instances. At the moment we are transitioning from Python to Javascript and their libraries as the latter appears to give faster results when combined to our web-app. By using this application we can show our visualizations to any user worldwide in a clear and convenient way. This paper also includes discussions on our choices and a contentious review on the limits of our implementations.

Index Terms—Hierarchical data visualization, Sunburst Diagrams, Trees, Multiple perspectives, Web-based visualization

1 Introduction

During this era of Big Data, information visualizations appear to be of great significance, for they make it much more effortless to communicate data effectively. Yet we must reach a balance between validity, to inform the user, and creativity, to create visual attractiveness. There have been created many different visualization techniques, yet most of them can't provide an optimal overview for larger data sets.

Within this report we will describe our own visualization tool, which is a web-application combined with two different visualization techniques: tree diagrams, to give more details about nodes, and sunburst diagrams, which is more focused on the part-to-a-whole perspective. By providing this tool within a web-application, we can ensure that it can be used worldwide when we finally get a server.

Our main goal is that our tool will be able to process the NCBI taxonomy file that has been provided to us, which consists of more than 300.000 elements. Yet at the moment it still runs rather slow for larger sets, so within this report we will focus on what we have achieved so far with smaller sets. However, as we are still working on this project, more updates will follow in the next report.

2 Related Work

There have been created multiple different visualization techniques, yet none of them have proved to be the perfect one to visualize hierarchical data. We use two different techniques to give the user more perspectives, but there are more ways to visualize the same data we use.

The most representations use the same basic techniques, but in a different form. These techniques include stacking, nesting, fractal concepts, node-links or indentations. For example, tree-maps are a 2D representation that use nesting techniques, just like Voronoi diagrams and foammaps, but all three of show a completely different side of the dataset.

Moreover, we can distinguish 2D, 3D and hybrid techniques. Most standard 2D visualizations, like tree diagrams, contourmaps and treemaps have a 3D counterpart. The latter often gives more interaction possibilities and perspectives, but is harder to implement due to it's complex structure.

3 Data Model

Our visualizations are specially made for hierarchical data sets, which are data models that are organized into tree-like structures. Each data point within these sets are called record and they are all connected with eachother by so-called links. These records are collected in fields and each field has a certain type. In other words, the type decides in which field the record is contained.

However, we need these hierarchical data sets in a certain format for our visualizations being able to process them. This format is called the Newick tree format, which uses nested parentheses to represent trees in a computer-readable format. An example of a Newick format would be:
(A,(C,D,E)B,)F;
Which will create a tree with root F, its children nodes A and B and node B's children C, D and E. When a tree without root is written in

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

the Newick format, an arbitary node will be chosen as its root. Because of the fact Newick formats are easily read by the computer, it's greatest benefit is how easily it can be processed by software, in this case our visualization program. So far we haven't noticed any drawbacks and are rather satisfied with this format as it is the most used format and easy to store.

4 The Visualization Tool

Our web-application combined with our two visualizations, the tree diagram and the sunburst, will provide the user a multi-view visualization tool that can provide more insight and different perspectives. Within this section we will discuss the GUI, the specific techniques we used and the details about our implementation.

4.1 Graphical User Interface

We used a web-based visualization tool, which you can see within section 4.4 in figure 1 & 2, On this page you have access to both of our visualizations and their descriptions, a tool to upload your Newick files and a special feed for our reports and videos. At last some general information about our group and its members can be found on this page as well. The user can find everything on one page and when he uploads a Newick file he will see the visualizations immediately.

Our wish is to show both visualizations at the same time after the user uploads a file, but due to visualization 2 not being interactive yet, we chose to show only visualization 1 in the "Visualization 1 2" Section. In our current implementation the user will still see the option to switch visualizations in the form of a drop down menu, where he also can find visualization 2, yet the user won't be able to interact with it yet.

4.2 Visualization Techniques

We use two different visualization techniques: the tree diagram and the sunburst diagram. The reason why we choose for these two different ways to visualize hierarchical data is that they complement eachother. The tree diagram gives a lot of detail about the individual nodes and their structure, while the sunburst diagram gives more insight about the part-to-a-whole structure.

The sunburst diagram consists of a root node in the center and several rings around it. Each ring corresponds to a level in the hierarchy and all rings are sliced up into slices/nodes and they are divided based on the connection with the other slices. It is possible to use colors to highlight different kind of fields to categorize the data set.

On the other hand we have the tree diagram, a more classic way to visualize data. A tree diagram has a root node, which is a node without parents and several inner nodes and leaves. The difference between inner nodes and leaves is that the latter has no children. The tree diagram shows each node individually and can be used to give more information about the relations and descent of each node.

4.3 Implementation Details

Our implementation consist of three parts, server side, data processing and client side. Starting with the server side. During the research phase we have debate over two python webapp frameworks, flask and Django. In the end we have decided to go with flask, because of its simplicity and the minimalistic approach. It has a lower learning curve therefore it was easier for all of the member of the group to learn it and be able to work around it. Moreover Flask includes a HTML templating format, which was one of the key factors in choosing it over Django, it enables us to create a template in which data can be later pass to, depending on the outcome of the data processing. In addition Flask provides plugins, such as the uploader, that speed up the development process and allows us to write simpler code.

The second part of our implementation is the data processing. This happens in two steps, firstly the Newick tree file is converted into JSON format, secondly the data is passed through into the visualizer. At the beginning we have experimented with couple of python libraries to see, which one would suit us the best. At first we looking into biopython and networkX for the conversion of the Newick file into JSON, however these libraries didnt provide what we were looking for. Thus we have moved on to ETE3, which is a library that reads the Newick format, converts the root into an object with nodes as sub objects. This has allowed us for greater manipulation with the tree itself therefore allowing us to create a converter. The converter itself is not too complicated, it takes a Newick format as input, converts it into a tree object from which we take individual nodes and put them into a hierarchical JSON file.

The second part of the data processing is the visualization itself. At the start we discussed couple of possibilities for the visualization libraries. We talked about using Bokeh, Ete3 or D3.js. In the end we have decided to use D3.js, this was due to the fact that Bokeh didnt provide us with tools we were looking for. It mostly had tools for visualization of network graph, not trees, therefore we decided to drop it. Subsequently, we only considered ETE3, because we have been already using it for the file conversation. However it turned out that ETE3 doesnt provide an interactive framework, which was the key feature that distinguished D3.js from the group. D3.js, as the name suggests, it a JavaScript framework, thus two of our member has to learn it from scratch. In addition D3.js has quite a high learning curve with lot of information to be absorbed at the beginning, but much less later on when programming the visualizations.

One of our concerns is that all the visualization processing takes place on the client side, however this is an issue we are trying to solve. We have experimented with server-side D3.js rendering and passing the render objects to the templating. While this method is the easiest it would force us to change the server side engine. We would have to migrate from Flask into node.js, which is a server-side JavaScript execution software, this decision is mostly influenced by the visualization framework, D3.js, this is because D3.js when visualizing each node of the Newick tree it generated a DOM object in out browser, DOM object are particularly RAM demanding, therefore when we tried to visualize the big NCBI tree (as the tree not sunburst; without edges) it was already using 3.5 GB of RAM. Because of this issue we considered the Node.js as mentioned before, however there is another method of converting the DOM objects into Canvas objects, which are in comparison very lightweight and do not require such about of RAM to keep them stored, however this method is one we still have to look into.

Lastly, our implementation consist of the client-side, which is mostly the user UI and the place holders for the visualizations themselves. As we have mentioned before Flask provides a templating engine, Jinja2, which allowed us to create a responsive website layout. Moreover we used Bootstrap for the CSS templating, to make the website look consistent. As the part of the website we added the upload button, which is mostly handled by the Flask server, the uploader checks the file format to make sure it is supported, as well as, the Newick format. For the visualizations, right now a selector has been implemented to choose, which visualization is shown. However in the future we are planning to make the visualizations interact with each other, therefore allowing us a greater insight into details about the trees.

4.4 Screencaptures .

Figure 1

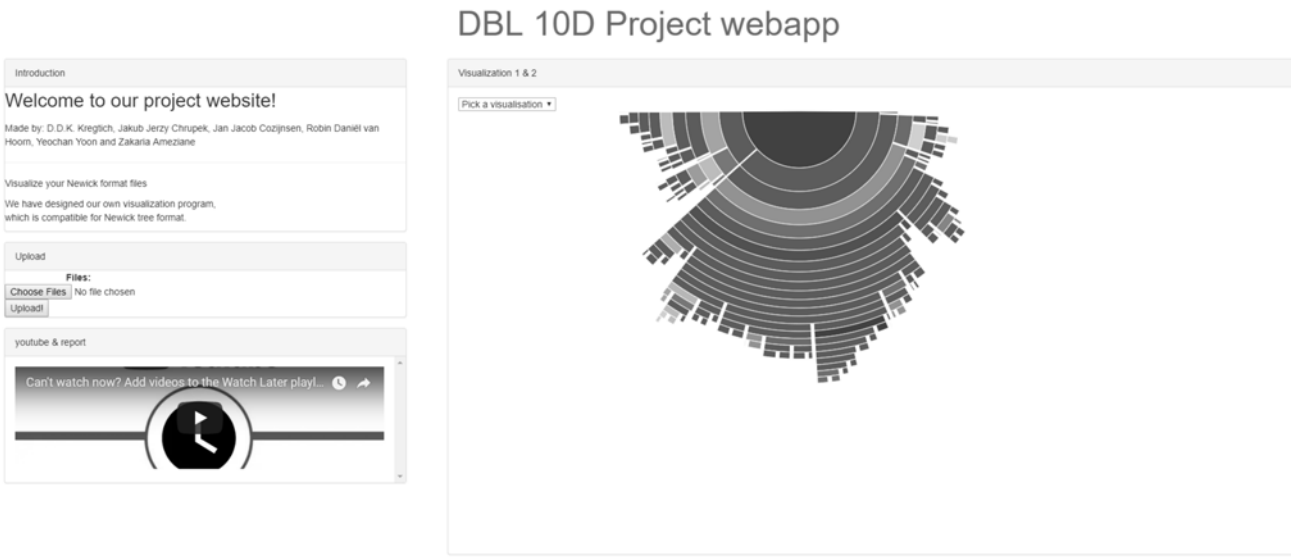


Figure 2

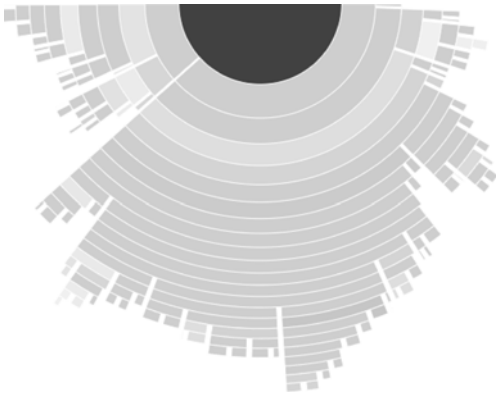
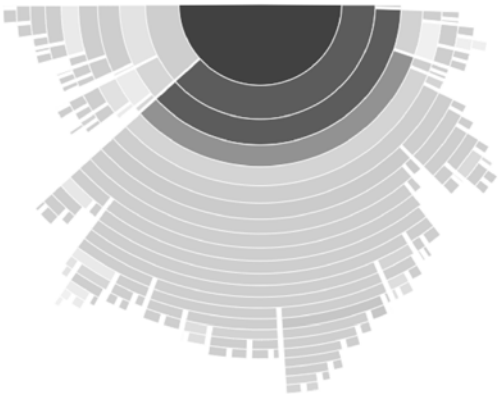


Figure 3



- 0.99985 - 0.99985 - 0.760404

Figure 4

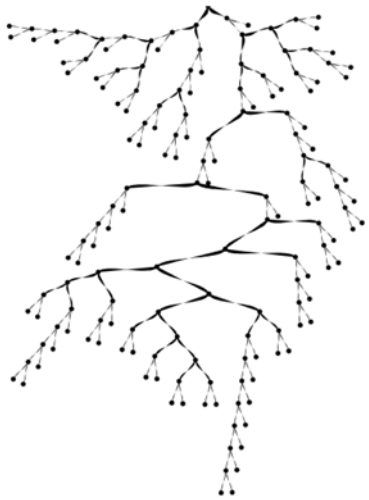


Figure 5

5 Application Example

If we show the dataset as a sunburst diagram, figure 3 & 4, we can see a that the root node is shown as a big colored centre with multiple rings around it. Notice that in this case the most outer rings are the thinnest but also are divided into small slices. We can easily see who the parents are of each node and how each node compares to the whole set. This visualization is interactive and when we select a node, this node and its ancestors will get highlighted. When a node is selected, you need to click on the centre node a few times to fully zoom-out again. Moreover, in the lower left you will see the ID's of the selected node and its ancestors

Furthermore, if we give a look at the tree diagram, figure 5, we get a completely different perspective on the same dataset. We definitely see a better structure in this visualization and it gives a more perspicuous view on the individual nodes, especially on the leaves.

6 Discussion and Limitations

Our goal is that our visualization tool can give multiple and clear perspectives of any hierarchical data set, even large ones. Our sunburst diagram is more useful for larger databases in comparison to the tree diagram, because it will give a much clearer overview of the elements thank to its slices and rings, which are easier to distinguish in large files than trees. However the bad navigation system in the sunburst diagram appears to be a limitation, because when you select a node in the sunburst diagram, you need to click on the on the centre node a few times to fully zoom-out. This can be bothersome when you are dealing with large sets.

Moreover, as mentioned earlier, our visualization processing is still rather slow. This cause the tool to take a long time to load larger sets. At the moment we are still using the client-side, which is the reason why it takes so much time. Therefore, we hope this problem will be fixed when we are switched to server-side rendering with 3D.js.

At last, our visualizations do not appear fully when the webpage is in windowed mode. This is not a major issue, yet it can cause limitations when the user wants to use our tool in windowed mode instead of fullscreen. We think this just to be a layout issue within the code to link our web-application and visualization, so we expect to get rid of this minor limit rather soon.

7 Conclusion and Future Work

We can conclude that our first visualization is working efficiently with small databases. Together with the web-application we can confidently state that we made a reasonable first version of our visualization tool. However, we are still working on visualization 2, as we want to give it more interactions like 'selection', 'expanding' and 'zooming in'. Moreover, for our future sprints we would like to optimize both visualizations so they can run larger data bases. We are also going to experiment with server-side D3.js rendering to speed up the visualization processing and we want to make the sunburst diagram easier to navigate, so it can be user-friendly for larger databases.

Acknowledgments

We would like to thank our tutor for her support during this DBL and for always showing up on time. Moreover, we would like to thank all group members for their hard work during the process of making these visualizations.

8 References

http://evolution.genetics.washington.edu/phylip/newicktree.html

http://marvin.cs.uidaho.edu/Teaching/CS515/newickFormat.html

https://docs.microsoft.com/en-us/sql/relational-databases/hierarchical-data-sql-server?view=sql-server-2017

https://www.cc.gatech.edu/ stasko/papers/infovis00.pdf

https://lifeonearth.seas.harvard.edu/wp-content/uploads/DeepTree$_SDR_Harvard.pdf$

http://www.treevis.net

http://syncfusion.com

F. Beck ., M. Burch, T. Munz, L. Di Silvestro and D. Weiskopf: Generalized Pythagoras Trees for Visualizing Hierarchies, 2014

J. C. Roberts. State of the art: Coordinated multiple views in exploratoryvisualization. InProceedings of Fifth International Conference on CMV,pp. 6171, 2007.

J. Rosindell and L. Harmon. OneZoom: A fractal explorer for the tree oflife.PLOS Biology, 10(10), 2012.
H. Schulz. Treevis.net: A tree visualization reference.IEEE Computer-Graphics and Applications, 31(6):1115, 2011.

H. Schulz, S. Hadlak, and H. Schumann. The design space of implicithierarchy visualization: A survey.IEEE Transactions on Visualizationand Computer Graphics, 17(4):393411, 2011.

M. Sondag, B. Speckmann: Stable Treemaps via Local Moves, pg 729-738 (2017)

References