Status Update: *Judicial Case Law History Timeline* (November 2015)

Ken Mansfield, *CPSC 547*

Abstract—The aim of the Judicial Case Law History Timeline is to provide a solution to visualizing the citations (cross-references) between case decisions (written judgements) related to the treatment of particular legal concepts over time for legal research and collaboration. The dataset is dynamically queried from data parsed from the BC Laws Database.

1 Introduction

TWO weeks has elapsed since the project proposal, this document will serve as an interim update.

2 Work Completed

2.1 Chord Diagram

Judicial cases can cite many cases and be cited by many more. These citations can be viewed as a network similar to a social network where children nodes can be linked to many other nodes. For this visualization the chord diagram idiom has been chosen for its ability to encode a considerable amount of information with limited occlusion of important data.

Data-Driven Documents (d3) [1] has been chosen as the technology of choice by the collaboration parter, Knomos Inc. [2]. The chord diagram example from the d3 website [3] provides example source code that is easily extensible. The first proof of concept iteration is to encode case citations in each of the chords and link cases together. This is achieved by constructing a matrix that defines these conne-cions:

```
 [[5,  5, 5, 5],
[ 10, 5, 0, 0],
[ 10, 0, 5, 0],
[ 10,   0,  0, 5],]
```

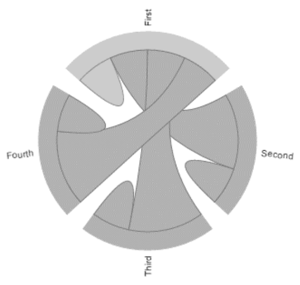


Fig. 1 First work. Initial test of the chord diagram.

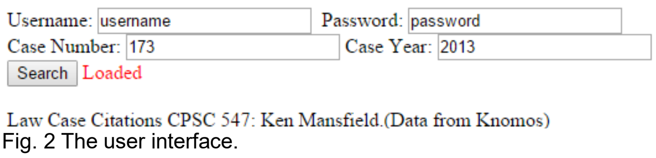
2.2 Constructing the Matrix

The data is provided by querying the Knomos web server's API which responds with structured JSON data. Each API response returns the data for a single judicial case. This would be enough to construct a graph with one parent and it's nodes but for a more complex visualization we need to iterate over more cases. To do this we need to make iterative asynchronous queries over the child nodes to collect 2nd degree nodes. Once this data is collected into javascript objects pushed into a matrix-array we can construct our chord diagram with live-data.

2.3 Visualizing Live Data

Adding a text field and a submit button enables the user to select the primary case they wish to build the chord-diagram for and initiates the load. Since several API requests are needed to build the matrix, the visualization takes approximately 20 seconds to load. A simple loading counter is provided to list how many requests have been made.

Once the matrix has been built the chord diagram visualization could be made with live-data. This iteration took its style (and some source code) from the dependency wheel visualization [4]. This iteration of the visualization made it evident that the nature of the data has a big effect on how the visualization would look and many changes would need to be made. The dependency wheel example had a carefully thought out gradient colour which did not come out right in the visualization. On top of that, the data we had loaded had many links coming out of the primary case and it's children, allowing them to have wide chords, but the 2nd degree nodes have no references loaded meaning they had 0 width causing crowding and text occlusion (a figure could not be provided because this version no longer works do to server API changes). This was somewhat alleviated by increasing the 2nd degree nodes widths. A new colour scheme was included for the current iteration (Fig. 3).



Username: username Password: password
Case Number: 173 Case Year: 2013
Search Loaded

Law Case Citations CPSC 547: Ken Mansfield.(Data from Knomos)

Fig. 2 The user interface.

• *Ken Mansfield, UBC M.Eng student. E-mail: kmansfield@ ece.ubc.ca.*

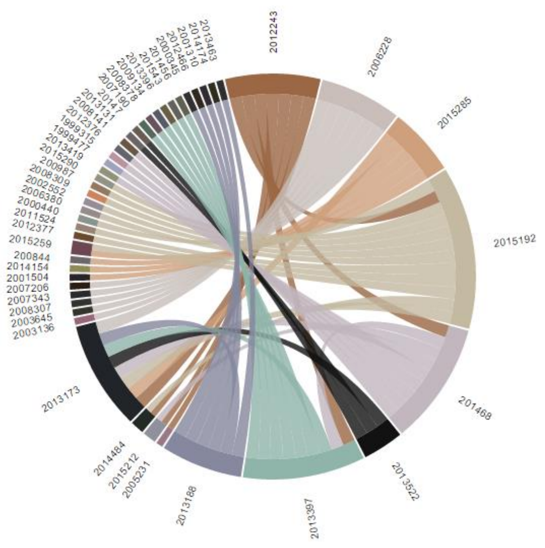


Fig. 3 Current iteration of the d3 chord diagram.

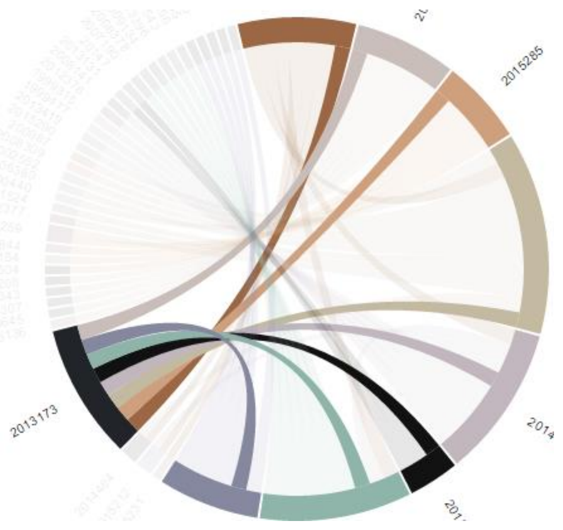


Fig. 4 Current visualization with the mouse hovering over the primary chord allowing the user to view all its links.

3 Obstacles and Changes

The first obstacle that was encountered was the inability to query the server API directly via javascript in the browser. Apparently this is a common limitation for security reasons. The server team eventually added a "CORS header" to the API to allow remote API queries. The other option would have been to make a local proxy server but this would have the limitation that it would not be visible to others.

The next obstacles came in the form of API changes. First the JSON structure changed requiring a client side change to parse the data differently. Next the API became password protected, requiring an account to be created and authentication code to be added to the client. These changes exemplify some of the issues of dealing with an external, live server data.

After initial work with the chord diagram several problems have arisen. Links between chords have a specified width that are different on both ends. For case references it is unclear what this width should encode. Links are not directional so it unclear whether a link from chord A represents a *cites* or a *cited-by* link. It is also unclear what is encoded by color in this iteration. Opposing chords have different colours, but links between the chords have the color of one chord or the other. This coloring might be useful for encoding directionality.

4 Related Work

There are many popular ways to visualize network data.

4.1 Events in the Games of Thrones

The first example suggested by the Knomos team was the "Events in the Game of Thrones" visualization by Jerome Cukier [5]. Each link between one bubble node to another indicates a kill. The visualization is a very creative way to depict the interactions between Game of Thrones Characters, but it falls short as being useable as production level software. For example, there is no indication of who killed who (ie. which direction) between nodes connected by lines. The visualization is also crowded with no labels on the bubbles, the user must hover over a node to determine the character's name and then hover over the node that it links to, to determine who that character killed or was killed by.

4.2 Initial Project Proposal

The proposed modification would use a similar layout as above:

- Each node represents a case decision, clustered by jurisdiction (eg. Supreme Court of Canada).
- The size of the node represents the frequency of the case's citations throughout other cases decisions.
- The color of each node signals the relative treatment of that case decision in other cases (eg. Green = favourable, yellow = distinguished, red = overturned).
- The lines between nodes represent interactions (the citations) between case decisions.
- The time control scrollbar can be shifted to adjust the visualiztion, and stopped at a specific point in time.
- Content scope can be filtered by including or restricting the years or jurisdictions selected beneath the scrollbar.

In the original "Events in the Game of Thrones" visualization the nodes were grouped by families and kills mostly happened between one family to another. For judicial cases, citations are most likely to come from the same group (ie. within BC Laws). It is highly likely that all cases

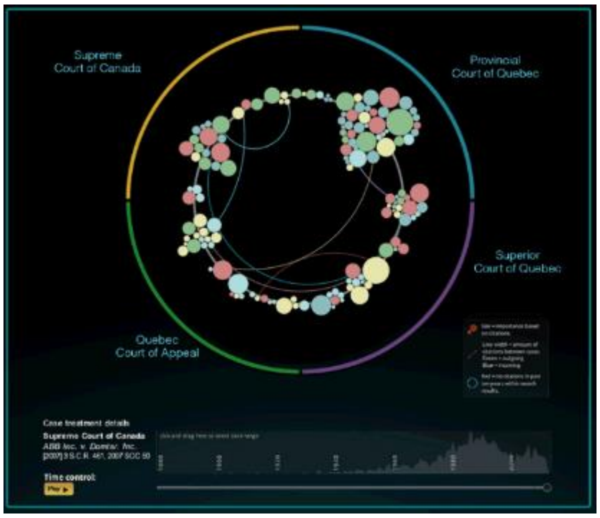


Fig. 5 Proposed visualization based on the Events in the Game of Thrones visualization.

could all pertain within one jurisdiction so grouping the cases this way with lines (citations) looping back to the same group would be difficult to distinguish. Additionally, for this project we are limiting our data to one jurisdiction so the depicted clustering will not work.

A character is usually killed only once, meaning there is only one link coming into it. For judicial citations, there is likely to be a many to many relationship. With the large clusters in the Game of Thrones visualization it is hard to see where the lines are coming in and out.

4.3 Node-Link Layout

The most common idiom for visualizing network data is the Node-Link Diagram. The Judical case data would be very easy to adapt to this approach as each case could be thought of as a node and the citations can be thought of as links. The d3 Force-Directed Graph visualization [6] is a popular example employed by many. One glaring issue with this approach is that text label occlusion is likely due to the location and density of nodes, as well as link occlusion in large clusters. The force directed graph is already used by Knomos as well so it is useful to study different approaches.

4.4 Other Social Networks

Citation networks can be thought of as a type of social network. Adapting this to judicial cases, we could think of the primary case as the “ego”, and it’s children citations as “alters”. A recent paper, egoSlider [7], specifically focuses on viewing the changes of an ego’s network over time. The approach that they have taken would not lend itself well to judicial cases, however, because a case does not persist over time the same was a human does. Judicial cases have a single time-stamp so it would not make sense to analyze things like tie-strength.

5 Future Work

The purpose of the visualization is to dynamically view the case citation changes over time, so the next logical step would be to add a timeline and animate it.

References

[1] Mike Bostock, “Data-Driven Documents,” d3js.org. 2015.

[2] Adam La France, Jesse Abney, “Knomos,” www.knomos.ca. 2015.

[3] Mike Bostock, “Chord Diagram,” http://bl.ocks.org/mbostock/4062006. 2012.

[4] François Zaninotto, “Dependency Wheel,” http://www.redotheweb.com/DependencyWheel/. 2013.

[5] Jerome Cukier, “Events in the Game of Thrones,” http://www.jeromecukier.net/projects/agot/events.html. 2015.

[6] Mike Bostock, “Force-Directed Graph,” http://bl.ocks.org/mbostock/4062045. 2012.

[7] Yanhong Wu, Naveen Pitipornvivat, Jian Zhao, Sixiao Yang, Guowei Huang, Huamin Qu, " egoSlider: Visual Analysis of Egocentric Network Evolution," *IEEE Trans. Visualization and Computer Graphics*, vol. 22, no. 1, Jan 2016, (IEEE Transactions)