Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

Interactive Treemaps With Detail on Demand to Support Information Search in Documents

Stefan Schlechtweg[1], Petra Schulze-Wollgast[2] and Heidrun Schumann [2]

[1] Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg, Universitätsplatz 2, D-39106 Magdeburg, Germany
[2] Institute of Computer Science, University of Rostock, Albert-Einstein-Straße 21, D-18055 Rostock, Germany

Abstract

This paper addresses the issue of how information visualization techniques can be used to assist full-text search in electronic documents. Our approach supports multiple term queries with interactive treemaps. We use a treemap to visualize the basic structure of the document and exploit color coding to show the distribution of query terms on various levels of the hierarchy. Furthermore, we include filtering techniques to concentrate on those parts of the structure that actually contain the requested information, and, finally provide interactive tools to give access to detailed information whenever the user wishes.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces (GUI) I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques

Keywords: information visualization, hierarchy visualization, treemaps, full-text search, filtering, detail-on-demand

1. Introduction

When working with electronic documents, full-text search is one of the most often performed tasks and an important advantage of electronic texts over their paper counterparts. The problem of full-text search can be more formally stated as follows. Given a text and a set of query terms, find all occurrences of the terms in the text and give the reader the possibility to locate the respective positions in the text. Connected herewith is the assessment whether the context of the located finding fulfills the information need of the reader and to which extend. While the first part of the problem – finding the occurrences of given terms in a text – has been solved, there are still open questions connected with the second part.

In our work we concentrate on searches in well structured texts. In contrast to almost unstructured texts, like novels, there are other requirements. Such structured texts (like textbooks, conference proceedings, scientific works, etc.) are usually not read cover-to cover but instead certain chapters are treated in an order determined by the reader's interest. To speed up full-text searches in this case, we need to tell the user, which structural units contain relevant information. Therefore, it is not sufficient to simply show these locations of query terms and let the user jump back and forth between them. The reader is more concerned with identifying larger structural units of the document that describe the topic characterized by the given term collection.

Given this intention, the following issues are important:

- quickly finding the "best" matches for the issued query,
- easily assessing to which extent a unit of text matches the query (i.e., which terms are contained how often in a unit of text) by connecting structural information of the text with the query result.

In this paper we propose to use Information Visualization techniques to incorporate the above mentioned issues in the process of full-text search. More specifically, we use treemaps to give the user access to structural information and enrich these treemaps with interactive techniques to relate the resulting location to the context on different structural levels of the document and to provide detail on demand.

The remainder of the paper is organized as follows. In Section 2 we give an overview of related material before we state our approach in Section 3. Starting with the main

problem. Section 4 shows how treemaps can be used to integrate structural information and full-text search results. In order to enhance the visualization, alternative presentations are introduced in Section 5. Further user support is given by providing detail-on-demand techniques in Section 6. We then conclude the paper and give some ideas for future work.

2. Related Work

We are looking for visualizations of term distribution in highly structured scientific texts. Therefore, the research presented in this paper is especially connected to two areas:

1. visualization of hierarchical structures and
2. visualization of query results in documents.

Several visualization methods exist that emphasize hierarchical relations (see Keim et al. for an overview [KMS02]). We distinguish between explicit methods, i.e., drawing a graph of the hierarchy with edges and nodes, and implicit space-filling methods, that show hierarchical relations by spatial arrangements of nodes.

For visualizing a document hierarchy, the child-parent relations between structural units can be regarded as containment relations, i.e., the document *contains* several chapters, one of these *contains* several sections, and so on. Therefore, implicit hierarchy visualization methods with a nested arrangement like treemaps [Shn92] are usable. Here, the containment relations can be deducted from the spatial arrangement of the elements. However, hierarchy visualization methods focus on presenting the structure itself, especially dealing with large structures consisting of a huge amount of nodes (e.g., as shown by Fékete and Plaisant [FP02]). Our main focus is on the visualization of certain complex features and attributes (regarding the query term distribution) in connection with such a structure visualization.

A standard way of visualizing the distribution of query terms in a document as it is exploited by various application programms (as, for instance, internet search engines) highlights the terms using different text attributes within a visualization of the document. The user may then browse through the text or use a "find next/previous" function to jump between the locations in a sequential manner. In some cases, this is sufficient if the reader is interested in finding all locations of a given (set of) term(s).

Besides emphasizing the query terms in a textual representation of the document, there are approaches that graphically visualize information about the frequency of occurrence of terms after a search in a document collection. These approaches first try to find documents that contain specific terms. Spence describes two general approaches for graphically presenting the results of that search [Spe01]: a map-like fashion as Themescapes, e.g., [WTP*95], or a point cloud version as Galaxies, e.g., [Ren94]. Both approaches are based on a special arrangement of the documents either on a map or in a multidimensional feature space (given by a set of keywords). Beside these general strategies several techniques were developed to address special problems. For example, changes in the frequency of terms (and, hence, the change of topic) over time in a large document collection can be visualized as ThemeRivers [HHWN02]. On the other hand, Tilebars [Hea95] were developed to show the term distribution in large text documents. The document is analyzed for topic changes which impose a "tiling" – a sequential partitioning – on the document. A special visualization technique is then used to show the distribution of query terms in this text. The user interface allows the specification of the query terms and the search in multiple documents.

In our case, however, the locations of query terms within a probably long single text have to be presented. A possible solution for this problem are so called Information Murals as introduced in [JS98]. An Information Mural is a two-dimensional reduced representation of the entire document that fits entirely in a display region. Jerding and Stasko have shown the use of such representations for visualizing the distribution of query terms, however, without further information about the context the user is not able to assess whether a location of a term fits his or her information needs.

3. Approach and Challenge

Our approach supports multiple term queries with interactive treemaps. We have chosen to use treemaps since they provide an optimal representation of the containment relations that form the document structure. The chapter structure can be extracted while parsing the text. We have further subdivided the text up to the level of paragraphs. Special parts, such as images, tables, equations, and inset text boxes are treated as paragraphs. As an example throughout this paper serves a textbook on computer graphics.

Our approach can be described as follows:

1. using a treemap to visualize the basic document structure,
2. exploiting color coding to show the distribution of query terms on various levels of the hierarchy,
3. including filtering techniques to concentrate on those parts of the structure that actually contain the requested information, and, finally,
4. providing interactive tools to give access to detailed information whenever the user wishes.

Treemaps have been introduced by Shneiderman [Shn92] and the use of treemaps for different tasks and problems is well-known. Treemaps are introduced as a 2D space-filling representation for arbitrary trees where child nodes are represented as rectangles that are contained within the rectangle representing the father node. Several extensions to the original treemap algorithm were published mainly addressing layout problems. Considering the original "slice-and-dice" treemap algorithm it presents an intuitive layout where the

structure of the hierarchy can be seen easily. Also, the ordering of the underlying data is preserved in the visualization. Bruls et al. present "Squarified Treemaps", an algorithm that generates layouts in which the rectangles approximate squares [BHvW00]. However, the ordering is no longer preserved. Shneiderman and Wattenberg present "Ordered treemap layouts" that adhere to the ordering in the underlying data even in the squarified form [SW01].

Our aim is to establish a highly interactive treemap that fulfills the "Information Seeking Mantra" *Overview first, zoom and filter, then details-on-demand* stated by Shneiderman [Shn96]. Hence, we allow different views with different levels of detail for information representation on the basis of a treemap. We use the "slice-and-dice" treemap layout since it provides a sufficient base for our work. Moreover, we integrate alternative presentation styles to highlight some regions (e.g. by using cushion treemaps [vWvdW99] for special nodes of interest), or to simplify other regions that do not contain relevant terms. Doing so, we provide a rich functionality for an intuitive exploration of structured documents and for a well directed search for information. There exist various possible application scenarios, as for instance an intuitive search interface for concept indexes, or for full-text queries of documents in a digital library. In the following we will outline our approach in more detail.

4. Combining Structure and Search Results – Overview

Given an electronic document containing a well structured text, the structure of the text can be extracted and forms a tree that is to be visualized. Using standard treemaps, the text structure becomes clearly visible. The nested box character of the document structure can even be emphasized by using a framed version of the standard treemaps. As can be seen in Figure 1, paragraphs of text (the leaf nodes of the hierarchy) are displayed as rectangles and all inner nodes (chapters, sections, etc.) frame these so that the nesting becomes clear. The use of treemaps yields the possibility to encode various other information at the same time as, for example, a characterization of the contents of a node as can be seen in Figure 2. Here, each leaf node is characterized as to whether it is a paragraph of text (orange), an image (magenta), an equation (yellow), a list (blue for bullet lists and red for numbered lists), a table (green), or an algorithm (purple).

There are basically two possibilities to include additional attributes in a treemap visualization: changing the size or the color of the rectangles according to a given attribute value. In the above examples and in the following, we compute a weighting function for each node within the hierarchy that then determines the size or color of the rectangle to be drawn for each node. To visualize the structure of the document alone this function should be designed in such a way that larger parts of the document also cover larger rectangles in the treemap so that the distribution of the material within the

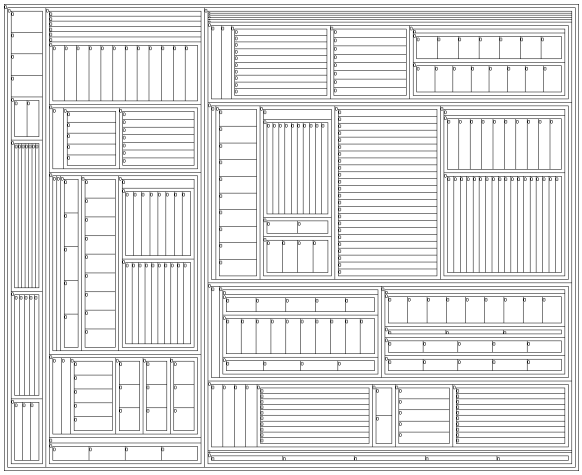


Figure 1: *Using treemaps to visualize the structure of the document without showing any additional information.*

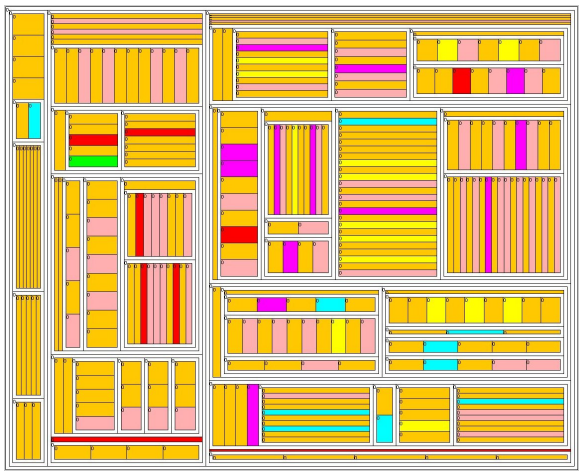


Figure 2: *Color coding the type of leaf nodes in addition to a structure visualization.*

document becomes clear. The most simple solution counts the number of leaves in a subtree and uses this measure as weight for the root node of a subtree. We compute such a measure – $\omega_{\text{leaf}}(n)$ – recursively bottom-up starting at the leaf nodes:

$$\omega_{\text{leaf}}(n) = \begin{cases} 1 & n \text{ is leaf node} \\ \sum_{i=1}^{m} \omega_{\text{leaf}}(n_i) & n \text{ is inner node} \end{cases}$$

In the above equation n_i are the child nodes to node n. Given A_n as the area available to draw an inner node n, i.e., to draw the subtree rooted at n, the area covered by the i-th child of n, denoted as n_i, is then computed as

$$A_{n_i} = \frac{\omega_{\text{leaf}}(n_i)}{\sum_{j=1}^{m} \omega_{\text{leaf}}(n_j)} A_n$$

The area of node n is subdivided according to these fractions. Since we are using the standard treemap algorithm, the areas are alternately subdivided horizontally and vertically. Setting $\omega_{\text{leaf}}(n) = 1$ for leaf nodes yields the same size for all sibling leaves. Figures 1 to 3 are generated using this approach. There are other possible weighting functions that support the visualization of the document structure, for example, using the length of the text forming one paragraph or similar measures. Also, Strahler numbers [**?**] could be used as weights since these values give a quantitative information about the complexity of a (sub)tree (see also Herman et al. [**?**]). In the following, we use ω_{leaf} as basis for further derived weighting functions with the goal to present not only the structure but also the results of a full-text query, i.e., to visualize the frequency of occurrence of the query terms.

Performing a query with a single query term results in assigning an integer value – the number of occurrences of the query term – to each leaf node of the tree (the paragraphs) and summing up the values of the child nodes for each inner node. The root node finally contains the sum of the occurrences for the complete document. Therefore, we introduce a new weighting function $\omega_{\text{occ}}(n)$ as follows:

$$\omega_{\text{occ}}(n) = \begin{cases} 1 + t(n) & n \text{ is leaf node} \\ \sum_{i=1}^{m} \omega_{\text{occ}}(n_i) & n \text{ is inner node} \end{cases}$$

with $t(n)$ being the number of occurrences of a query term in a paragraph (leaf) node. This function combines the structure of the tree (counting the number of leaf nodes) and the number of occurrences of the query term. We can map these integers onto the color of the treemap regions as in Figure 3. Here, we have chosen to use the same color (hue) for each node and only change the brightness, so that the color gets darker with an increasing number of occurrence of the query term, i.e., the brightness for each node is a function $b_n = f(\omega_{\text{occ}}(n))$.

It becomes apparent that color coding enhances the recognition of the search result on higher levels of the hierarchy since here the number of occurrence is significantly higher resulting in a significant darkening. Up to now, the *size* of the treemap regions only represent the document structure as it is introduced with the weighting function $\omega_{\text{leaf}}(n)$. We can emphasize the recognition of query term locations by also using $\omega_{\text{occ}}(n)$ to determine the size of the treemap regions. Hence, by combining both size and color coding of query results, we achieve an interesting overview of the term distribution compared to the document structure as shown in Figure 4. The process of changing size and color is visible to the user so that he or she gets another indicator of where search results are found.

Full-text search operations are usually performed with more than one query term. This yields the need to (1) find a way to merge the search results for all terms in one integer value that is then visualized and (2) find a way to inform the user which term distribution exists at a specific location.

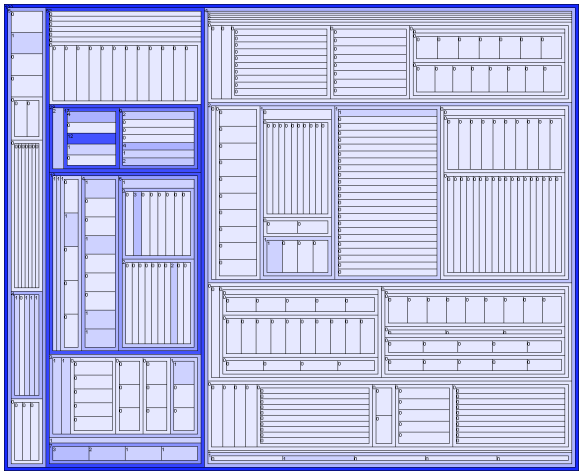


Figure 3: *Encoding the number of occurrences of a single query term in the color of the rectangles.*

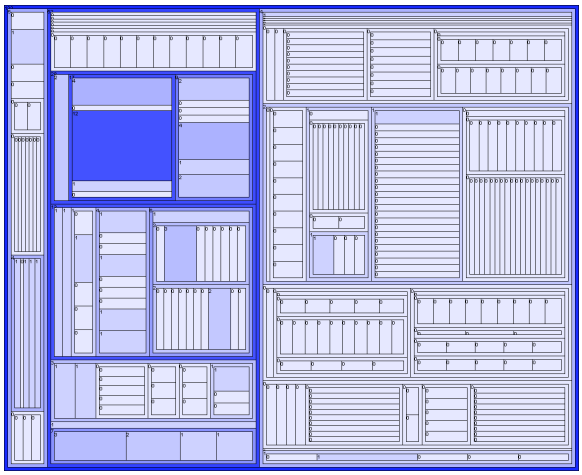


Figure 4: *Combining size and color coding in one view results in an overview of the term distribution related to the document's structure.*

The most simple solution to the first part of the problem is to sum up the values for the query terms and present this sum as the overall weight for a paragraph (or higher level node). This is sufficient in most cases since the user is most probably interested in places where any of the query term is found, i.e., the query terms are combined using a logical "OR". Given the document and a set of k query terms that occur $t_1(n), \ldots, t_k(n)$ times in a paragraph, we derive the weighting function $\omega_{\text{comb}}(n)$ as

$$\omega_{\text{comb}}(n) = \begin{cases} 1 + \sum_{j=1}^{k} t_j(n) & n \text{ is leaf node} \\ \sum_{i=1}^{m} \omega_{\text{comb}}(n_i) & n \text{ is inner node} \end{cases}$$

Here, the combination with the document structure (as it was

already done for ω_{occ}) becomes especially valuable. Finding a section or chapter containing the query terms better satisfies the information need at hand than finding just a series of sentences. In Section 6 we will show how to solve the problem of visualizing the term distribution.

5. Alternative Presentations – Filtering

As the examples have already shown, the query results are easily recognizable. However, if the document structure becomes larger or more complicated, the presented view might contain too much information. We need filtering techniques to even further emphasize where query terms are found and to deemphasize the less relevant parts. Also here, the user directly sees the process of the changes so that an additional indication is given where the filtering occurs.

In the following we built upon the result that a combination of size and color coding the query results gives the best view of the data. First, we can omit drawing the internal structure for those nodes where no search terms have been found but keeping the size of the rectangles as if the substructure was present (cf. Figure 5). Here, the size of the rectangles is computed as above using $\omega_{comb}(n)$ as weighting function to reflect the number of occurrences of all query terms. However, when drawing the treemap, we stop subdividing a rectangle if there are no occurrences in the subtree rootet at the respective node n, i.e., if $\omega_{comb}(n) = \omega_{leaf}(n)$. The problem here is that the size of a rectangle in this case represents the size of the respective subtree while on the other hand the size of a rectangle for a non-empty node represents also the number of occurrences of query terms. This might be confusing and misleading.

To prevent this, such nodes where a substructure is hidden are also treated as leaf nodes when drawing the treemap resulting in a size of the rectangle that is similar to a paragraph node at the same level of the hierarchy. An example is given in Figure 6. For leaf nodes we compute the weight as described in Section 4 while for inner nodes the computation is changed as follows:

$$\omega_{filt}(n) = \begin{cases} 1 + \sum_{j=1}^{k} t_j(n) & n \text{ is leaf node} \\ 1 & \text{if } \sum_{i=1}^{m} \omega_{filt}(n_i) - \omega_{leaf}(n) = 0 \\ \sum_{i=1}^{m} \omega_{filt}(n_i) & \text{otherwise} \end{cases}$$

This yields a size as for an empty leaf node at the respective hierarchy level if no search term was found.

Still, there is another source of misunderstanding. Two very different kinds of nodes are now displayed in one view without visually distinguishing them: *real* leaf nodes that do not contain the query term and roots of subtrees that do not contain query terms and hide a complete substructure. However, nodes not containing query terms are irrelevant for the current search so that it is more important to emphasize these parts of the structure that do actually match the query. We therefore combine cushion treemaps [vWvdW99] with regular treemaps. All those nodes that represent findings of query terms are displayed as cushion treemap nodes while the rest remains flat. In addition, the label of such rectangles that hide substructures with no query terms is changed. The result is shown in Figure 7.

Figure 5: *Omitting the structure but keeping the size as that of the whole subtree*

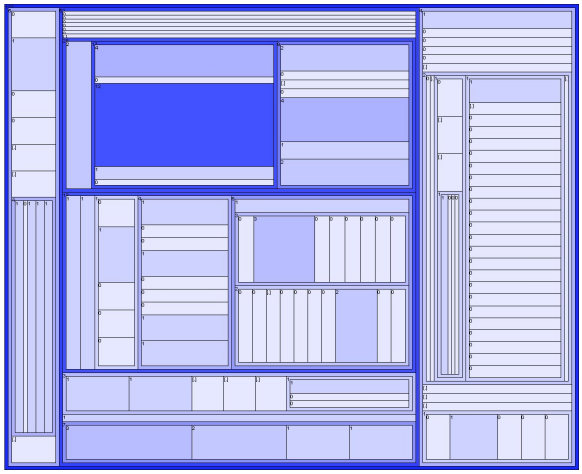


Figure 6: *Omitting the structure and adjusting the size*

The resulting visualizations simplify the structure by not showing irrelevant parts (i.e., hiding all subtrees that do not contain any query terms) but still make the structure visible. If at least one query term was found, then the second level of the hierarchy (in our case, the chapters of the document) is always displayed since the root node of the complete document is not empty. Also, the areas of the document that match the query are clearly recognizable due to the alternative presentation. In order to even more raise the effectivity of the visualization, additional information should be given to the viewer on request.

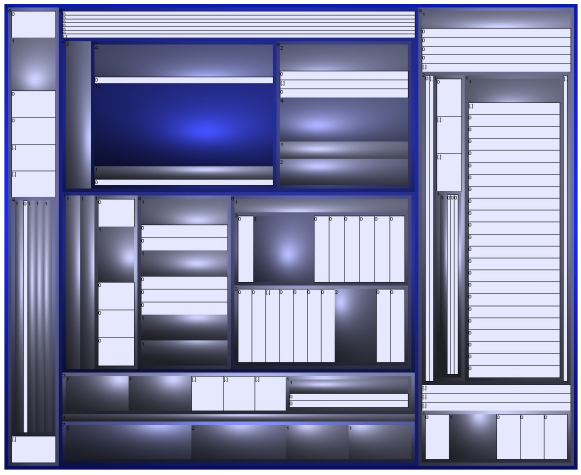
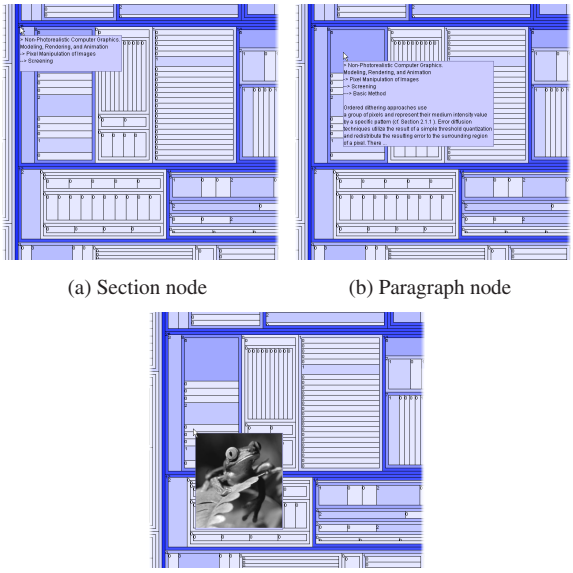


Figure 7: *Using cushion treemaps in areas where query terms have been found and leaving all other areas flat.*



(a) Section node (b) Paragraph node

(c) Image node

Figure 8: *Integrating context information as tooltips. The images show just a part of the treemap*

6. Interactive Tools – Details on Demand

While we have concentrated so far on integrating a visualization of query results and a visualization of the document structure, we still need to provide tools to assess the context of a location. Again, the document structure gives information of how well a larger unit of text (section or chapter) might fit. We provide local and more global information via tooltips as one technique to provide detail on demand. An example that uses such techniques in another application domain is the "Map of the Market" on `www.smartmomey.com` where tooltips are used to present specific data and a selection of further information sources.

In our scenario, moving the mouse in an area of the treemap causes a box to appear containing information that depends on the type of the node under the mouse. Grouping nodes, such as sections and chapters, are well characterized by their title and their relation to higher order nodes. So we show all section titles that are found on the direct path from the root to the respective node (see Figure 8(a)). Paragraph nodes (leaf nodes) usually do not have a title so that here either the first part of the paragraph can be displayed or, preferrably, the part of the paragraph that contains the query terms. Here, also, the position of the paragraph in the document hierarchy is shown by the titles of all higher order nodes (Figure 8(b)). Finally, if other document parts – such as images – are also contained in the hierarchy, they can be displayed and help to estimate the context (Figure 8(c)).

Such tooltips can also be used to visualize the distribution of query terms from a multi term query. When presenting the query results simply by giving the overall sum of matches, there is no way of assessing whether the respective location fits the user's information needs or not. Imagine a set of query terms (a, b, c) being found with the following quantities. Paragraph A contains 20 findings with the distribution (6, 7, 7) while paragraph B also contains 20 findings but this time distributed as (0, 20, 0). Visualizing the number 20 alone (as it is done with the techniques described in Section 4) gives no chance of distinguishing between these two situations. Therefore, the user has no possibility to reject paragraph B and concentrate on paragraph A. We have decided, nevertheless, to use the above mentioned technique to visualize the overall numbers in a treemap and include an additional visualization that shows the term distribution when interacting. We can show a histogram of the term distribution as tooltip when moving the pointing device over the treemap. This gives immediate access to all needed data but prevents, on the other hand, the display of the information about the context of a term location. Giving priority to the latter kind of information, we therefore show the histogram alongside the treemap in a different area of the query interface as can be seen in Figure 9.

Finally, the search interface is tightly coupled with a standard presentation of the underlying text, for example, in a browser or a special reader. Selecting a node in the treemap moves the selection to the specified location in the text. This functionality is comparable to the user interface used for TileBars [Hea95] where mouse interaction within the TileBar visualization leads to a selection of the text in the document. While TileBars work on a sequential text partitioning, our approach also supports navigation based on hierarchy information.

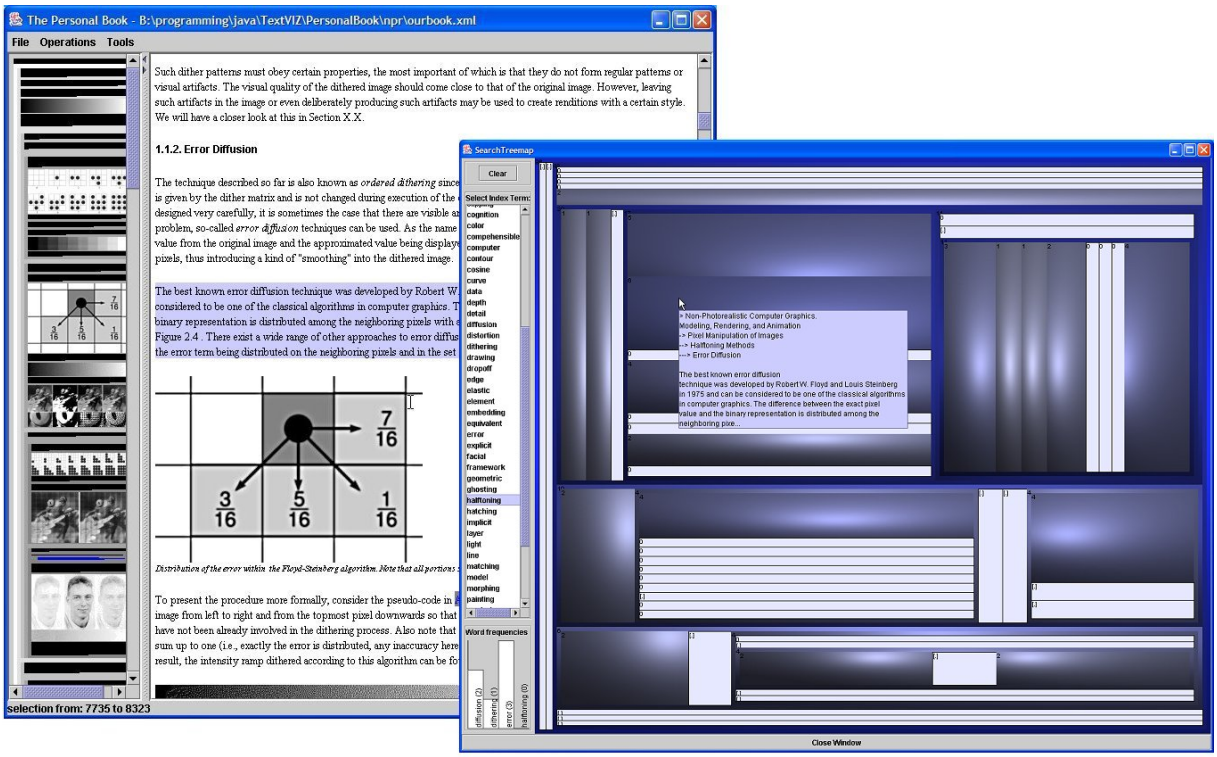


Figure 9: *The complete search interface. In the right window, the query terms are selected from the list and the treemap is built. The term distribution in the currently pointed at node is shown in the histogram in the bottom left corner of that window. Finally, the user selected this specific paragraph and the document browser (left window) shows the respective position.*

Putting it all together, the process of full-text search in a document now consists of two steps. After issuing the query, a treemap visualization of the document is built that helps to find the part of the document that best fits the user's information needs. From the treemap he or she can then directly jump to the desired location in the text (cf. Figure 9).

7. Conclusion

In this paper we have addressed the problem of full-text search in documents. The problem of assessing to which extend a finding matches the intention of the query needs a close connection of the search itself and the placement of the location in the document structure. Given a query built from one or multiple query terms, we build a treemap based visualization of the document structure and integrate quantitative information about the query term distribution by encoding the number of occurrences via color and size of the treemap regions. This yields an overview visualization to help locating interesting parts of the document. To further enhance this visualization filtering techniques hide substructures that do not match the query. We use alternative graphical representations to communicate the result of this filtering to the user. Finally, for an assessment of the local surrounding of a location, interactive tools are provided to give context details on demand. The visualization is closely coupled with a standard display of the document so that it can be used for navigation.

Further questions that arise from this research are mainly connected with the alternative presentations introduced in connection with the filtering techniques. Since the transition between the areas displayed as cushion treemaps and using standard treemaps is a rather abrupt one, we are searching for other techniques to deemphasize irrelevant parts of the structure. Semantic Depth of Field [KMH01] as one possible solution is only adequate if many adjacent regions are to be blurred since otherwise there are only uniquely colored rectangles and the effect can be neglected. One approach would be to use different textures on the treemap regions or to include schematic representations, such as TileBars or Information Murals, of the respective paragraph directly in the treemap.

Another question arises when the documents get larger. Then, the treemap regions become too small to be recognizable. Therefore, adaptive techniques seem to be promising where the detail of the structure visualization is set according to the number of occurrences of query terms. Since it is important to provide local detail in the surrounding

of query term locations, simply visualizing the document structure only up to a certain level is insufficient. Here overview + detail techniques should prove useful. Adding a dimension to the visualization and use 3D layouts like BeamTrees [vHvW02] or Information Cubes [RG93] can be another possibility to handle large documents.

Finally, an even closer coupling between the search visualization and the rendered text would further enhance the usability. One example would be to present the paragraph text using text attributes and images in the tooltips since humans tend to remember such graphical information better than the actual contents of a paragraph. In general, a user study should compare our approach with standard tools known from WWW browsers or eBooks. We hypothesize that the task completion time for finding a specific section in a document based on a roughly stated idea could be lower using our approach than with standard tools. Therefore, such tools as presented here provide even more added value for electronic documents.

References

[BHvW00] BRULS M., HUIZING K., VAN WIJK J.: Squarified Treemaps. In *Proc. TCVG 2000, the Joint Eurographics and IEEE TCVG Symposium on Visualization* (2000), IEEE Computer Society, pp. 33–42. 3

[FP02] FEKETE J.-D., PLAISANT C.: Interactive Information Visualization of a Million Items. In *Proc. InfoVis 2002* (2002), IEEE Computer Science Press, pp. 117–127. 2

[Hea95] HEARST M. A.: TileBars: Visualization of Term Distribution Information in Full Text Information Access. In *Proc. CHI'95* (1995), ACM SIGCHI, pp. 59–66. 2, 6

[HHWN02] HAVRE S., HETZLER E., WHITNEY P., NOWELL L.: ThemeRiver: Visualizing Thematic Changes in Large Document Collections. *IEEE Trans. on Visualization and Computer Graphics 8*, 1 (Jan. 2002), 9–20. 2

[JS98] JERDING D. F., STASKO J. T.: The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Trans. on Visualization and Computer Graphics 4*, 3 (Sept. 1998), 257–271. 2

[KMH01] KOSARA R., MIKSCH S., HAUSER H.: Semantic Depth of Field. In *Proc. InfoVis 2001* (2001), IEEE Computer Society, pp. 97–104. 7

[KMS02] KEIM D., MÜLLER W., SCHUMANN H.: Visual Data Mining. In *State of the Art Reports – Eurographics'2002* (2002), Eurographics Association. 2

[Ren94] RENNISON E.: Galaxy of News. An Approach to Visualising and Understanding Expansive News Landscapes. In *Proc. UIST'94* (1994), ACM Press, pp. 3–12. 2

[RG93] REKIMOTO J., GREEN M.: The Information Cube: Using Transparency in 3D Information Visualization. In *Proc. Workshop on Information Technologies & Systems (WITS'93)* (1993). 8

[Shn92] SHNEIDERMAN B.: Tree Visualization With Treemaps: a 2-d Space-Filling Approach. *ACM Transactions on Graphics 11*, 1 (1992), 92–99. 2

[Shn96] SHNEIDERMAN B.: The Eyes Have it: A Task by Data Type Taxonomy for Information Visualization. In *Proc. VL'96 Symposium on Visual Languages* (1996), IEEE Press, pp. 336–343. 3

[Spe01] SPENCE R.: *Information Visualization*. ACM Press, New York, 2001. 2

[SW01] SHNEIDERMAN B., WATTENBERG M.: Ordered Treemap Layouts. In *Proc. InfoVis 2001* (2001), IEEE Computer Society, pp. 73–79. 3

[vHvW02] VAN HAM F., VAN WIJK J. J.: Beamtrees: Compact Visualization of Large Hierarchies. In *Proc. InfoVis 2002* (2002), IEEE Computer Science Press, pp. 93–100. 8

[vWvdW99] VAN WIJK J. J., VAN DE WETERING H.: Cushion Treemaps: Visualization of Hierarchical Information. In *Proc. InfoVis'99* (1999), IEEE Computer Society, pp. 73–78. 3, 5

[WTP*95] WISE J., THOMA J., PENNOCK K., LANTRIP D., POTTIER M., SCHUR A., CROW V.: Visualising the Non-Visual: Spatial Analysis and Interaction With Information from Text Documents. In *Proc. InfoVis'95* (1995), IEEE Press, pp. 51–58. 2