Bewilder

Handling Web Resource Complexity in Online Learning/Research

Eric Easthope - eric.easthope@me.com

Abstract

TBD

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:

1. **How do we make it visually and cognitively easier to review collected web links with additional context?**

2. **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.

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 interest of mine over the past few years, but without expert knowledge. Instead, I 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.

I have also prototyped various forms of ontological or knowledge-driven software tools with the broader intent to support my own work. During my undergraduate degree this centred on the problem of digital notetaking and how to share learning resources without just copying them entirely.

Related Work

Tools for URI/DOI management, and more broadly tools for online exploration and archiving, have been explored in the past. Some of the notable efforts are Zettelkasten (for its rigour), HyperPhysics (for its longevity), and Pocket (for its cross-platform compatibility). In academia, Zotero is popular too, particularly for managing bibliographies.

The OpenGraph protocol, Google's Knowledge Graph, as well as local efforts like UBC's Dataverse, also seem to reflect a desire for interlinked and metadata-rich web content.

Data & Task Abstraction

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 to connect web items to each other. This means giving users guided control over their data, especially to modify and explore the details of their collections. Automation and "smart defaults" are used, where possible, to arrange and re-arrange items based on their metadata. These requirements give rise to three distinct task abstractions:

- **Query/Summarize : Overview** (e.g. "*Show me all of the links I saved yesterday.*")
- **Produce/Annotate : Annotate** (e.g. "*This resource helped me because …*")
- **Produce/Derive : Arrange** (e.g. "*Order my links chronologically. Group by tags.*")

> **Note**: Data from user discovery, as well as personal bookmark collections exported from Chrome, Safari, and Raindrop.io, will inform early prototypes. This data comes in the form of `.webloc`, `.url`, and `.html` files.

Solution

Bewilder provides users two vis idioms for URI/DOI management: a global matrix overview, and a local graph view for previewing near-neighbour graph topology between connected URIs/DOIs (see **Figure 2** and **Figure 1** below, respectively, for a sketch).

A third view, the *annotation workflow*, which overlays the other two views, allows users to add notes and to edit and add URI/DOI metadata not provided automatically through OpenGraph.

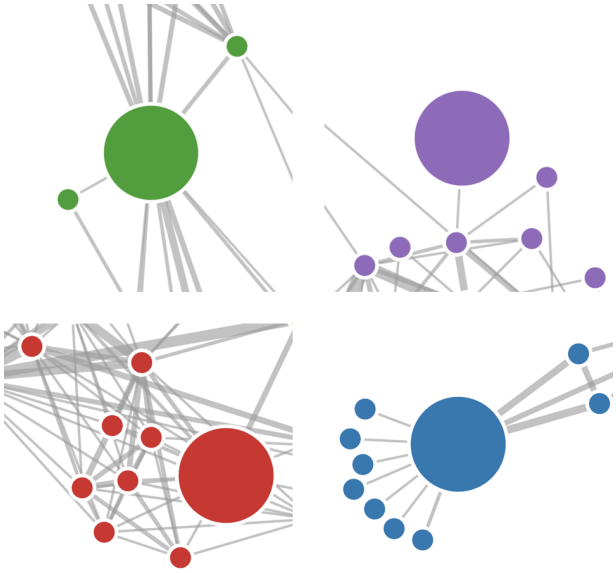

Figure 1: Examples of local graph topology. Node of interest is enlarged. *Adapted from "Force-Directed Graph" by Mike Bostock.*

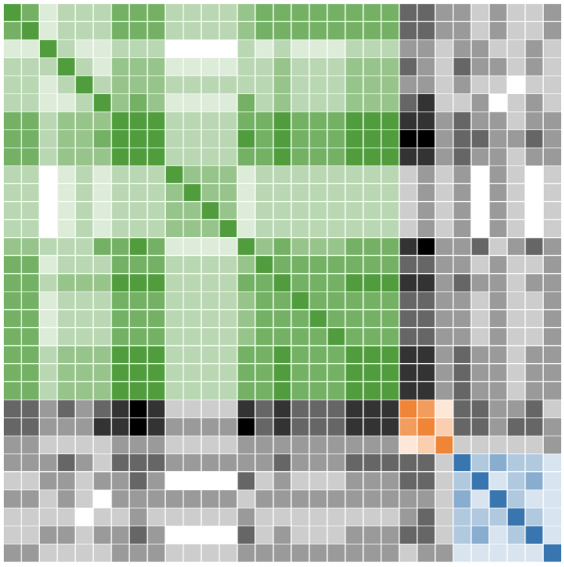

Figure 2: Example of a matrix overview. *Adapted from "Les Misérables Co-occurrence" by Mike Bostock.*

Implementation

Bewilder is implemented in JavaScript and hosted online at www.bewilder.me using the React Native framework. Hosting is integrated with Vercel to enable continuous deployment from GitHub by automatically updating the website when changes are made. Source files are also version-controlled and hosted in a private repository on GitHub.

> **Note**: The choice of React Native over other JavaScript frameworks is to preserve cross-platform support for later developments after this project.

All visualizations are written in JavaScript using D3.js and React.js. D3.js is used for developing the vis idioms and for data manipulation, and React.js is used for UI layout and to enable the development of component-based views. TypeScript adds simple checks for matching data types, primarily to catch and prevent unexpected data handling errors.

Some code-driven features, like the ability to get a web items's URI/DOI data by dragging it into Bewilder's view, borrows from code I previously wrote.

> **Note**: OpenGraph data, which can be programmatically retrieved, will populate initial URI/DOI metadata for vis tooltip (or equivalent) previews.

Results

Bewilder is built upon "user stories," which inform some practical use-cases.

First, for all users: a user opens tabs and/or windows as usual during online search in their preferred web browser. Then, the user exports active webpages to a bookmark file (commonly an `.html` file), and drags this bookmark file into Bewilder. Alternatively, individual links may be dragged directly into Bewilder from the browser address bar.

> *"As a student I want to be able to quickly aggregate and group URIs by task so that I can solve resource-heavy assignments faster."*

Scenario: The user toggles "smart defaults" to get Bewilder to try to automatically group URIs based on OpenGraph metadata (or other URI-related metadata). Alternatively, the user manually modifies the metadata of related URIs by entering the annotation workflow. Then, the user exits the annotation workflow to return to the URI/DOI overview and preview URIs (see **Figure 1** and **Figure 2** above for a sketch).

> *"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."*

Scenario: Un-toggling "smart defaults" to enable manual re-arrangement, the user clicks and drags vis elements to manually place and re-order them. To annotate and add notes, the user similarly enters, makes changes within, and then exits the annotation workflow.

> *"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."*

Scenario: The user toggles "chronology" before sequentially dragging URIs into Bewilder. Then, the user modifies URI metadata and link directionality through the annotation workflow before exiting to return to the URI/DOI overview.

> *"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 certain information."*

Scenario: (*same as the journalist*) The user toggles "chronology," drags in URIs/DOIs, and modifies URI metadata and link directionality.

Milestones

Import URIs/DOIs (*in progress*, estimated completion: *before end of October*)

1. Get URI/DOI data from a dragged-in `.webloc`, `.url`, or `.html` file.
2. For each URI/DOI, add a JavaScript object to a "global" data state.
3. Render a prototype vis item for each JavaScript object in the global data state.

Features: annotate/cluster/arrange/timestamp/… (estimated completion: *before mid-November*)

4. Enter the annotation workflow when a vis item is clicked, and update the global data state when a user exits the workflow.
5. During annotation, allow users to add custom metadata fields to the JavaScript object corresponding to each vis item.
6. Show a vis item metadata tooltip, OpenGraph data preview, or equivalent, upon vis item mouseover.

Vis idioms (estimated completion: *before end of November*)

7. Add a global matrix view for URI/DOI overview.
8. Re-arrange/sort the matrix view automatically based on annotations, web content, OpenGraph data, etc.
9. Add a local graph view for near-neighbour graph topology (likely a node-link diagram).

Discussion, Future Work, & Conclusions

TBD

Bibliography

M. Bostock, “Les Misérables Co-occurrence,” bost.ocks.org, 10-Apr-2010. [Online]. Available: https://bost.ocks.org/mike/miserables/. [Accessed: 22-Oct-2020].

M. Bostock, “Force-Directed Graph,” Observable, 15-Nov-2017. [Online]. Available: https://observablehq.com/@d3/force-directed-graph. [Accessed: 22-Oct-2020].