Exploiting Coordinated Views for Scholarly Reading and Analysis

Francesco Poggi
*Dept. of Computer Science and Engineering
University of Bologna, Italy
francesco.poggi5@unibo.it

Paolo Ciancarini
*Dept. of Computer Science and Eng.
Universities of Bologna and Innopolis
paolo.ciancarini@unibo.it

Angelo Di Iorio
*Dept. of Computer Science and Eng.
University of Bologna, Italy
angelo.diiorio@unibo.it

Silvio Peroni
†Dept. of Classical Philology and Italian Studies
University of Bologna, Italy
silvio.peroni@unibo.it

Fabio Vitali
*Dept. of Computer Science and Eng.
University of Bologna, Italy
fabio.vitali@unibo.it

***Abstract*— There are several ways of reading scientific papers, and in general, documents. The sequential reading of the text flow can be complemented with alternative visualizations such as tree-based views, maps and charts. Infoview techniques are very promising in this context but not yet fully exploited. This paper presents a system called DocuDipity that is able to smoothly integrate infoview techniques, in particular a SunBurst visualization and a flow text reader, in a coordinated environment that helps researchers to read and analyze scientific articles. We discuss some end-user tests that showed that our tool is highly effective to find patterns and disclose some habits in writing articles. For instance, we used DocuDipity to study how the authors organise their content and use citations. The paper presents both the system and the details of the evaluation, together with some indications on future developments.**

***Index Terms*—Document visualization, information interfaces and presentation, scholarly articles, structural patterns, sunburst**

I. Introduction

In the last three decades, the advent of the World Wide Web and the consequent development of hypermedia content has produced a profound impact in the way people access, use and consume information. This fundamental shift has also involved the way researchers access scholarly information. For instance, the results of a study based on surveys performed in time span of more than 30 years show that the average number of readings per year per science faculty member continues to increase, while the average time spent per reading is decreasing[1][2]. The fact that the amount of time available for reading scientific articles is likely reaching a maximum capacity poses challenging issues.

Things are even more complex as scholars read documents in many different ways, depending on the goal they want to achieve and the time they can spend,our assumption is that they need different tools.

In some cases scholars need to scan documents thoroughly, nitpicking every single word and character; in others, a bird's eye view is enough: they browse pages, searching relevant parts, or they jump to specific sections, starting from the table of content. Consider for instance the citations. Incoming citations are checked for evaluation purposes: we just count them to measure the impact of a paper, and, as a consequence, of a given researcher. When we review a paper, instead, we also look at outgoing citations so as to evaluate if it is up-to-date and fits a given research area. We might also scan self-citations in order to get indications about the relevance and impact of that work. It is also common to identify the citation context (the sentence in which a work is cited) or the citation function (the reason why a work is cited) to better characterise and weight citations. A full analysis of the citations requires researchers to *scan* documents. To do so, the *readers* jump from the text of the paper to the bibliography, alternating *focused readings* to *overview readings*, and moving from one part to another.

The heterogeneity of these *reading patterns* is the main focus of this work. We limit the concept to the definition of Hornbæk et al. in [3]: "how readers navigate and manipulate documents as they try to accomplish their aims with reading."

Our results are directly connected to those of [3]. The authors investigated the support provided by a representative set of visualization tools to scholars in their reading and analysis activities. Three interfaces were compared in the study: a linear interface (where a document is shown as a linear sequence of text and pictures), a fisheye interface (where the document parts considered of minor importance are initially distorted below readable size, but can be expanded and made readable if the user clicks on them with the mouse) and an overview+detail interface (where a linear interface called detail pane is complemented on a side by an overview of the document, that can be used navigate the document - by moving the viewport of the detail pane).

The users of the experiment were asked to perform two types of tasks (i.e. writing an essay and answering questions) on the same set of documents using the three aforementioned interfaces. Experimental results show that the linear interface is in many ways inferior with respect to other ones: it has the

DOI reference number: 10.18293/DMSVIVA2019-021

lowest satisfaction and usability scores, and leads to lower essay grades. The fisheye interface is the fastest interface, but subjects using it obtained lower incidental learning scores, resulting in outcomes of lower quality. The conclusion is that the fisheye interface is more appropriate for tasks where a detailed understanding of documents is not the main aim (e.g. relevance judgements in information retrieval systems). The overview+detail interface led to the highest quality outcomes, and subjects strongly preferred this interface. However it was slow for question-answering tasks. The authors explain this fact with the overall higher affordance of this interface, which engages users and encourages them to perform deeper analyses. In fact, the study of the experiment logs showed that users spent more time to further explore the document after the first contact, and also that - on average - these explorations took more time than in the other interfaces. In other words, the frequency and duration of further explorations are higher than those of the counterparts, increasing the quality of the results.

The basic idea of our work is to improve the overview+detail approach by combining *coordinated views* so as to help researchers to discover traits of the scientific articles, **without necessarily reading their full content**. The system we present here, called DocuDipity[2], combines the hypertext flow layout of the article, in which readers can scroll the sequential content of the paper and read it, with a novel way of displaying tree-based documents, in particular XML ones, based on SunBurst visualization [5]. SunBurst is a form of radial visualization designed for depicting and summarizing in a very compact form even complex information hierarchies. Initially proposed to surf filesystems, SunBurst tools have been applied to various contexts, e.g. for summarizing navigation paths [6] or showing firewall configurations [7]. However, to the best of our knowledge, it has not been used in document-centric systems as the one we propose in this paper.

The main contribution of this work is, in fact, to demonstrate how the SunBurst visualization can be successfully combined with the flow layout in order to help scholars in their tasks. Our goal is not only to confirm the exploitability of the SunBurst but also the strengths of *coordinating* these two views in a single integrated environment that allows readers to easily access different aspects at different levels of the same paper. DocuDipity is also highly customizable, since it relies on editable rules that can be shared and refined by the users. This paper presents a set of rules, but the system is totally independent from them, and can be extended to carry out other analyses.

This paper describes the system and some experiments in either using the pre-loaded rules or writing new ones. We tested DocuDipity with 25 users studying the articles' internal structures and hierarchical organization, the self-citations, and some writing styles and authoring habits. The results of the test proves that DocuDipity is effective and usable. Some limitations still exist when editing rules but the overall users' perception was very good, as detailed in the experiments section.

The rest of the paper is organized as follows. Section II describes some related works, focusing on similar visualization techniques. Section III presents our solution in detail. Section IV describes how DocuDipity, and in particular its rule-based engine, can be used to support document readings and analysis, together with some examples of applications and findings. The experimental evaluation is presented in Section V, while some discussions and conclusions are in Section VI.

II. Related Works

One of the most popular principle in visual analytics tools is to provide an overview of the entire information space, so as to give users the context to support their browsing and search tasks[8]. This principle is at the base of many visual analytics systems such as Jigsaw [9], a popular tool that provides coordinate views for investigating concepts (e.g. places, persons, dates, etc.) extracted from text reports. A good overview about these technique is provided by [10]. Only a few works explored the application of this principle for supporting non-linear reading paths in large document collections. Most of them focused on supporting the navigation between documents in large document collections, or searches within a set of predefined keyword/features [11] [12] [13].

Another relevant issue concerns how to visualize the logical structure of a document. This problem may be reduced to the well-known problem of visualizing hierarchies. In times where typical data sizes grow much faster than the available screen sizes, a first element to consider is the efficient use of space. For this reason, implicit hierarchical visualizations (i.e. those that resort to an implicit representation of parent-child relations by positional encodings of the hierarchy items) must be preferred to explicit techniques (i.e. those that explicitly show parent-child relations as straight arcs or lines) [14].

One of the first and more popular alternatives is the Treemap [15], a space-filling slice-and-dice technique based on a rectangular layout. In a treemap, each element of a tree is depicted by a rectangle, which is then tiled with nested rectangles representing sub-branching. The color and area of each item correspond to attributes of the item as well: for example, these visual variables may be used to encode the type of the element and the number of contained characters, respectively. A plethora of variations and improvements of the initial treemap techniques have been proposed over the years in the literature (e.g. 3D Treemap [16], Triangular Aggregated Treemap [17], Quantum Treemap [18], Cascaded Treemap [19], etc.). For representing document information, one of the most suitable alternatives is specific kind of Treemap named Ordered Treemap [20], since it has the property of preserving the order within the document hierarchy.

[2]The name DocuDupity stresses on the *serendipity* of the system and its ability to make unexpected behaviors emerge, following the Kingrey's definition of *information seeking*[4]: "information seeking involves the search, retrieval, recognition, and application of meaningful content. This search may be explicit or implicit, the retrieval may be the result of specific strategies or serendipity, the resulting information may be embraced or rejected, ... and there may be a million other potential results."

The SunBurst technique [5], which we adopted in our tool, is another relevant alternative based on a space-filling visualization approach that uses a radial layout to represent tree-like structures. In SunBurst, items in the hierarchy are laid out radially, with the root element at the center and deeper levels further away from the center. Comparing Treemap and SunBurst, the former has a longer learning curve and a less explicit portrayal of the hierarchy structure, as described in [6]. Apart from these two techniques, which are the most relevant, a wealth of implicit tree visualizations have been proposed in the last 30 years. A summary of these methods and approaches are described in [21].

Widening the scope of the analysis without limiting to implicit techniques, a quite complete catalog[3] of tree visualization techniques is presented in [22]. Two of the most notable examples that apply such a tree-based visualization to documents are the Cone Tree [23] and Hyperbolic Tree [24]. Cone Tree is one of the first focus+context technique that embeds the tree in a three dimensional space. An expensive 3D animation support is required, since the tree has joints that can be rotated to bring different parts of the tree into focus. Another limitation of Cone Tree is its scalability: in fact, trees with more than 1000 nodes are difficult to manipulate. The Hyperbolic tree is the two dimensional counterpart of Cone Tree. The basic idea is to use a radial layout, and put the nodes in focus in the center, while out of focus ones are compressed near the boundaries. A better use of space, and the relatively modest computational needs, makes Hyperbolic tree potentially useful on a broad variety of platforms.

More recently, document visualization research has focused on the development of tools and techniques to support specific task relevant to document analysis. Usually, these tools require a pre-analysis of text to extract and compute a set of relevant features, that are given as input to an engine that produces a visual representation of the document based on them. For example, in [25] Kleim et al. used a pixel-based technique, which they call "literature fingerprinting", to understand and visualize document signatures, such as vocabulary richness and sentence length. Three coordinated views of document are provided to explore the feature values of text corpora at different levels of detail. Another notable example is DocuBurst [26], which infers word relationships within documents by traversing the WordNet hypernym graph, and applies a radial layout to show the word hierarchy.

III. DocuDipity

In this section we describe DocuDipity [4], the interactive web-based tool we developed to support the exploration and analysis of heterogeneous document collections.

The DocuDipity interface is composed of four main parts, as shown in Figure 1. The navigation bar on the top lets users select the document to investigate, whose content is presented in the two different and coordinated forms in the central area. On the left, a view based on the SunBurst technique provides a summarization of the document structure, while on the right side the content of the document is displayed as an hypertext.

Finally, users can select coloration rules to highlight relevant features and patterns in the document under investigation. Each rule changes the color of some selected elements in the document. users can modify or create their custom rules to support personal investigations, and interactively evaluating them on-the-fly. Rules are written in Javascript and CSS and will be briefly described below.

It is worth noting that DocuDipity is a general tool that can work on *any* XML document, without requiring any background information about the format documents are written in. In fact, the DocuDipity engine is based on the structural pattern theory[27]. By leveraging the pattern identification algorithm described in [28], DocuDipity is able to extract relevant information about the structure of the set of documents provided as input, and use it (a) to visualise the documents and (b) to provide an analysis framework to inspect their content easily.

A. FullText View

The FullText view provides a classical reading environment where the content of the document is shown as an hypertext. As all the other DocuDipity components and features, also this view is generated through an automatic analysis and segmentation of the document. The model of the document under investigation used by DocuDipity engine is based on the theory of structural patterns [27] and any textual document encoded in XML can be loaded into DocuDipity.

To generate the hypertextual representation of documents, DocuDipity converts XML documents into HTML composed of generic containers, blocks and inlines associated with some JavaScript code and CSS rules. For instance, the Fulltext view highlights the logical structures of the document (e.g. headings, sections, subsections, paragraphs, text fragments, structured data, etc.). References to figures, tables and bibliographic items are converted into hyperlinks, and other elements (e.g. blockquotes, footnotes, etc.) are collapsed and can be expanded on request. DocuDipity also leverages the pattern-based analysis to automatically generate the table of contents shown at the beginning of the document.

B. SunBurst View

DocuDipity uses the SunBurst technique [5] for providing an explicit but at the same time very compact view of the overall structure of documents. In the SunBurst, items in the document hierarchy are laid out radially, with the root element at the center and deeper levels farther away from the center. In addition to containment, the SunBurst view also preserves information about the element order, by starting to draw elements from midnight, and proceeding clockwise.

Other graphical attributes are used to highlight other information about the document under focus. For example, the angle swept out by each element corresponds to the number of characters contained, even recursively, by it.

[3]The interested reader can find an interactive catalog of the tree visualization techniques at the address http://treevis.net.

[4]DocuDipity is available online at http://eelst.cs.unibo.it/docudipity2

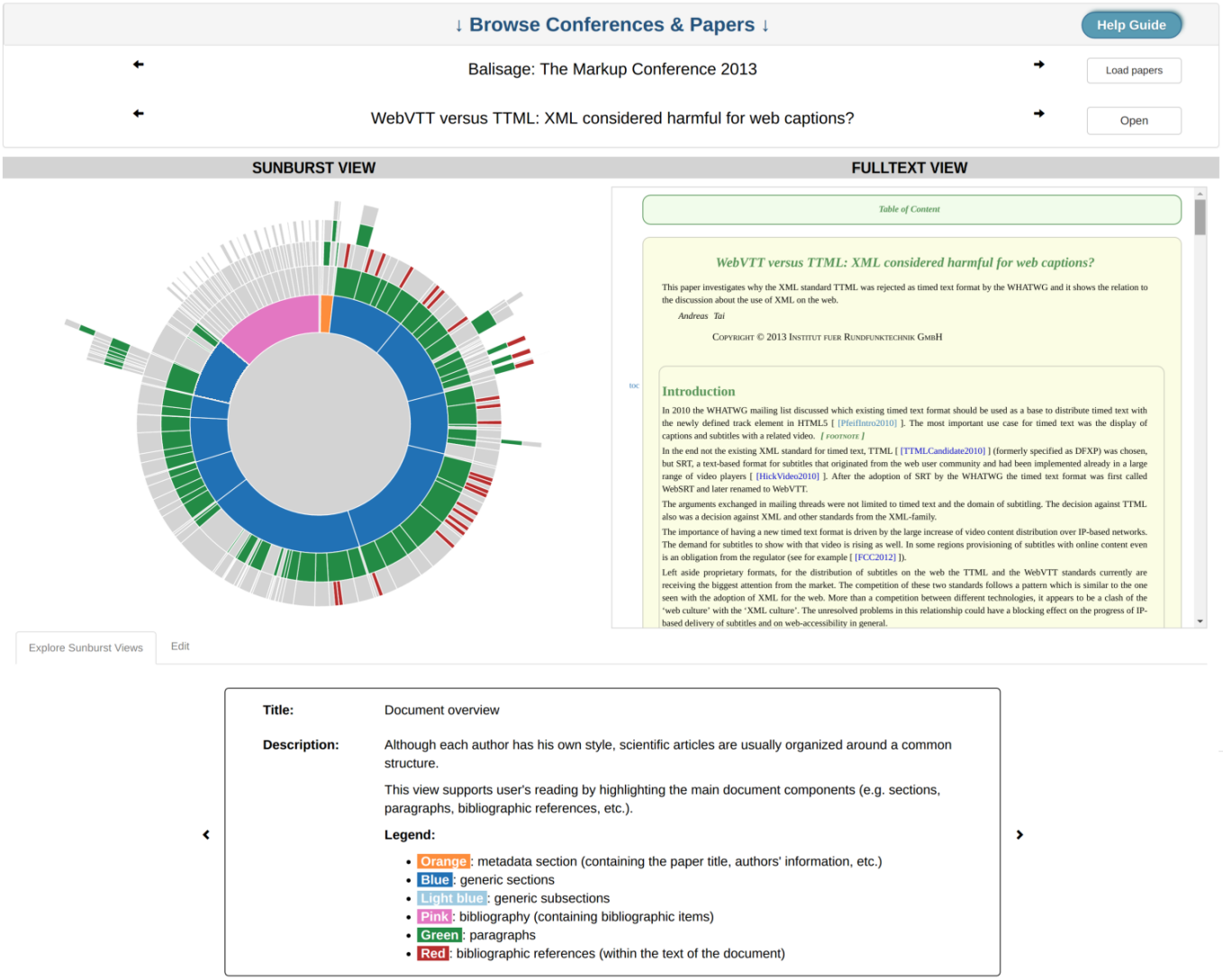


Fig. 1: An overview of the DocuDipity interface. The user's analysis starts with the selection of the document to investigate (top area), whose content is displayed with the SunBurst technique (center left) and as an hypertext (top right). In the area in the bottom, users can select predefined coloration rules to highlight relevant document features, or edit new ones and interactively evaluate them on-the-fly.

The SunBurst view has been chosen among the others representing tree structures (e.g. direct graphs, treemaps, etc.) for two main reasons: (i) because it is a space-filling technique that provides a very compact summarization of the document structure, and (ii) because it is more effective and easy to learn (i.e. it has a shorter learning curve) than the other main alternatives [6].

The default coloration rule highlights the most relevant components of the document (i.e. sections, bibliography, metadata, paragraphs, citations, etc.). The document in Figure 1 for example, after an initial metadata section (in orange) contains nine sections (in blue: three quite short at the beginning, followed by two longer in the middle of the document, and four very short at the end), and a quite long bibliography (in pink). No citations are contained in the second part of the document (they are spread equally in the first four sections, plus two at the beginning of the fifth one). Users can switch between rules to analyze different aspects of the document, looking at the SunBurst view.

C. Coordinated Behaviours

The SunBurst view and the FullText view are coordinated during the user's investigation: when he/she clicks on an element in the SunBurst view, the FullText view scrolls to the corresponding element. Moreover, when the user hovers the mouse cursor over a text fragment in the hypertext view on the right, both (i) the text fragment in the hypertext view and (ii) the corresponding element and all its ancestors in the document hierarchy on the left are highlighted; similarly, when a user focuses on an element in the SunBurst view, the cor-

responding element in the hypertext view is highlighted. The coordination between these two views is crucial in DocuDipity, since it facilitates customized and personal investigations of document collections. The SunBurst view provides a compact overview of the document, and users can easily jump into the details of the document (e.g. by reading a text section in the FullText view) without losing the overall context. This is fundamental for supporting readings that are different from sequential ones, providing at the same time a general view and a more detailed one, and allowing users to jump among them.

This behavior is based on the so-called information-seeking mantra "overview first, zoom and filter, then details-on-demand" [29]. The idea is to develop interfaces that give a general context for understanding datasets by summarizing their most salient features (overview), reduce the complexity of the representation by removing extraneous information from the view and allowing for further data organization (zoom and filter), and finally reveal additional information on a point-by-point basis while the user interacts with the visualization (details-on-demand).

An important issue faced during DocuDipity development is that, as the document hierarchy grows in size, many items in the SunBurst view become small and peripheral slices are difficult to distinguish. Also this problem has been solved using interaction by allowing users to zoom in and out: when users double-click on an element, the SunBurst is re-drawn to show only the subtree rooted in that element (zoom in); to move up one step in the hierarchy, users can double-click on the central circle (zoom out).

D. Editing Coloration Rules

In the bottom area DocuDipity provides an editing area where users can define coloration rules that can be applied to the SunBurst view to highlight relevant features and patterns in the document under investigation. To this end, users need a language to translate their hypotheses into (even complex) conditions, and a method to verify their validity. Instead of creating new languages and tools from scratch, we decided to use well-known web technologies like JavaScript and CSS.

By switching on the "Edit" tab, two textual areas are shown in the bottom of the DocuDipity interface, as shown in Figure 2. In the former one, the user can use JavaScript (in particular JQuery 3) to select elements and to assign CSS classes to them. In the latter one, the user can define CSS rules to specify a style for the classes assigned by the JavaScript code. To facilitate the development of new rules, users can also inspect the XML source of the current document by clicking on the "inspect XML" button.

For example, the following excerpt is the JavaScript code for coloring paragraphs according to their length. This code assigns the class `para-short` to short-length paragraphs (i.e. containing less than 400 characters), `para-medium` to medium-length paragraphs (i.e. containing between 400 and 799 characters), and `para-long` to long-length paragraphs (i.e. containing more than 800 characters).

```
var limitShort = 400;
var limitLong = 800;
// Selects all the paragraphs
var toFilter = $('para');
// evaluates the provided function over
    each paragraph
toFilter.each(function() {
  // Counts the characters contained in
  // the current paragraph
  var textLength = $(this).text().length;
  // Tests if the current paragraph is
  // short, medium or long, and assigns
  // the appropriate CSS class
  if (textLength < limitShort) {
    $(this).addClass('para-short');
  } else if (textLength >= limitShort &&
             textLength < limitLong) {
    $(this).addClass('para-medium');
  } else if (textLength >= limitLong) {
    $(this).addClass('para-long');
  }
}
```

The following CSS rules associate colors to the elements belonging to one of the previously defined classes(i.e. green to short-length paragraphs, yellow to medium-length paragraphs, and red to long-length paragraphs), and paint lightgray all the remaining elements.

```
path {fill: lightgray !important;}
path.para-short {fill:green !important;}
path.para-medium {fill:yellow !important;}
path.para-long {fill:red !important;}
```

By clicking on the "view" button, DocuDipity evaluates both the edited JavaScript and CSS fragments over the document under focus, and updates the SunBurst view.

IV. Reading and Exploring Scholarly Articles with DocuDipity

In order to showcase DocuDipity we deployed a test installation on a collection of 211 papers from a niche computer science conference, called Balisage Markup Conference [5], active for more than 20 years, whose collection of papers is publicly available in XML format since 2008. The same installation has been also used for the use-case study described in Section V.

We loaded an initial set of rules covering some common tasks that researchers already perform today by combining different reading patterns. These rules are not meant to be exhaustive but to give readers indications about the potentialities of the tool.

The following coloration rules are currently available:

- *Document overview:* this view highlights the main document components (e.g. sections, metadata section, paragraphs, bibliography, citations, etc.);
- *Citations:* this view help users to study citations networks by highlighting the position of the citations (i.e. reference in the text of the paper pointing to a bibliographic entry);

[5]The complete proceedings of the conference are freely available online at the address http://balisage.net/Proceedings/index.html

Fig. 2: The editor panel to define custom coloration rules.

- *Self-citations:* this extends the previous one by distinguishing between generic citations and self-citations (i.e. references in the text the paper pointing to a bibliographic entry authored by one of the authors of the current paper);
- *Paragraphs length:* this view helps users to study different writing styles by using different colors to highlight short, medium and long paragraphs;
- *British vs American English:* this view help users to distinguish the sentences containing American English spellings and peculiarities (e.g. airplane), from the ones containing British English forms (e.g. aeroplane), or a mix of them.

We discuss two coloration rules here but similar considerations can be generalized to all others and to new ones.

A. Citation analysis

Let us consider, first of all, the use of citations. While reading a new paper, citations are primary tools to find related works and to quickly get an overview of the research space around the paper. Though citations are often aggregated in related works section, this is not always the case. In some disciplines authors tend to aggregate citations in the introduction, in some others to scatter them throughout the paper. The SubBurst view is very effective to locate citations and to go the relevant part of the paper. Consider now the review process of an article or a report: reviewers often check, for instance, the number of self-citations or the publication year of the cited paper or even the publication venue. Instead of reading the flat bibliography, a reviewer could exploit the SunBurst view to get this information and then continue reading, even jumping to the citation she/he thinks is more interesting and worth checking.

Given our research interests in citation networks for scholarly communications [30][31], we experimented a few DocuDipity rulesets that illustrate the positioning of citations (`xref` elements in the Balisage XML) within documents, and we found interesting and different patterns in how citations scatter in the sections.

A glimpse to the DocuDipity SunBurst chart can tell us immediately about how the paper is structured. For instance, we can immediately observe structures where:

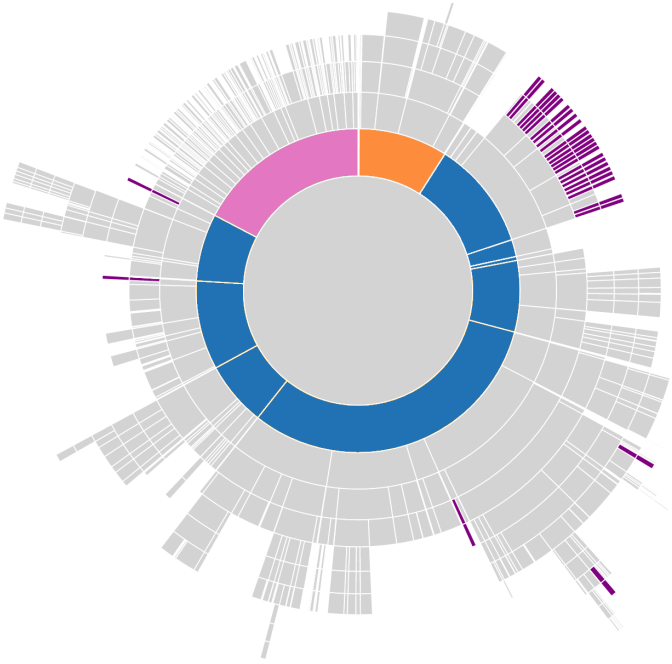
- authors use citations to provide the scaffolding and context of their own contribution, by collecting most of them in an initial section before introducing the core contribution, as in Fig. 3a; or
- authors use citations to provide a final comparison of their contribution against similar other ones in the relevant literature by collecting most of them in a section at the end, as in Fig. 3b; or
- authors scatter citations throughout the paper, so as to signify a continuous and meticulous dialogue between the contribution and the literature, as in Fig. 3c; or, even,
- authors (possibly practitioners, rather than scholars, not versed in the habits of scholarly communication?) that provide a rich set of citations in the final references section... totally not cited in the body of the paper, as in Fig. 3d.

B. Content and style analysis

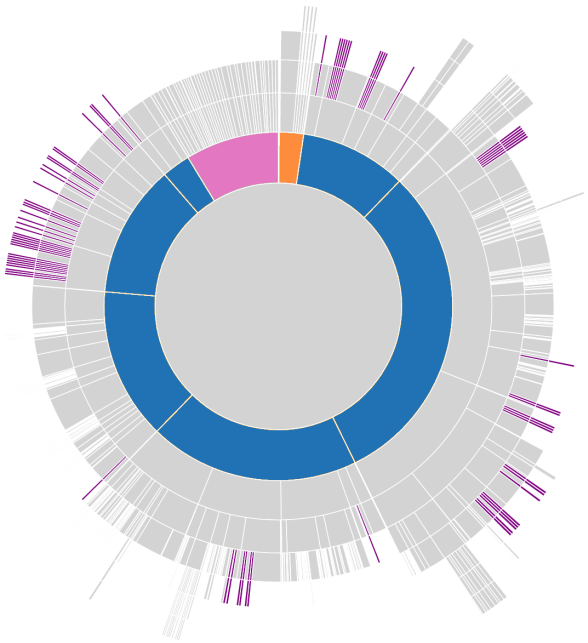
The stylistic and linguistic analysis of an article is another peculiar task of scholars when writing their own articles, or when giving feedback to other colleagues, or when reviewing articles for a conferences or a journal. Such an analysis goes with the actual content reading, which is and has to be a personal activity of the scholar. On the other hand, DocuDipity can be exploited to automatically identify some features of the article under evaluation.

For instance, we could consider the length of individual paragraphs mentioned in the previous Section and color gray the paragraphs that are about average, red the paragraphs that are much longer than average, and green the paragraphs that are way shorter. Is is easy to identify authors whose style uses long, winding, multi-conceptual paragraphs (e.g., in Fig. 4) from authors preferring paragraphs with short individual sentences straight to the point (as the one in Fig. 5).

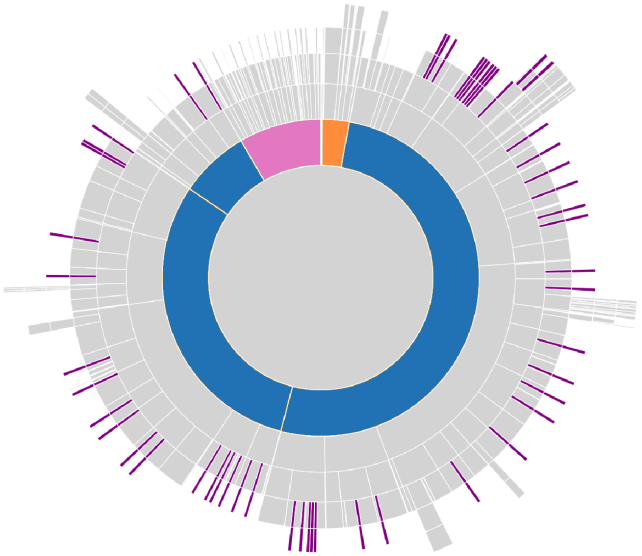
So what happens if you point the rules to a paper with multiple authors? Maybe you could distinguish the individual contribution of one author having a writing style different from the others, as in Fig. 6.



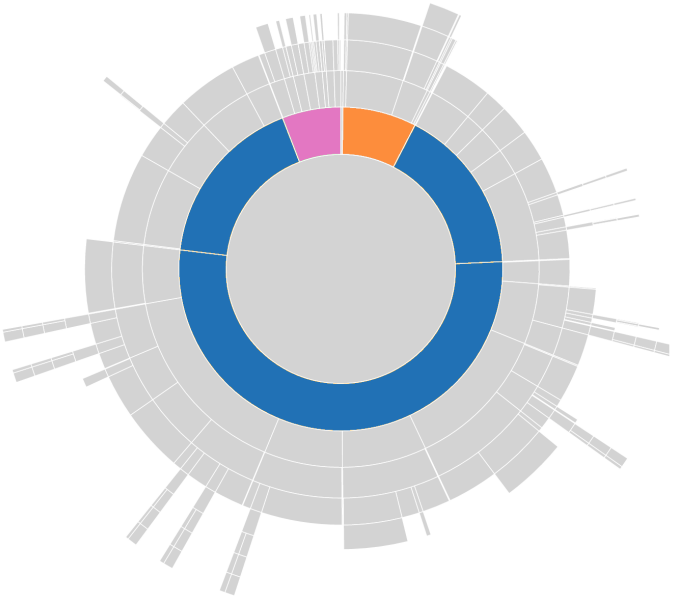
(a) The DocuDipity visualization of a paper with citations (i.e. purple slices) grouped in the initial part of the document. This writing style is common in the humanities.



(b) The visualisation of a paper with a ”Related Works” section at the end (top-left corner of the SubBurst).



(c) A paper with citations scattered throughout the body of the paper. This writing style is common in scientific articles.



(d) The visualization of a paper that contains plenty of bibliographic items in the bibliography section (in pink), none of which is mentioned in the document body. The SunBurst clearly shows an incomplete usage of XML for marking citations.

Fig. 3: Plots of papers where citations highlights different styles and structures

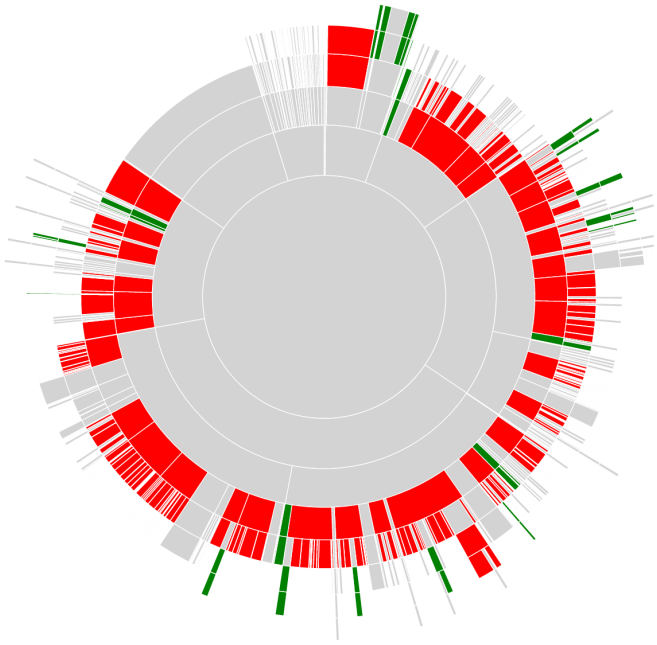


Fig. 4: A paper made of mostly long paragraphs (in red). The paper was written by a non-native English speaker.

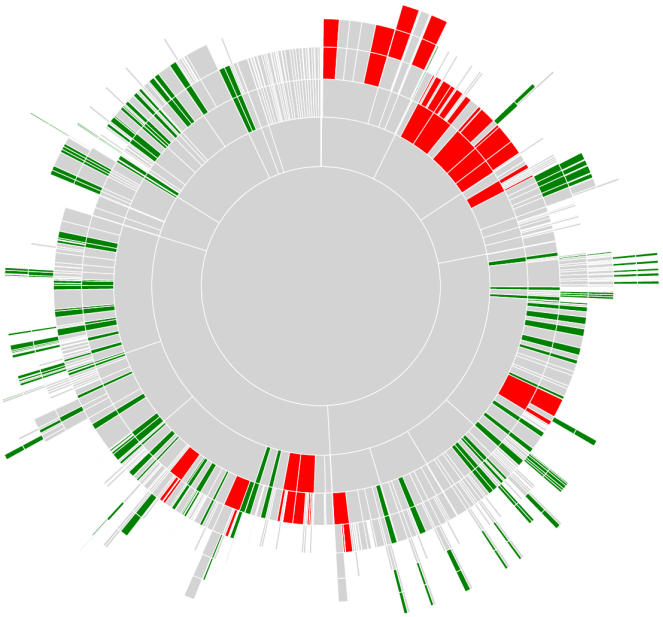


Fig. 6: A paper by multiple authors with a striking difference in the length of the paragraphs, possibly pointing out the individual style of one of the authors. The introduction (shown in the top-right corner) was in fact written by an author, who only worked on that section and whose style is more verbose than that of the others'.

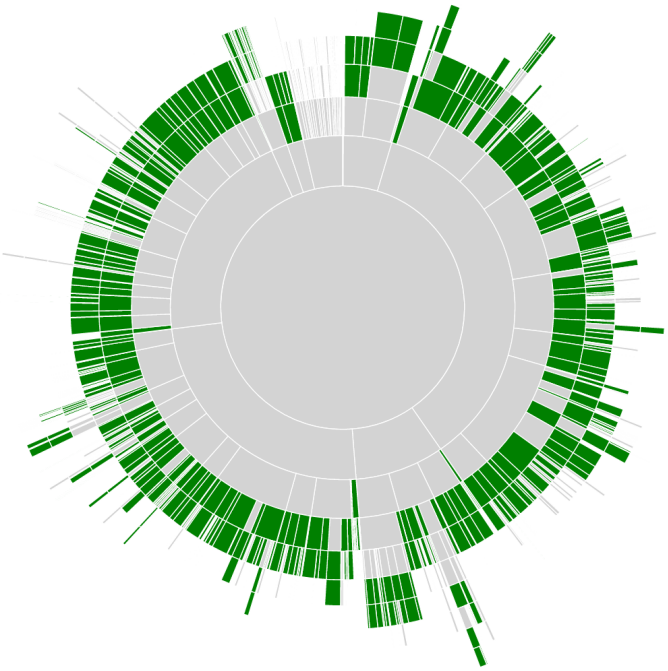


Fig. 5: A paper made of mostly short paragraphs (in green). The paper was written by an English native speaker, and is fluid and easy to read.

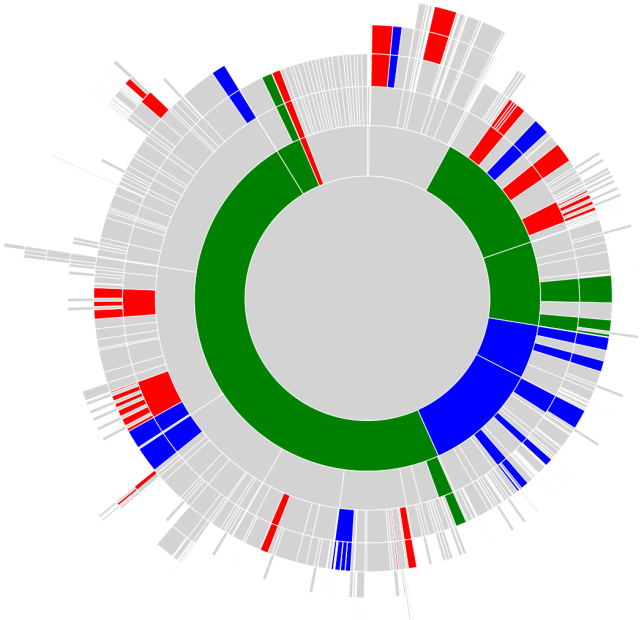


Fig. 7: A paper with a mix of American (in red) and British (in blue) words within – the inner circle are the first level section of the paper. The colour green is used when a section and/or paragraph has been written according to both UK and US English.

Or maybe, you could find yourself examining a multi-author paper by non-native English speakers, one of whose authors takes pride in his English, learnt during his post-doc station at the Oxford University, while the others have gotten theirs from the usual mish-mash of styles gained from magazines, blogs, novels, and Hollywood movies, plus the peculiar international English of scientific literature. Point DocuDipity over that paper, add rules that distinguish American English versus British English spelling and peculiarities, such as the *-ise/-ize* endings[6], and presto: you can immediately find out which sections were written by one individual and which by the others, as in Fig. 7.

V. Evaluation

We performed some end-user tests in order to evaluate DocuDipty and to identify its main strengths and weaknesses. Following the ISO 9241 standard[7], we measured the system in terms of *efficacy*, *efficiency* and *satisfaction*. The tests involved 25 users in total, with different backgrounds and expertises, and was split in two parts. In the first test we measured how DocuDipity can support researchers to explore scientific papers by using the rules already available in the system, and involved 20 users. In the second one we asked the other five (more expert) testers to add new rules to DocuDipity and to employ them for further analysis. These two parts are discussed separately in the following subsections.

A. Using DocuDipity

The first test was further split in two parts. We divided the 20 participants in two groups, each formed by 10 testers equally distributed in terms of professional level and skills. Each group was asked to analyze a set of scientific articles and complete two assignments. The source articles were 184 in total, taken from 7 different editions of the Balisage Markup Conference Series[8].

The first group was asked to use DocuDipity to complete the tasks, while the second could only use the official Balisage web site (where the articles are freely available). The site contains one web page for each article with a sidebar showing a clickable table of content. It is worth remarking that the two groups were given the same input data and tasks, so as to compare their behaviors.

The real test was preceded by a warm-up session, in which the testers were asked to complete a simple task so as to become confident with the platform they were about to test. In the case of DocuDipity, the users were also shown a 5-minutes demo of the system. After completing the two assignments, the participants were asked to fill some questionnaires about their experience, in the form of closed and open questions.

The assignments are summarized below. Each of them was repeated on three different articles:

[6]Remembering also that some words, such as *devise*, do not have spellings in the two dialects...

[7]https://www.iso.org/standard/16883.html

[8]The proceedings of the Balisage Series Conferences are freely available at http://www.balisage.net/

TABLE I: Execution time for each subtask.

	T1.1	T1.2	T1.3	T2.1	T2.2	T2.3
DocuDipity	2:38	0:38	1:00	1:15	1:16	1:32
Balisage	4:51	4:22	2:06	5:09	7:02	3:13

TABLE II: The percentage of testers who answered correctly for each subtask.

	T1.1	T1.2	T1.3	T2.1	T2.2	T2.3
DocuDipity	80%	80%	90%	100%	40%	60%
Balisage	40%	10%	30%	20%	0%	10%

T1. **Study self-citations**: given the title of a paper, the testers were asked to indicate how many self-citations it contains. We expected users to search a paper by title and to inspect both the bibliography section and the citations in the text. Note that the complexity of the task depends on the number of authors of the paper. For instance, if we just look for authors' surnames in the bibliography, in a paper written by dozens of authors (which is not unusual in some disciplines like physics) the task would require to manually repeat the search for each author. Moreover, it would be important to count self-references only once in case of multi-authored papers.

T2. **Study length of paragraphs :** the testers were asked to find the paragraphs whose length is under/over/in-between a given threshold. They were expected to check the length of the paragraphs by counting characters, manually or with an external tool.

Details of each evaluation are presented below:

1) Efficiency: We first measured how much time was needed by the testers to complete the tasks, comparing the users' performance with or without the help of DocuDipity. Table I shows the average execution time of the two tasks. For each task, we report the execution time on three different papers. As expected, a specialized tool like DocuDipity outperformed the plain sequential reading of the Balisage web site content. The difference however is significant and worth remarking.

2) Efficacy: We also studied if the testers could complete the tasks correctly, by manually comparing the answers given by each tester to the expected answers. Though some participants failed to complete some tasks, the overall efficacy was very high for DocuDipity. On the other hand, the efficacy of searching the same information on the plain web site was very low. Table II summarizes our results. The difference is evident, even more than we expected.

The efficacy of DocuDipity on assignment T1 (self-citations) was very high on all three papers taken into account. The interpretation of one specific user lowered the three scores of 10%. In fact she/he failed all three questions but in a consistent way: each self-citation was counted twice since the SunBurst visualization, in presence of a self-citation, colored

TABLE III: Comparison of the SUS scores for the tests performed with DocuDipity and Balisage

TestSet	n	Mean	SD	t	df	p	95% CI
DocuDipity	10	81	10.29	-	-	-	-
Balisage	10	64.25	7.17	-	-	-	-
Total	20	68.13	9.04	3.92	16	.0012	7.83 - 26.27

in red both the segment corresponding to the XML element and the segment corresponding to its textual content. For Task 1.1, for instance, the given answer was 4 while the correct one was 2. This suggests us that a clearer coloring scheme is needed for segments corresponding to text fragments. We do not have further elements to speculate on the other wrong answers, given by two other different users.

The results on Docudipity were less positive for Task 2 (length of the paragraphs). Apart from T2.1, whose percentage of success was 100%, the efficacy goes much lower for T2.2 and T2.3. The reason is that most testers had difficulties in manually counting the number of paragraphs highlighted in the SunBurst. They confirmed us that the visualization was clear but they got confused while counting the SunBurst slices. In fact, as discussed later, most of the testers suggested us to add counters and summaries of the results in the DocuDipty interface. Though not shown in the table, note also that the deviation between the wrong and correct answers was very low. Consider for instance T2.2: the correct answer was 19, while 30% of the users answered 18 and 10% answered 16. This confirms the difficulties that the testers experienced while counting the items. As expected, such difficulties are accentuated without the help of a specialized tool like DocuDipity.

3) Usability: After the completion of all assignments, the testers were asked to fill two questionnaires. The first one was a *System Usability Scale* (*SUS*) [32], a well-known questionnaire used for the perception of the usability of a system. It has the advantage of being technology independent and it is reliable even with a very small sample size [33]. The mean SUS score for DocuDipity was 81 (in a 0 to 100 range). The value shows how the users' satisfaction was very high, confirming that the coordinates views and the SunBurst visualization are well accepted by the testers.

In addition to the main SUS scale, we also examined the sub-scales of pure *Usability* and pure *Learnability* [34]. They gave us a more precise characterization of the users' feedback. The mean values for Usability was 81 and Learnability was 82. This means that the system was perceived both easy to learn and effective in supporting users to perform their tasks.

We also repeated the SUS test for the plain Balisage web site, just to compare this score and the DocuDipity one. In fact, the difference was very high: 64 against 81, which is statistically significant according to an unpaired student t-test ($t = 3.92$, $p < 0.01$) as shown in Table III.

In order to go deeper, we also included four open questions, only to the testers that used DocuDipity:

- What were the most useful features of DocuDipity to help you realise your tasks?
- What were the main weaknesses that DocuDipity exhibited in supporting your tasks?
- Currently Docudipity includes a limited set of analysis rules. Can you think of any additional one(s) that would be useful/interesting for you?

We subjected the text answers to a *grounded theory* analysis. Grounded theory [35] is a method often used in social science to extract relevant concepts from unstructured corpora of natural language resources. In opposition to traditional methods aiming at fitting (and sometimes forcing) the content of the resources into a predefined model, grounded theory aims at having the underlying model emerge bottom-up. We proceeded first with *open coding*, with the purpose of extracting actual relevant sentences – called *codes* – from the texts, and subsequently performed the so-called *axial coding*, which is the rephrasing and aggregation of original codes in concepts.

Figure 8 shows the concepts we extracted. The size of each bar depends on the number of codes which contributed to the corresponding concept. Note that the codes mentioned by less than two users were not taken into account.

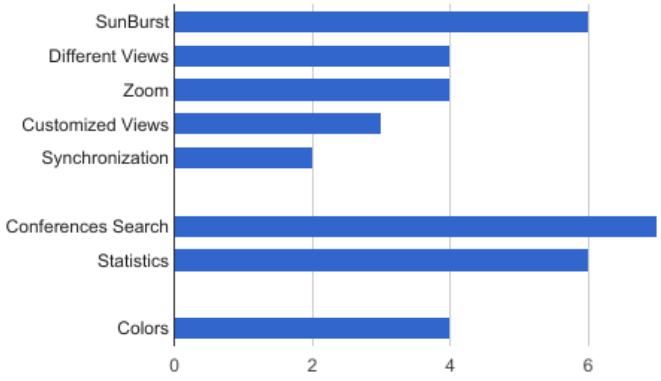


Fig. 8: The codes mentioned by at least two users, ordered by their frequency and grouped in positive and negative codes

The Figure is actually split in three parts. The top area contains the aspects rated as positive by the testers, followed by the negative aspects and, at the bottom, some mixed notes about the use of the colours in DocuDipity. The overall users' perception was very good, considering that the total number of positive codes is significantly more than the negative ones.

In particular, the testers appreciated the presence of the SunBurst view labelling it as a very effective tool for the proposed tasks. The coexistence of multiple views was also considered a further strength of DocuDipity, together with the possibility of zooming into the document' components and navigating the tree XML structure in an alternative way. This confirms our initial assumption that mixing plain text reading and infoview techniques is a valuable tool for scholars.

An element that we expected to be more appreciated by the testers is the synchronization between the two DocuDipity views. However, only two testers stressed its importance,

while all others used both views but never together: when looking a SunBurst segment, for instance, they did not read the corresponding element in the text flow. The possibility of customizing views and modifying rules was instead enjoyed by some more testers. The result is not optimal, we speculate, because the editing of the DocuDipty rules is not simple (as we will discuss later) and hinders the full exploitation DocuDipity.

The plot also shows that the main weakness of DocuDipity is the interface to surf conferences and papers. In fact, DocuDipity requires users to scan the list of conferences, load papers and then scan the list of the papers. This caused some confusion and impacted negatively on the feedback given by the testers. Most of them suggested us to include a search by title option. This was already in our plan but we had preferred to focus on the coordinated views and the SunBurst at this stage. With hindsight, this was not a good choice. The time needed to find papers and the frustration in missing some of them, impacted negatively on the overall judgment of some testers, more than we expected.

The second element which basically all testers agreed on is the fact that some statistics, for instance some counters about the number of paragraphs, sections and inline elements, would have make DocuDipity much more effective and usable. We liked this suggestion and decided to implement it. The last bar in the plot is about colours: some testers considered suboptimal the choice of colors, in particular, in the SunBurst view: the difference between the segments was not very evident and they had difficulties in selecting and extracting some information. A more balanced palette will be included in the next release of the tool.

The answers given to the last question (about new rules for DocuDipity) are also worth mentioning. Apart from one user, all others suggested to get information that could be easily spotted by DocuDipity. The suggestions can divided in three groups. Most users suggested to identify document components, such as figures, tables, charts, etc.. Other proposed to process the textual content of the articles, for instance by characterising elements according to their language and the presence of grammatical/syntactical mistakes or imperfections. Finally, some testers suggested to go deeper and highlight topics and claims within the articles. Apart from the details of each suggestion, we consider very positive their variety, confirming again the potentialities that users see in DocuDipity.

B. Editing rules in DocuDipity

In order to complete our evaluation we also asked five participants, over 10 who used DocuDipity, to create new visualization rules. Half of the testers were actually excluded since they declared to have a very limited knowledge of Javascript and would have add great difficulties in completing the tasks. Indeed, the main direction of our research is to make it simple to edit rules for such non expert users as well.

The test consisted of two assignments preceded by a warm-up and some training on the Javascript and CSS syntax of DocuDipity:

T3.1 **Studying lists**: testers were asked to add a rule for identifying lists whose number of items was higher than a given threshold. They were expected to find lists, count the items in each list and filter. The rule was then quite similar to one that they have already seen in the first part of the test.

T3.1 **Searching titled sections**: testers were asked to add a rule for identifying sections whose title contained the string ‘XML’ or ‘XSLT’. The expected rule was slightly different from the previous ones and based on regular expressions.

After they completed the assignments, we asked testers to answer some open questions about strengths and weaknesses of the editing process. The overall feedback was positive but all testers confirmed that the editing process is still quite complex. There was no particular difference between the two assignments. As expected, the second one was a bit easier after having learnt how the DocuDipity rules work.

All the testers appreciated the possibility of editing rules directly in the interface and to verify results on the fly; the fact that the two DocuDipity views are synchronized was also considered very positive: the users needed to validate the results found by their queries and they were helped a lot by the side-by-side view.

On the other hand, they all agreed that editing rules is not simple. The testers complained that basic Javascript skills are not enough but some knowledge of JQuery, and in particular of selectors, is needed. Most testers suggested us to add a least some documentation about JQuery selectors and API, so as to mitigate this issue. Two testers also stressed on the difficulties in debugging their code, as they basically had to use directly the Javascript console. They also spotted some constraints in the CSS classes that were not well documented (we fixed it now). A more integrated editor and debugger would definitely be helpful.

One tester also complained about the font size of the interface, though all others gave positive feedback about the overall organization of the editing and visualization areas.

In conclusion, DocuDipity rules were well accepted but still a bit difficult to write and some simplification is needed. We are envisioning an intermediate layer that takes as input visualization rules written in a simplified language and convert them in JavaScript in a transparent way. For instance, a very simple XPath expression would be enough to handle the previous case. Anyway, DocuDipity is totally independent from a particular framework, and any other tools can be used (e.g., EXT-JS) even instead of those included in the current implementation.

Connected to this aspect, there is also the need for a library of visualization rules within DocuDipity. The goal is to create a shared *reading* environment in which users can experiment new rules, write them rules incrementally and reuse them. This opens very interesting perspectives: the initial experiments of one user could be followed by other ones, and even partial intuitions could give input and urge new ideas. Existing rules might be also used to take inspiration by inexperienced users.

VI. Conclusions

This paper presented a system that exploits coordinated views to analyse scientific articles. In particular, DocuDipity integrates a plain text sequential reader and a SunBurst visualization. These views have been successfully applied to explore scholarly writing habits and to study, for instance, the distribution of citations and the internal organization of the articles' content. DocuDipity proved to be very effective in our tests. On the other hand, the editing process was still considered quite difficult and we are studying how to make it simpler and faster.

Another direction we are investigating is the integration of further views in the overall interface, based on infoview techniques. In fact, these techniques have been successfully deployed in data-centric applications but not yet fully investigated on documents.

Our plain is also to apply the DocuDipity analysis to other documents and other domains. In particular, we are keen to investigate the usage of well-know XML languages, like TEI or DocBook, in structuring real document. To this end, we would also like to make DocuDipity easy to use to other scholars with no expertise nor skills in Computer Science, e.g., scholars in Humanities, to discover things far beyond our knowledge.

References

[1] C. Tenopir, D. W. King, S. Edwards, and L. Wu, "Electronic journals and changes in scholarly article seeking and reading patterns," in *Aslib proceedings*, vol. 61, no. 1. Emerald Group Publishing Limited, 2009, pp. 5–32.

[2] C. Tenopir, R. Mays, and L. Wu, "Journal article growth and reading patterns," *New Review of Information Networking*, vol. 16, no. 1, pp. 4–22, 2011.

[3] K. Hornbæk and E. Frøkjær, "Reading patterns and usability in visualizations of electronic documents," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 10, no. 2, pp. 119–149, 2003.

[4] K. P. Kingrey, "Concepts of information seeking and their presence in the practical library literature," *Library Philosophy and Practice*, vol. 4, no. 2, pp. 1–14, 2002.

[5] J. Stasko and E. Zhang, "Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*. IEEE, 2000, pp. 57–65.

[6] J. Stasko, R. Catrambone, M. Guzdial, and K. McDonald, "An evaluation of space-filling information visualizations for depicting hierarchical structures," *International journal of human-computer studies*, vol. 53, no. 5, pp. 663–694, 2000.

[7] F. Mansmann, T. Göbel, and W. Cheswick, "Visual analysis of complex firewall configurations," in *Proceedings of the ninth international symposium on visualization for cyber security*. ACM, 2012, pp. 1–8.

[8] D. F. Jerding and J. T. Stasko, "The information mural: A technique for displaying and navigating large information spaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 3, pp. 257–271, 1998.

[9] J. Stasko, C. Görg, and Z. Liu, "Jigsaw: supporting investigative analysis through interactive visualization," *Information visualization*, vol. 7, no. 2, pp. 118–132, 2008.

[10] J. C. Roberts, "State of the art: Coordinated & multiple views in exploratory visualization," in *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*. IEEE, 2007, pp. 61–71.

[11] D. M. Eler, F. V. Paulovich, M. C. F. de Oliveira, and R. Minghim, "Coordinated and multiple views for visualizing text collections," in *Information Visualisation, 2008. IV'08. 12th International Conference*. IEEE, 2008, pp. 246–251.

[12] M. Jern, S. Palmberg, M. Ranlof, and A. Nilsson, "Coordinated views in dynamic interactive documents," in *Coordinated and Multiple Views in Exploratory Visualization, 2003. Proceedings. International Conference on*. IEEE, 2003, pp. 95–101.

[13] R. Gove, C. Dunne, B. Shneiderman, J. Klavans, and B. Dorr, "Evaluating visual and statistical exploration of scientific literature networks," in *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*. IEEE, 2011, pp. 217–224.

[14] M. J. McGuffin and J.-M. Robert, "Quantifying the space-efficiency of 2d graphical representations of trees," *Information Visualization*, vol. 9, no. 2, pp. 115–140, 2010.

[15] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Transactions on graphics (TOG)*, vol. 11, no. 1, pp. 92–99, 1992.

[16] B. S. Johnson, "Treemaps: visualizing hierarchical and categorical data," 1993.

[17] M. C. Chuah, "Dynamic aggregation with circular visual designs," in *Information Visualization, 1998. Proceedings. IEEE Symposium on*. IEEE, 1998, pp. 35–43.

[18] B. B. Bederson, "Photomesa: a zoomable image browser using quantum treemaps and bubblemaps," in *Proceedings of the 14th annual ACM symposium on User interface software and technology*. ACM, 2001, pp. 71–80.

[19] H. Lü and J. Fogarty, "Cascaded treemaps: examining the visibility and stability of structure in treemaps," in *Proceedings of graphics interface 2008*. Canadian Information Processing Society, 2008, pp. 259–266.

[20] B. Shneiderman and M. Wattenberg, "Ordered treemap layouts," in *Proceedings of the IEEE Symposium on Information Visualization 2001*, vol. 73078, 2001.

[21] H.-J. Schulz, S. Hadlak, and H. Schumann, "The design space of implicit hierarchy visualization: A survey," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 4, pp. 393–411, 2011.

[22] H.-J. Schulz, "Treevis. net: A tree visualization reference," *IEEE Computer Graphics and Applications*, vol. 31, no. 6, pp. 11–15, 2011.

[23] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone trees: animated 3d visualizations of hierarchical information," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1991, pp. 189–194.

[24] J. Lamping, R. Rao, and P. Pirolli, "A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 401–408.

[25] D. Oelke, D. Spretke, A. Stoffel, and D. A. Keim, "Visual readability analysis: How to make your writings easier to read," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 5, pp. 662–674, 2012.

[26] C. Collins, S. Carpendale, and G. Penn, "Docuburst: Visualizing document content using language structure," in *Computer graphics forum*, vol. 28, no. 3. Wiley Online Library, 2009, pp. 1039–1046.

[27] A. Di Iorio, S. Peroni, F. Poggi, and F. Vitali, "Dealing with structural patterns of xml documents," *Journal of the Association for Information Science and Technology*, vol. 65, no. 9, pp. 1884–1900, 2014.

[28] ——, "A first approach to the automatic recognition of structural patterns in xml documents," in *Proceedings of the 2012 ACM symposium on Document Engineering*. ACM, 2012, pp. 85–94.

[29] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Visual Languages, 1996. Proceedings., IEEE Symposium on*. IEEE, 1996, pp. 336–343.

[30] A. Di Iorio, A. G. Nuzzolese, and S. Peroni, "Identifying functions of citations with citalo," in *Extended Semantic Web Conference*. Springer, 2013, pp. 231–235.

[31] A. Di Iorio, R. Giannella, F. Poggi, S. Peroni, and F. Vitali, "Exploring scholarly papers through citations," in *Proceedings of the 2015 ACM Symposium on Document Engineering*, ser. DocEng '15. New York, NY, USA: ACM, 2015, pp. 107–116. [Online]. Available: http://doi.acm.org/10.1145/2682571.2797065

[32] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[33] J. Sauro, *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability LLC, 2011.

[34] J. Lewis and J. Sauro, "The factor structure of the system usability scale," *Human centered design*, pp. 94–103, 2009.

[35] J. Corbin, A. Strauss *et al.*, "Basics of qualitative research: Techniques and procedures for developing grounded theory," 2008.