Bewilder: Handling Web Resource Complexity in Online Learning/Research

Eric Easthope
Department of Computer Science
University of British Columbia
Vancouver, Canada
eric.easthope@me.com

***Abstract*—TBD**

***Index Terms*—TBD**

I. Introduction

Organizing web links like Uniform Resource Identifiers (URIs) and Document Object Identifiers (DOIs), which is often done within browser-based folders, remains a visually cluttered and awkward user experience. The alternative, to save and archive tabs offline (also usually within folders), is no more effective. Given many web items, the amount of manual categorization a user must do quickly becomes overwhelming.

New browser features like tab grouping get us closer to being organized, but are in effect just views of additional folders. Moreover, many in-browser bookmark and resource managers, or other paid software tools, put heavy emphasis on users to manually annotate and sort their own content without much automation.

Task complexity increases further if we try to organize and connect common web links to other URI/DOI-based resources such as arXiv papers or government information. A user may also wish to add notes to URI/DOI metadata to recall later why they have saved URIs/DOIs in the order they are in, and how they are connected. Two key visual challenges arise:

- How do we make it visually and cognitively easier to review collected web links with additional context?
- How do we visualize these collections at scale?

Enter Bewilder. Bewilder gives us a way to overview, annotate, and arrange collected web items in a way that retains *why* and when we are saving them, and how they are connected to other resources. Bewilder aims to facilitate large-scale browser-heavy exploration and research by giving users, primarily sensemakers, “knowledge workers” (e.g. engineers, scientists, etc.), and students, a way to produce annotations and traceback for online content.

A. Personal Expertise

How to construct effective representations of connected URI/DOI data, and how these representations could aid learning and research, has been an off-and-on personal interest over the past few years, but without expert knowledge. Previous efforts focused more on the programmatic acquisition and handling of online content, but not so much the visualization of user-collected representations of this content. In preparation for this work, we have engaged with learners and researchers directly to learn about some the resource-related problems they face in their work, and what tools they currently use to archive and retrieve online content.

We have also prototyped various forms of ontological and knowledge-driven software tools in the past using D3.js with the broader intent to support our own work and data-driven research. Some of these previous developments also centred on the problem of digital notetaking and how to share learning resources without just copying them entirely.

II. Related Work

Tools for URI/DOI management, and more broadly tools for online exploration and web-based archiving, have been explored in the past. Some of the notable online efforts are [1] (for how systematic it is), [2] (for how long it has been maintained), and [3] (for its cross-platform software compatibility). In academia, [4] is popular too, particularly for managing large-scale research-driven bibliographies. There is also [5], [6], as well as [7], which all seem to reflect a desire for interlinked and metadata-rich web content.

In the academic literature, there is also [8], [9], [10], and [11]. Closest to our own interests for this paper is the “discussion topology” view in *Kialo* (that is [8]), which effectively visualizes the topology of arguments between participants on a topic (shown below in Fig. 1). Like our own design (to be described in greater detail later on), a radial vis idiom shows a hierarchy for added items, and provides hover-activated tooltip previews for these items. However, the visualization in [8] differs from ours in that its perceived hierarchy is only partially “acausal” (i.e. partially ordered in terms of when items are contributed), and the items shown must be contributed through alternative text-based subpages of the hosting website for [8]. Papers [10] and [11] also discuss and develop radial vis idioms, but do so to satisfy different task requirements for distinct problem domains.

While *Zettelkasten* (that is [1]), described as “a personal tool for thinking and writing,” provides an interesting and methodical way to hyperlink thoughts and writing, and one from which some successful software tools have emerged, we think the method and accompanying software is incomplete. These software tools, namely [12] in particular, but also others such as [13] and [14], do not seem to emphasize or enable the visualization of connected graph-like content and the overall topology of such graphs. Moreover, modifying this graph topology through individual items is done indirectly by navigating through multiple views, which do not often retain context and therefore add cognitive load. This indirectness distracts users from the underlying task of forming recognizable and actionable connections between objects and associated categories. *Note to readers: I wasn't sure if a more "critical" paragraph like this is appropriate in a Related Work section. What do you think?*)

Other tools like [2], [3], and [4], while serving their primary purposes, namely education, archiving, and research automation respectively, seem to lack much user interactivity and visualization. Moreover, the underlying topologies of [5] and [6], which could have much utility if they were applied, are not readily accessible to the public. We speculate that some of this inaccessibility is due to a lack of viable free software tools. *Note to readers: Should this last sentence perhaps be a footnote?*)

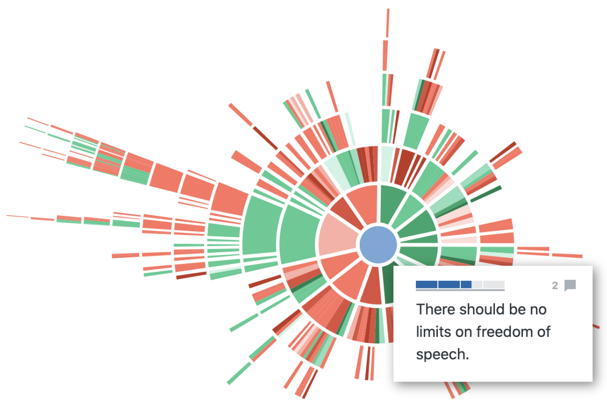


Fig. 1. *Kialo's* "discussion topology" view.

III. Data & Task Abstraction

To give users guided control over their own collections of web links and research articles, especially to modify and explore the details of these collections, Bewilder must first unify a number of often disparate forms of hyperlinked online web content. This web content is frequently saved as `.webloc`, `.url`, and HTML files. Many modern web browsers also provide a bookmark export feature to produce HTML-based archives of web content, often storing both the web URIs and titles of individual webpages.

Data from user discovery, as well as personal bookmark collections exported from Chrome, Safari, and [15], have informed our task definitions as well as early prototypes. These bookmark collections are structured as nested hierarchies of web links, where HTML headers, sub-headers, and so on, represent user-customized bookmark folders.

All of these file formats (`.webloc`, `.url`, and HTML) can be unified as nested JavaScript Object Notation (JSON) data (e.g. Fig. 2). We can also use programmatic access of [5] to retrieve titles, types (e.g. `video.movie`), and images associated to each URI. Many DOIs are also hyperlinked to specific URIs, and can therefore be represented within the JSON file format.

Broadly speaking, Bewilder should produce at-scale overviews for as many as hundreds or more annotated and interlinked web items. Bewilder should also enable users to annotate and edit web link metadata, such as the titles, types and images mentioned, as well as custom user-selected categories, to create dependencies and connect web items to each other.

Our task abstractions are defined through "user stories," which are derived from discussions and interactions that we have had with students, researchers, and other learners and information foragers outside the academic community. Each user story briefly describes a practical use-case for Bewilder. First we itemize these user stories, and then introduce tasks that arise from each user's needs:

1) "*As a student I want to be able to quickly aggregate and group URIs by their similarities so that I can solve resource-heavy assignments faster.*"
2) "*As a researcher I want to be able to re-arrange and annotate URIs/DOIs so that I can take notes on a per-link basis during literature search.*"
3) "*As a journalist I want to be able to make directed and/or causal links between URIs so that I can trace through the events leading up to a breaking story.*"
4) "*As a self-directed learner I want to be able to create and annotate URIs/DOIs with traceback so that I can recall later how I found specific information.*"

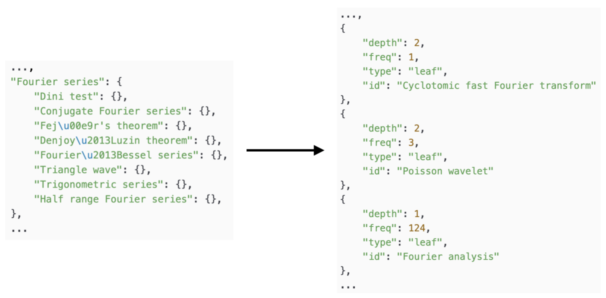
All four users are trying to **derive** *tree* or *graph*-like data from individual web *items*. In particular, we treat URIs and DOIs as *nodes*, and their connections, defined through custom categories or through similarities in their metadata (such as mutually shared types, which are also categorical), as *links*. These nodes and links are representable as JSON data. Also, by keeping track of when nodes are added, we can derive a directed graph representation.

User (1) takes this directed node-link data, and combines it with categorical node *attributes* (either annotated by the user or retrieved from [5]), to derive a new graph *topology* by adding links. These links indicate new dependencies of attributes within the graph. The user can then **reduce** this graph by *aggregation* based on these dependencies.

User (2), like user (1), uses the annotation of categorical attributes on the graph's nodes to produce new node-to-node dependencies. The user can then **encode** new graph arrangements, *ordered* or otherwise, to **identify** *features* and/or *outliers*.

Users (3) and (4) perform mutually similar tasks. However, unlike users (1) and (2), these users produce and **arrange** a directed graph topology by adding many nodes, then adding categorical attributes to produce dependencies (which produces links), and finally re-ordering these nodes based on the order in which they are added.

These four user stories, described in this way, inform the functional design of Bewilder.



```
...,
"Fourier series": {
    "Dini test": {},
    "Conjugate Fourier series": {},
    "Fej\u00e9r's theorem": {},
    "Denjoy\u2013Luzin theorem": {},
    "Fourier\u2013Bessel series": {},
    "Triangle wave": {},
    "Trigonometric series": {},
    "Half range Fourier series": {},
},
...
```

```
...,
{
    "depth": 2,
    "freq": 1,
    "type": "leaf",
    "id": "Cyclotomic fast Fourier transform"
},
{
    "depth": 2,
    "freq": 3,
    "type": "leaf",
    "id": "Poisson wavelet"
},
{
    "depth": 1,
    "freq": 124,
    "type": "leaf",
    "id": "Fourier analysis"
},
...
```

Fig. 2. Sample of nested JSON transformed to derived node data (*note to readers: will replace this figure with more detailed and URI/DOI-related JSON for the final report*).

IV. Solution

Bewilder provides users with a radial force-directed graph idiom that uses a partial ordering scheme (shown in Fig. 3 as annuli) to create a sense of hierarchy relative to the first node that is added during a session. Node radii are also varied to emphasize "central" nodes within the hierarchy (technically speaking, nodes that are not *leaves*). This hierarchy is not strictly enforced, and can be modified by simply dragging and dropping nodes into another annulus (which updates the underlying "global" data structure accordingly, in particular the `level` attribute). Also, nodes without links are de-emphasized with lower opacity.

Users can add web links by dragging in and dropping `.webloc`, `.url`, and HTML files from their desktop, which are parsed automatically to produce a JSON-like data structure. HTML files, in particular, are parsed as JSON based on the nesting of web links within bookmark folders. From this nested JSON data, we can derive node and link data (e.g. Fig. 2).

If a user clicks and holds onto a node for an extended period of time (without dragging, which keeps the user in the radial graph view), they enter the annotation view (Fig. 4, *note to readers: will be added in the final report as it is currently under revision*), which is shown as an semi-opaque overlay on top of the radial graph view. The annotation view displays and allows users to modify a specific node's data attributes, such as the URI/DOI, its title (which is used by default as a tooltip-based label), its categories (which can determine link dependencies), or to add custom notes.

The choice of making the annotation view semi-opaque is to allow users to retain a sense of visual context, while still focusing their attention on modifying node-specific attributes. Again, modifications to a node's attributes update the underlying global data structure. Also, node attributes that are dynamically and automatically retrieved from [5] are shown in this view. Clicking outside this view, or pressing the "×" in the top-right corner, hides this view and returns the user to the radial graph view.

Users can hover over any node in the radial graph view to see a text preview for which nodes are connected to it. These connections are shown with respect to incoming and outgoing link directions (i.e. is the hovered-over node connected *to* or *from* another node?). Users can also hover over an annulus as well to see a text preview for which nodes share the same `level` attribute.

A "smart defaults" feature automatically creates links and groups URIs/DOIs based on attributes retrieved from [5]. To satisfy user stories (3) and (4), a "chronology" feature forces node ordering based on timestamps of when their associated web links are added to Bewilder.

Lastly, users can download their modified data as a JSON file.

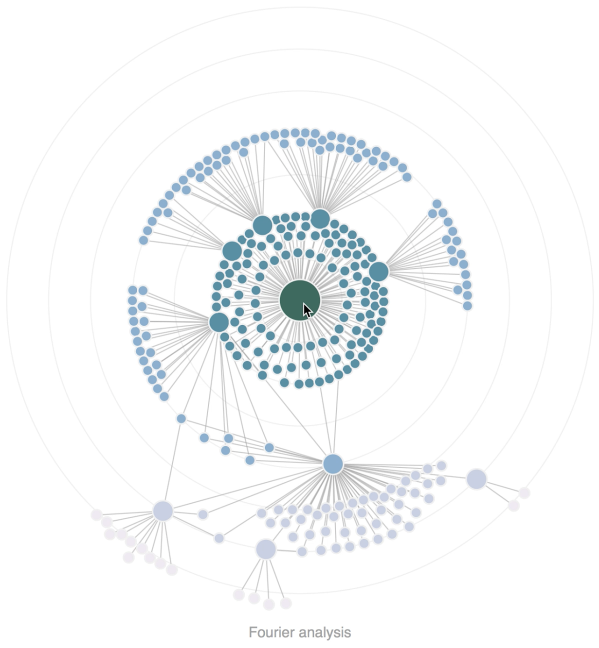


Fig. 3. Radial force-directed graph idiom (*note to readers: de-emphasized node opacity, text previews, and link directionality are not shown here*).

Fig. 4. Annotation view (*note to readers: will be added in the final report as it is currently under revision*).

A. Implementation

Bewilder is implemented in JavaScript using Webpack, and will be hosted online at *www.bewilder.me*. Hosting is integrated with Vercel to enable continuous deployment from GitHub by automatically updating the website when new production versions are released. Source files for Bewilder are also version-controlled and hosted in a private repository on GitHub.

All visualizations are also written in JavaScript using D3.js. D3.js is used for developing the vis idioms, for data manipulation, and for modifying the appearance of some user interface (UI) elements. Redux is also used (client-side) to manage UI-related state.

We also use TypeScript to add simple checks for matching data types, and to catch and prevent unexpected data handling errors during web link importing and in-app management of imported URIs/DOIs.

Some code-driven features, like Bewilder's ability to get a dragged-in web item's URI/DOI data, borrows from and adapts previously-written code for this purpose. Also, node attribute data from [5], which can be programmatically and dynamically retrieved, is used to supplement initial URI/DOI attributes and provide immediately useful tooltip-related metadata.

B. Milestones

Here we itemize both finished and ongoing milestones for Bewilder (*note to readers: these milestones have been revised since our proposal to more accurately reflect the current development pathway*):

1) **Add support for importing URIs/DOIs**: We can get nested URI/DOI data from any dragged-in `.webloc`, `.url`, or HTML file, as a JSON object within JavaScript (we have also developed a file "dropzone" to make this step easier). For each URI/DOI, we add a JavaScript object to the underlying "global" data state, and render a prototype vis item for each JavaScript object in within that state (began October 19th, revised October 30th, and finished November 10th: approximately 20-25 hours).
2) **Create the radial vis idiom**: We can render a radial force-directed graph layout with a partial hierarchy, and set node radii based on whether they are *leaves* or not (described in more detail above). We still need to add some of the drag-and-drop interactivity (especially to enter the annotation view), fine-tune some force-directed graph parameters (to prevent unwanted and/or distraction motion), and add tooltip support. Some stylistic changes are likely to emerge as well, but are not a priority (began November 1st, concurrently with the third milestone, and is ongoing: approximately 30-40 hours so far).
3) **Add features: annotate/aggregate/arrange/etc.**: We can enter a prototype annotation view container when a node is clicked, but still need to update the underlying global data state when the users exits the view (*note to readers: this view is currently undergoing revision*). Also, for annotation, we need to allow users to add custom attribute fields to the JavaScript object corresponding to each vis item. While we are able to show a node's `title` attribute to users when they hover over the node, we also want users to see the neighbouring node text previews described above. Lastly, if there is time, we will develop the "smart defaults" and "chronology" features (began November 1st, concurrently with the second milestone, and is ongoing: approximately 10 hours so far).

C. Results

TBD

V. Discussion & Future Work

TBD

VI. Conclusions

TBD

Acknowledgments

TBD

References

[1] S. Fast, "Introduction to the Zettelkasten Method," zettelkasten.de, 27-Oct-2020. [Online.] Available: https://zettelkasten.de/introduction/ [Accessed: 16-Nov-2020.]

[2] R. Nave, "HyperPhysics Concepts," *HyperPhysics*. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/index.html [Accessed: 17-Nov-2020.]

[3] *Pocket*. (2007). Mozilla Corporation. Available: https://getpocket.com

[4] *Zotero*. (2006). Roy Rosenzweig Center for History and New Media. Available: https://www.zotero.org

[5] *Open Graph Protocol*. (2010). Facebook. Available: https://ogp.me

[6] A. Singhal, "Introducing the Knowledge Graph: things, not strings," *The Keyword*, 2012. [Online.] Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/ [Accessed: 17-Nov-2020.]

[7] G. King, "The Dataverse Project," *Dataverse*. Available: https://dataverse.org [Accessed: 18-Nov-2020.]

[8] S. Chaudoin, J. N. Shapiro, D. Tingley, "Revolutionizing Teaching and Research with a Structured Debate Platform," working paper, Harvard University, Cambridge, MA, USA, 2017. [Online.] Available: https://scholar.harvard.edu/files/dtingley/files/structureddebate.pdf [Accessed: 17-Nov-2020.]

[9] S. Fabbri, E. Hernandes, A. di thommazo, A. Belgamo, A. Zamboni, C. Silva, "Managing Literature reviews information through visualization," in *ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems*, 2012. [Online.] Available: https://pdfs.semanticscholar.org/e5fb/4d8f9207dbec71a1b60115230e70add1b2ab.pdf [Accessed: 17-Nov-2020.]

[10] C. Collins, S. Carpendale, G. Penn, "DocuBurst: Visualizing Document Content using Language Structure," *Eurographics/IEEE-VGTC Symposium on Visualization 2009*, 2009. [Online.] Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.7051rep=rep1type=pdf [Accessed: 18-Nov-2020.]

[11] J. Stasko, R. Catrambone, M. Guzdial, K. McDonald, "An evaluation of space-filling information visualizations for depicting hierarchical structures," *International Journal of Human-Computer Studies*, 2000. [Online.] Available: https://www.cc.gatech.edu/ john.stasko/papers/ijhcs00.pdf [Accessed: 18-Nov-2020.]

[12] *The Archive*. (2017). Available: https://zettelkasten.de/the-archive/

[13] *Roam*. (2019). Roam Research. Available: https://roamresearch.com

[14] *TiddlyWiki*. (2004). UnaMesa Association. Available: https://tiddlywiki.com/

[15] *Raindrop.io*. (2013). Available: https://raindrop.io/