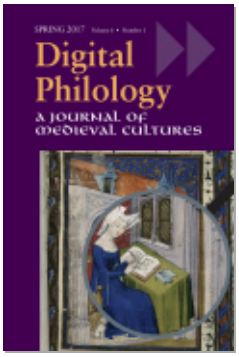
CATview: The Colored and Aligned Texts Tool by M. Pöckelmann, Paul Molitor, and Jörg Ritter (review)



Paul A. Broyles

Digital Philology: A Journal of Medieval Cultures, Volume 6, Number 1, Spring 2017, pp. 163-166 (Review)

Published by Johns Hopkins University Press
DOI: *https://doi.org/10.1353/dph.2017.0006*

CATview: The Colored and Aligned Texts Tool

By M. Pöckelmann, Paul Molitor, and Jörg Ritter. Halle: Martin Luther University Halle-Wittenberg, 2015. <http://catview.uzi.uni-halle.de/>

CATview, developed within the Semi-Automatical Difference Analysis of Complex Text Variants (SaDA) project at Martin Luther University Halle-Wittenberg, is a JavaScript component that can be added to a digital edition to visualize the correspondence among parallel witnesses. The tool allows developers to add a simplified representation of the witnesses' alignment to the interface of a website, offering a quick, useful visual overview of similarities and divergences, and facilitating easy navigation of the edition.

CATview displays each witness as a series of colored boxes representing individual segments of the text, with corresponding segments from different witnesses displayed side by side. Where a witness omits a passage, a blank space appears in place of a box, and boxes can be shaded to show how greatly witnesses differ in their readings. The display may be zoomed out to offer a broad visual impression of similarity and difference across the entire text, or zoomed in to examine details of variation within a particular section. CATview can be displayed either horizontally or vertically and placed at any edge of the screen, allowing it to fit into a variety of interfaces.

CATview is distributed as one JavaScript and one CSS file. It requires the open-source D3 and Font Awesome libraries; both are bundled with the source code. (One undocumented method uses JQuery, also included.) It is available under the MIT license, permitting modification and redistribution, with acknowledgment, even in commercial publications.

The tool is designed for inclusion in a parallel-text edition. When CATview is fully implemented, readers can navigate within a text, clicking on a box to jump to the corresponding segment; the interface displays the reader's current position. Additional features allow the reader to filter the displayed text by selecting particular segments, and can work in conjunction with a search function to highlight sections of each witness in which a search term appears.

By default, CATview displays textual segments in blue rectangles shaded darker to indicate a greater degree of difference, resembling Juxta's color scheme. The current position is indicated by an orange bar, and search results are highlighted in yellow. These colors are customizable, but must be managed in two locations: some in the CSS, others by calling JavaScript methods. More substantial visual changes require modifications to the JavaScript code.

CATview performs no analysis—its purpose is strictly visualization. Editors will have to segment their witnesses, align the segments, and calculate degree of difference using other tools; CATview can then visualize the data and help navigate the text. Because CATview deals only with processed data and not with text directly, editors have a great deal of flexibility: they can divide their texts into segments (verse lines, sentences, paragraphs, etc.) and calculate difference in any way they see fit.

The data used by CATview takes the form of a list of witness names and a list of "edges." Edges "define an alignment of the witnesses' segments": an edge is composed of an array of numbers, one number for each witness, where -1 indicates that a witness has no reading at that segment and numbers from 0 to 1 show the degree of textual difference from other witnesses. So, in an environment with three witnesses, an edge where the first witness is authoritative, the second witness lacks the segment, and the third shows moderate variation might take the form [0, -1, 0.5]. CATview requires data in this format, specifying a value for each witness at each edge, which it uses to generate and shade the boxes.

As this data structure suggests, CATview can only visualize the linear alignment of texts divided into equivalent segments. It is not capable of displaying more complicated correspondences of text, such as overlaps, transposition, or displacement. Nevertheless, for straightforward correspondences, it provides a useful way of seeing textual difference.

The way CATview stores data imposes some requirements on page design. It identifies witnesses and edges only by their positions in arrays, so the HTML must be structured correspondingly. For instance, each witness's reading for a particular segment might belong to the "segment" class, grouped within a parent with the "edge" class, so that the eighth element in CATview's edge array corresponds to the eighth .edge element and the second .segment element for each edge represents the second witness. The ability to address edges or witnesses by ID might provide more flexibility, but would significantly complicate the data structure.

Adding CATview to a project is straightforward, requiring the implementer to include the JavaScript file and to execute a few methods

to configure the tool, load it with data, and initialize it. However, taking advantage of any but its most basic features requires additional programming. CATview provides the mechanism for displaying the alignment, but leaves it to developers to implement most user interactions. Accordingly, many standard behaviors—clicking on an edge to jump within the text, highlighting the current position—require additional JavaScript; CATview itself supplies only triggers and callbacks. Advanced features like highlighting search results and filtering the text based on user selection are even more complex. Fortunately, these techniques are documented on the CATview website; unfortunately, the documentation is not distributed with the source code. (The source ZIP file does, however, include helpful example webpages.)

CATview allows developers to extend the application by adding buttons to the interface and by manipulating the data, potentially enabling sophisticated features. Methods that add or remove edges and witnesses, for instance, suggest the intriguing possibility of tools for interactive alignment. However, the currently available methods are insufficiently robust even for an application that just loads witnesses dynamically. The included methods permit only adding, removing, or replacing entire edges at once; shifting one witness's segments by an edge or adding a new witness to an existing set requires rewriting each edge individually. Developers can of course implement such functions themselves, but including more ways of manipulating the data would allow CATview to meet more varied needs.

The interface works well in general when the navigation features suggested by the documentation are implemented. Several elements, however, could be improved. The interface does not scale as the browser window is resized, which also means that rotating a tablet can make the tool unusable. (Developers cannot simply monitor window size changes themselves, as I did not find a method to rescale the interface without reinitializing it entirely.) On the whole, it is fairly responsive to touch (tested on an iPad), though zooming did not work properly on a touchscreen.

Tools should provide more obvious feedback to users: it can be difficult to tell when a tool is active, because its icon does not change when clicked. Developers may want to add tooltips, as it can be unclear what tools like the Brush do; CATview provides no direct support for such labels. In addition, CATview does not directly support accessibility features, which will have to be implemented independently.

The helpful "scrollspy," which shows the current position by highlighting the first edge visible on the screen, would perhaps be even more

useful if CATview instead highlighted all visible edges. That way, readers could see not only their position, but what portion of the text they are looking at. It would also ameliorate a display oddity: because the last segment of the text will rarely scroll to the top of the screen, the scrollspy never reaches the end, making it look as though the reader has not scrolled all the way through the text.

There appear to be plans for further development, judging by the TODO statements appearing in the JavaScript, but neither the source code nor the website includes version information. (I downloaded my copy of CATview on October 17, 2016.) I hope the developers will label future versions to make it easier to track the tool's development, and will consider placing CATview in a public version control repository.

CATview suggests the great potential of a toolkit of modular components for digital editions: elements that could be added without having to develop them specifically, and could provide a familiar interface across varied editions. Those who need to visualize complex relationships among texts or build dynamic applications may find CATview insufficient for their needs, and editors who are not equipped to generate their own alignment data will be unable to use it. On the whole, CATview provides a useful, visually interesting mechanism for interacting with parallel texts, which can be tailored to the needs of a particular project without the expense of developing a system from scratch. It offers a useful addition to digital editions, and merits consideration by their producers.

Paul A. Broyles
North Carolina State University