A PDF Reader and Visualizer to Support Nonlinear Reading

Jordan A. Cazamias

Abstract—FlatPDF is a web-based PDF reader intended to aid readers of digital content to perform tasks other than linear reading. All pages are represented on the screen at once in order, giving the user a stronger spatial sense of where they currently are in the document. It also includes a search visualization that allows for softer, multi-word queries, as well as a stack-based navigation system that allows users to move back and forth between recently visited pages. These features are intended to encourage the user to travel between distant sections of a document and feel confident knowing they can return to previous sections. FlatPDF is intended to serve as a reader and visualizer in one, where visualizations are directly embedded in the reading experience, and is open to extension via different interchangeable visualizations.

Index Terms—Data Visualization, User Experience, Document Layout

1 INTRODUCTION

Digital books have many advantages over their physical counterparts: They’re more space-efficient, easily shareable through cloud storage, and typically offer additional metadata that can be used to enhance the reading experience. Yet, in terms of user experience, many still prefer paper. Why?

The discussion of how to improve the experience of reading digital books can branch into many parts of UX design, but to me, this problem is largely a visualization problem. Most digital book readers can display a wide variety of documents of various lengths, but they are generally laid out just like a book. However, digital books are not constrained by the same physical limitations of paper books. Books, by their very nature, are linear collections of pages which are bound together. The standard sizes of pages are easy to hold in one’s hands, and the binding of pages keeps them in their proper order. This comes with some drawbacks; for instance, it’s essentially impossible to view more than two pages at a time while reading, and the process of flipping back and forth to find a choice passage can be cumbersome. This is especially apparent when performing nonlinear reading, such as with textbooks and research papers where, oftentimes, different related concepts are located in non-contiguous parts of the document.

Because digital books are not bound by the same constraints of books, the possibilities of alternative layouts should be explored. What’s more, because digital books store both the content and metadata in an easy-to-parse format, there are plenty of opportunities to add additional visualizations to further aid reading by highlighting the relationships and structure within the text. This could be done to find connections between totally separate passages, such as with Concept Mapping [2] and Text Network Analysis [8], displaying a summary using machine learning-driven summarization [7], or much more.

This project is an exploration on representing digital books in an alternative way, one which helps support nonlinear reading and which allows for embedded visualizations that further enhance the reading experience. These visualizations should aid the user with oft-neglected tasks such as understanding the “gist” of a book and finding relationships and patterns among the different topics covered.

2 RELATED WORK

One of the primary inspirations for this work was a device created at interface designer Bret Victor’s lab. As described by writer Craig Mod:

[Bret Victor Magic Bookshelf] Against the far wall of [Bret’s] lab’s library stood a 10-foot wooden bookshelf. It was stuffed with manuals on the history of computers and programming and interfaces, novels and countless non-fiction books.

From behind me, Bret said: ‘Watch this,’ and pointed a small green laser at one of the books. The spine - the physical spine - lit up and above the bookshelf the book itself exploded onto an empty swath of wall. The entirety of its contents, laid out page by page by some hidden projector. The laser tracked by some hidden constellation of cameras. In his hand, Bret held an iPad, and as he pointed the laser at various projected pages they appeared on his device. As he slid from page to page on the iPad, the corresponding pages on the wall enlarged. It was a way to view both the macro and micro of a book - the overarching structure of the whole and the minutiae of the paragraph.

Bret’s multi-modal representation of books is most fascinating because it encourages the user to take on a more exploratory role as they...
read. They have all the details of the page contained in the iPad, and yet can easily refer to a spatial map of where they are in the book and chapter at all times.

In addition, for displaying large quantities of data with a linear ordering, such as calendar events or code files, I looked to Mackinlay’s Perspective Wall, which uses perspective distortion to simultaneously display the details of a task and its context [4], and Seesoft (Figure 3), a large-scale software visualization tool that allows for line-by-line highlighting of code based on different metrics [1].

![Fig. 2: Mackinlay’s Perspective Wall](image1)

Fig. 2: Mackinlay’s Perspective Wall

In addition, for displaying large quantities of data with a linear ordering, such as calendar events or code files, I looked to Mackinlay’s Perspective Wall, which uses perspective distortion to simultaneously display the details of a task and its context [4], and Seesoft (Figure 3), a large-scale software visualization tool that allows for line-by-line highlighting of code based on different metrics [1].

![Fig. 3: The SeeSoft visualization tool](image2)

Fig. 3: The SeeSoft visualization tool

Research also exists on visualizing books within the context of a larger library (such as Cubaud in [2], National in [6] and Rauber in [9]). This is a great possible extension of the project, especially because comparisons between documents could provide an interesting visualization opportunity. This will be talked about in the Future Work section.

3 METHODS

3.1 Design Principles

While the aforementioned visualizations are interesting, above all else I wanted to create a visualization tool for digital books that was not just a visualization tool. Rather, my goal was to create a tool that would serve as both the reader and visualizer. Therefore, the representation of the book should aid in visualization, and on the other end, the visualization should be directly embedded within the reading experience and augment it.

To achieve this, I chose the following principles to work with:

- Every page should be represented on the screen at all times
- The user should always be able to read the currently open page
- The spatial arrangement of the pages should not be broken

Adding these constraints narrows down the kind of visualizations that are possible. For instance, rearranging the pages to rank them is out of the question, and changing the sizes of pages is somewhat dubious as well. Therefore, color ended up being the central mark to differentiate between data.

![Fig. 4: Concept of the search visualization. The point of this alternative search is to encourage a different style of searching, where users should be able narrow down their search just by adding more relevant words](image3)

Fig. 4: Concept of the search visualization. The point of this alternative search is to encourage a different style of searching, where users should be able narrow down their search just by adding more relevant words

3.2 Implementation

The implemented program is called FlatPDF. It is a web app that is targeted specifically towards displaying PDFs. The principles of design used here could apply to any digital book format, but I chose PDFs specifically because they are practically ubiquitous online and easy to obtain and parse using open source tools.

The app uses two major libraries behind the scenes: PDF.js, which is provided by Mozilla and performs the PDF rendering, both to a full-sized page and to the thumbnail images; and D3.js, which provides the metadata visualizations on top.

3.3 Search Visualization

Perhaps the most important feature that digital book software offers over paper books is ability to search for a specific query word or phrase. The typical implementation of this search is either an exact word/phrase match or a substring match. This type of search is great when the reader has a specific query term in mind, and is effectively the digital equivalent of a physical book’s index.

For queries where the reader does not have a specific query term in mind or where they have many possible terms, this search is not as effective because any query with multiple words that don’t comprise a phrase is very unlikely to match anything. Essentially, this means that these kind of multi-word queries must be broken down into several individual queries instead.

FlatPDF does not provide exact phrase matching: instead, it samples words individually without considering the order of words, and scores pages based on the number of unique word matches it contains.

More specifically Given n pages and a query with m words, I assigned each page a score based on the following formula:

\[ \text{Score}_i = \sum_{m} l(m, i) \]

where

\[ l(m, i) = \begin{cases} 1 & \text{if word } m \text{ occurs in page } i \\ 0 & \text{otherwise} \end{cases} \]

The borders of the pages with nonzero scores are colored along this color ramp provided by ColorBrewer:
The main advantage of this scheme is that it’s simple to implement (using a single hash map that could be pre-computed) and it’s more flexible than exact word/phrase matching. However, some information is not fully utilized. For instance, this search scheme does not consider the order of words, how many times the same word appears on a page or, if a page matches multiple words, how closely the matching words are to each other.

Ultimately, the important thing to consider for the search query design is the task it is meant to support. In this case, the target task was finding new parts of the document to jump to and read because the reader has a general interest in the topic(s) that their search query covers. By “general interest”, however, this means that the reader is not necessarily looking for an exact match to their query, but rather, matches that hold similar meaning. To help with this, search queries need to be less strict with matching, and it’s very important that the strongest-matching pages easily stand out.

3.4 Stack Based Navigation

The stack-based navigation feature is meant to enhance the user experience of performing a query; namely, it allows a user needs to save their “state” when jumping from page to page. This should embolden the user to explore the document more since they will be more confident that they can return to their familiar place. And with that, in theory, a user would use the query feature and explore the match results more liberally.

3.5 User Study Design

To test the efficacy of the program, in particular the search visualization, I assigned users to work with either FlatPDF or a control reader (in this case, Adobe Reader CC). I familiarized them with the features of each reader and then gave them a task. The task was to find an unknown term in the document using a small list of suggested keywords to search for. After they completed this task, I switched them to the other reader and gave them a different task with a different document (to keep their search experience unbiased).

4 Results and Discussion

Overall, the most interesting behavior noted was the process by which users performed a search. When using the control program, users would think carefully about the "best" keyword to use in order to find their target term. They then used only this word, and scanned through all of the matches in linear order until they found the target.

While I was expecting this behavior to change when using FlatPDF, the result was similar. Users liked having a better spatial sense of where they were in the document, but they tended to stick with the old pattern of searching for one keyword generally works well. However, the effectiveness of this search heavily depends on the choice of this key word. Choose poorly, and the resulting matches could be a total dead end. Worse yet, there’s no way of knowing whether the matches are a dead end unless you check them all.

The FlatPDF search system takes some of the guesswork out of choosing a keyword, because users can add multiple keywords and they do not all need to be high-quality. Generally, as long as one of these keywords is high-quality, the target page will stand out. The quality of the results is still somewhat dependent on the quality of keywords chosen overall, as demonstrated in figure 5.

5 Future Work

The system is still in a prototype state. The existing features themselves can be improved on, and new features and visualizations could be added.

In particular, the search visualization could stand to go through some more iterations. Overall, I believe there is promise for this visualization, as it could change the way users perform searches in a way that’s less pre-planned and more exploratory. Also, when used with only one word, it seems to be as effective as the traditional search feature. One important way to help with that is to soften the matching process even more, where even synonyms of a word will match the word. This way, users wouldn’t even need to have the exact correct

Fig. 5: Search results for two queries for the SIFT algorithm in the same document. Left: results with the query “edges corners scale feature detection”. The search results clearly point out the most useful page (in red), but the query may be more difficult to make without prior knowledge. Right: results with the query “feature detection algorithm”. The query reflects more of a general query but the results are not as useful.
word in mind when performing a search, and they’d still be able to find matches that are close to the spirit of what they were searching for.

The representation of other aspects of the reader could be improved as well. The stack-based navigation scheme, while useful, currently does not offer any visualization other than showing a separate list of recently visited pages. One could easily imagine a “breadcrumb” visualization where the pages that a user visits are visibly connected with lines. As for the thumbnail images of the small pages, they don’t necessarily have to be shrunk down versions of the original pages. It would still be useful to preserve visual “landmarks” of the page, especially large images. Perhaps an iconified version of the page, where images are quantized to a few colors and possibly enlarged, may work well. Also, the layout of the pages should also include clustering for chapters or sections, to provide additional context and spatial landmarks for the reader.

For the general reader, one extra valuable piece that is still missing is a way to get the “gist” of the book, and optionally to prompt possible queries that the user could use to explore the document. Concept mapping is one possible way to do this, where pages of the book are clustered together by their overarching themes. Alternatively, each page thumbnail could display a few words that best describe the page. Or perhaps the query itself could be modified to show more prompts. Perhaps the Word Tree, as presented in [13], could be utilized to give the user suggestions as they are typing out a query. This feature is pretty open-ended but it would close the feedback loop for exploration. This way, they would have the means to form good queries, visualize the results of the queries, navigate to the most promising sections highlighted by the query, and repeat.

Finally, my goal for now was to create a representation for digital books that was very general, and also to provide some visualizations that would benefit the core reading experience that all PDFs require. However, let’s also consider the specific “class” of document that a user was reading, such as a research paper, a complex manual, or a textbook. It’s easy to think of specialized pieces of metadata that would be useful for each of these classes, and thus we could create different visualizations that would fit these classes. In other words, a large next step to take would be the prototyping of specialized reading environments.

For example, consider a reading environment where the document in mind is a research paper. These papers are full of marked citations, which are in essence hyperlinks to other papers in the local “citation network” that this paper resides in. Some common tasks that take place which are exclusively useful to research papers include:

- Comparing the paper to other papers that it cites
- Comparing the paper to other papers that cite it
- Understanding the local landscape of research that this paper resides in

These tasks can be partially accomplished using a bibliometric mapping tool like VOSViewer [11] to view the citation network of the paper as a graph. However, these tools generally don’t allow simultaneous reading of the paper and thus place a divide between the actions of reading and of exploring the network. What if these two actions were instead more tightly interwoven?

Regardless of the specifics of the document, however, the core act of reading is the same. When reading, especially with nonlinear reading, it’s important for the reader to understand where they are in the document, where they have been, and where they should go next. This is why the design principles used in this paper should still hold even if the domain were more specific. This is why it would be useful to explore visualizations for different domains of reading that still preserve the core design principles of this program.

The site is currently available to try at https://jaycaz.github.io/flat-pdf. Further improvements and additional visualizations may be added in the future, so the site may be different than shown in this paper!