Text Snippets from the DomGraph

Jacob Ratkiewicz[1,2]
jpr@cs.indiana.edu

Filippo Menczer[1,2]
fil@indiana.edu

[1]Complex Networks Lagrange Laboratory, Institute for Scientific Interchange Foundation, Torino, Italy
[2]Dept. of Computer Science, Indiana University, Bloomington, Indiana, USA

ABSTRACT

We explore the idea that the Document-Object Model tree of an HTML page — absent any semantic or heuristic interpretations of the tags and their positions — provides cues about the importance of the information it contains. This hypothesis is evaluated by constructing a DomGraph, i.e., a network of the DOM trees of pages connected by their hyperlinks, and using it in a snippet-extraction technique. In this process, we also address technical issues related to the processing of the resultant very large graph. Snippets produced by this technique are compared in a user study to those extracted by a reasonable and simple baseline method, and found to be clearly preferred by users.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Information networks, Performance evaluation (effectiveness), Question-answering (fact retrieval) systems*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*User issues*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*User issues*

General Terms

Algorithms, Design, Experimentation, Human Factors

Keywords

Document object model, graph topology, DomGraph, snippet extraction.

1. INTRODUCTION

Ranking a chunk of information by its usefulness in answering a query is a critical component of an information retrieval system. Various definitions of what a "chunk" is, and methods for ranking these chunks, have been proposed for use on the Web. The most widely-used concept of a unit of information on the Web is that of a single Web page, and the leading ranking algorithm for determining the prestige of these pages is PageRank [3], usually augmented by commercial search engines with proprietary improvements.

Users often interact with the results of a text information retrieval system by viewing a list of "snippets" of text from each retrieved document, in order to judge each document's usefulness in answering their query without actually visiting it. These snippets are classically derived from the text of the document, ignoring HTML markup if it is present. Snippet extraction algorithms use natural-language processing cues in an attempt to find coherent bits of text that contain the query terms and give the user some contextual information about the page when seen in isolation.

This paper attempts to unify the above two approaches. Rather than using PageRank at the page level to rank entire pages, we explore its use at the level of each page's Document-Object Model (DOM) tree, driving the ranking of relevant snippets of text from the page. In doing so, we explore the hypothesis that the structure of a page's DOM tree might lend cues about the relative importance of the different bits of information it contains — even absent any heuristic or semantic information about the HTML tags composing the tree.

The main contributions of this paper are as follows:

- We describe the construction of the DomGraph, a hybrid graph comprised of the page link graph in conjunction with the DOM trees of individual pages.
- We discuss the technical difficulties in processing the large graph that results, in the process creating Webgraph++, a more scalable translation of the Webgraph library [2] to C++.
- We evaluate by a user study the use of this graph for selecting query-biased snippets from pages.

2. RELATED WORK

Many current algorithms for computing the prestige of nodes in a Web graph are based on PageRank [3], which assigns prestige in a manner proportional to indegree to a first approximation [7]. Ranking of Web objects at a level higher than that of pages has also been explored, such as in HostGraph [1] or SiteRank [11, 12]. One of this paper's contributions is the novel idea of ranking Web objects at a gran-

SIGIR 2008 Workshop on Focused Retrieval July 24, 2008, Singapore

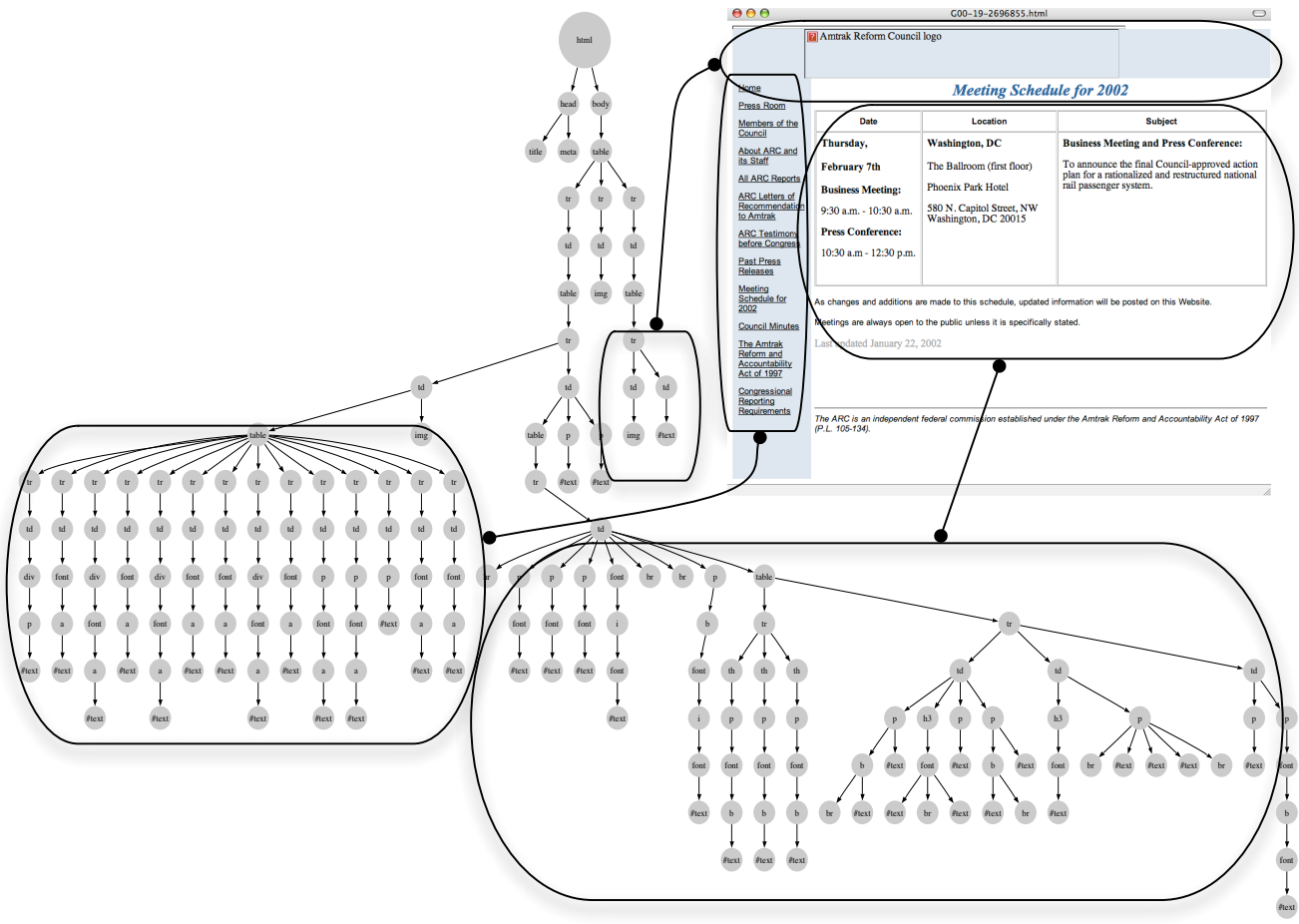

Figure 1: A sample page from the .GOV dataset, along with its DOM representation. Page elements such as tables and navigation bars can greatly increase the complexity of a page's DOM without affecting its apparent length. Corresponding areas of the page and the DOM representation are highlighted.

ularity lower than that of a page, namely individual DOM elements.

The Document-Object Model view of HTML was codified by a W3C standard,[1] and is widely used. Use of the DOM for data extraction has been discussed by several previous works. One of the contexts in which it has been explored is that of accessibility issues. Gupta *et al.* [9] outline a method for condensing a page to its most essential elements to facilitate viewing of the page on mobile devices, or interpretation of the page by accessibility software such as screen readers. Their method is based on the DOM and involves a number of heuristic rules for selecting the important parts of the page, based on its genre. Buyukkokten *et al.* [4] describe "accordion summarization," a similar technique which uses the DOM in an attempt to identify standalone "semantic textual units" of the page which can be viewed as summaries of parts of the page. The user can then drill down to the part of the page which most interests her. The use of the DOM to guide computerized data mining efforts is explored by Can *et al.* [5], who use DOM cues and heuristics to identify and extract postal addresses from HTML pages. The use of the DOM for topical crawlers has been proposed by Pant and Menczer [13] to examine the textual context of a link and evaluate whether to follow it.

In the domain of segmenting unstructured text using only endogenous cues, relevant works include LexRank [6] and TextTiling [10]. In the former, the authors propose a method for determining the prestige (or *salience*) of sentences in a document by a method which involves computing similarities between pairs of sentences, building a network of these pairs in which two sentences are connected if their similarity is greater than a given threshold, and then running a modified PageRank on this network. This method is proposed to drive an extractive summarization algorithm. TextTiling is a technique for breaking a long stream of text (a *discourse*) into a series of smaller *topics*. The method works by computing similarities between adjacent "blocks" (3-5 sentence chunks) of text, and setting topic boundaries in places where the similarity value is relatively low.

[1] `http://www.w3.org/DOM/`

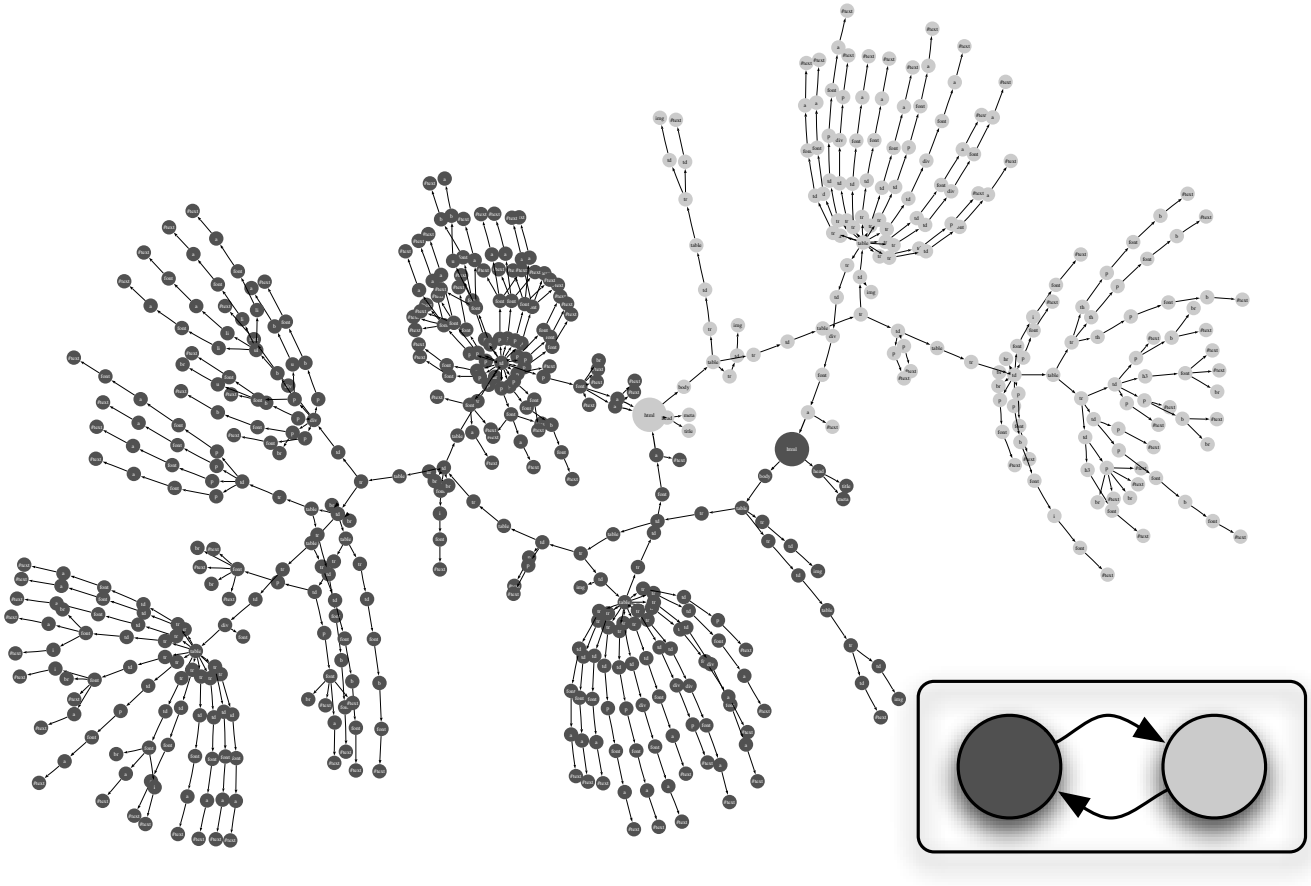

Figure 2: A subset of the DomGraph containing nodes belonging to two neighboring pages, one of them the page in Figure 1. For ease of visual identification, the root (<html> tag) of each DOM is displayed as a larger circle. In the Page graph (inset), the nodes representing each of these pages link to each other; however, in the DomGraph, distinct anchor tags each contribute their inlink to the target page's <html> tag.

Turpin *et al.* [14] outline a method for fast snippet generation in the search engine context, but focus more on the speed of the production than the quality of what is produced. A recent patent by Google, Inc. [8] describes a method for producing snippets, but no claim is made that this method is actually what is used by Google, and their true method is likely more complex.

3. THE DOM GRAPH

The DOM is in essence a way of viewing an HTML page as a tree — a fully-connected, directed acyclic graph with the node representing the `<html>` tag at the root. This simplifies access to the contents of the page by programming languages such as JavaScript, which can modify the DOM on the client side in order to implement interactive Web pages. Figure 1 illustrates a sample page and its DOM representation.

We wish to use this graph representation of each page as a stand-in for that page in the "standard" link graph of all pages. That is, we build a graph in which the nodes are elements of each page's DOM tree, and the links are induced both by parent-child connections in the DOM, and by hyperlinks between pages. Hyperlinks are treated as edges between the tag defining the hyperlink and the root of the target page's DOM tree. In the case of anchored hyperlinks (i.e. links of the form `<a href="a.com#anchor">...`), we connect the tag defining the hyperlink and the parent tag of the anchor in the target document. This expansion of a page into its DOM nodes typically increases the size of the graph by two orders of magnitude. Figure 2 illustrates a subset of the DomGraph. The two larger nodes in this graph are the `<html>` tags of the pages represented; the smaller nodes are nodes representing DOM elements created by the expansion.

Data is drawn from the `.gov` corpus.[2] This corpus contains about a million documents comprising approximately 20 GB of text. Not all of the documents are HTML; plain text documents and documents in other formats are also included. From this collection we extracted two graphs. The first is a "standard" graph, which we call S, in which the nodes are pages and the links between nodes are induced by page hyperlinks. The second is the DomGraph, D, which is described above. This latter graph is comprised of around 220 million nodes connected by around 230 million edges.

[2] `http://es.csiro.au/TRECWeb/govinfo.html`

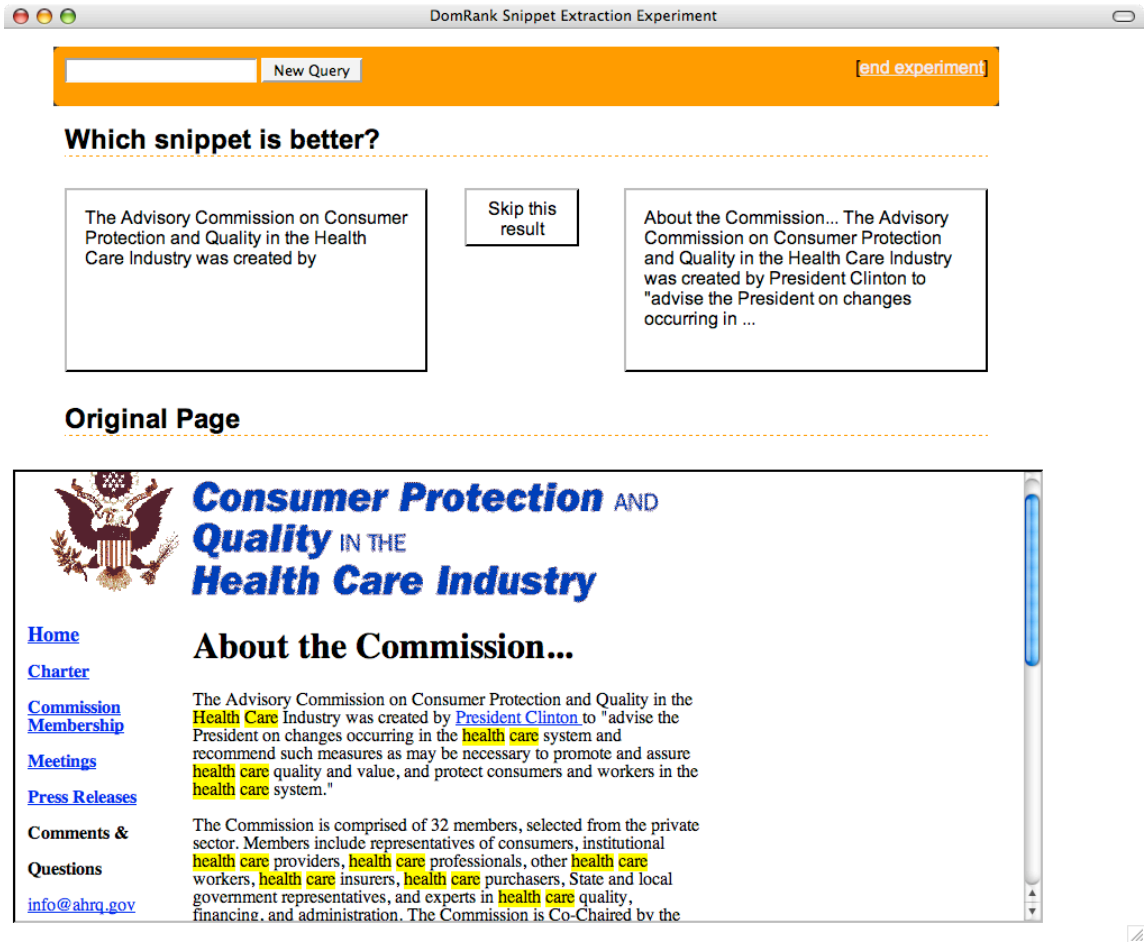

Figure 3: A screenshot of the user study experiment in action, with the query "health care." The candidate snippets are shown on the left and right, with the original page beneath. The position of the DOM snippet and baseline snippet was randomized at each instance. If at any time the user wished to enter a new query he or she could do so with the form at the top of the screen.

A graph of such size was a challenge to analyze with the computing hardware we had available. The library, Webgraph [2], that we chose to perform the initial analysis was written in Java. This language limits the sizes of certain data structures in a way which made our analysis impossible. To surmount this difficulty we undertook to translate this library into C++, and the resulting Webgraph++ library is available online.[3]

In order to leverage the S and D graphs for information retrieval, we compute the PageRank of the nodes in each. To distinguish the PageRank computed on the page graph from the results of that algorithm run on the DomGraph, we refer to the latter as DomRank. Further, we built two text indices using the Lucene[4] index and search platform. The first, for the standard graph, indexed the raw text of each page (i.e. with HTML markup stripped out). The second indexed the text of all textual DOM nodes. We refer to the first of these indices as the "page index," and the second as the "DOM index."

[3] `http://homer.informatics.indiana.edu/~nan/webgraph/`
[4] `http://lucene.apache.org/`

4. EXTRACTING TEXT SNIPPETS

The goal of our experiment was to measure the effectiveness of DOM cues for segmenting text in a meaningful way. To this end, we evaluated snippets extracted from the DOM against snippets extracted by a baseline method. We describe each of these methods in turn. Each method begins with a 'bag' of query terms, with stopwords removed, and an integer n which signifies which page in the result list should be considered for snippet extraction. It then performs a query against the page index to arrive at a list of page identifiers. The methods differ in what they do with these identifiers, as follows.

Baseline method. As a baseline method, we attempted to mimic the behavior of commercial search engines. This method takes the n-th page identifier from the list derived as above, and retrieves the plain text of the associated page

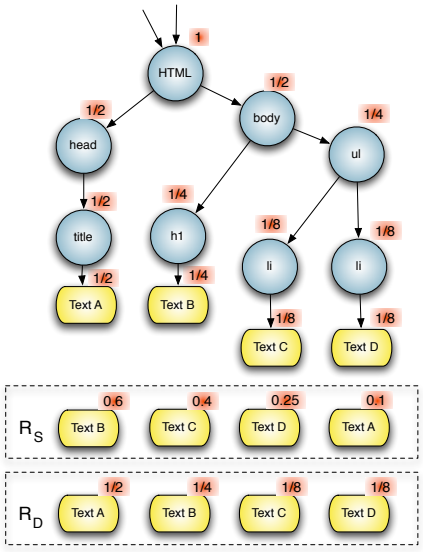

Figure 4: The DOM-based method for snippet extraction. Given a fixed page, the method extracts all text nodes and computes two rankings — one based on each node's similarity to the query as determined by Lucene, called R_s, and another by its DomRank, R_d. The method then returns the node that has the highest aggregate score, determined by $\alpha R_s + (1-\alpha)R_d$ for a constant α. This illustration assumes an incoming DomRank flow of 1.0 to the `html` node, and neglects the effect of the damping factor. Query similarity values are hypothetical. The winning node in this case would be `Text B` — it is ranked 4 (highest) in R_s, and 3 in R_d; thus in its case, $\alpha R_s + (1-\alpha)R_d = 0.7 \cdot 4 + 0.3 \cdot 3 = 3.7$.

(with HTML markup removed). It segments the text into sentences using the OpenNLP software package, trained on a corpus of text from the Wall Street Journal.[5] The method then attempts to return one of the following, in decreasing order of preference:

1. A single sentence containing all of the query terms.
2. A pair of sentences which together contain all the query terms
3. A "window" of text, possibly containing several sentences, which contains as many query terms as possible.

The size of the "window" in (3) is fixed at 30 words; further, if a snippet picked by (1) or (2) above is longer than 30 words, it is trimmed around the query terms with preference given to removing words at the end of the sentence. In our best assessment, this method seems to produce snippets similar to those returned by commercial search engines.

[5]Software package and trained model: `http://opennlp.sourceforge.net/`

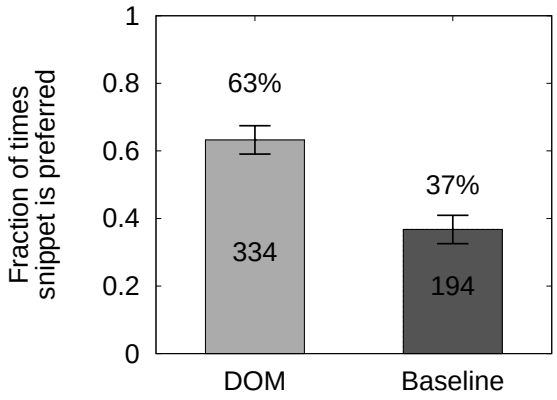

Figure 5: Results of the user study. The error bars are for a 95% confidence interval. The difference is statistically significant with $p < 0.01$ from a two-tailed t-test.

DOM method. The DOM method takes the n-th page identifier from the list just as the baseline method does; however, it derives its snippet in a different way. It performs a second query against the index of DOM nodes, limiting its search to the nodes which comprise the page used by the baseline method. This yields a list of text nodes ranked by their similarity to the query. It then creates a new list in which the rank of each item is determined by a linear combination of the each item's rank in the text similarity list and its rank in a list ordered by DomRank. Thus, if R_s is the node's rank when ordered by similarity and R_d is its rank by DomRank, its rank in the result list is determined by a sort on the quantity

$$\alpha R_s + (1-\alpha)R_d.$$

The value $\alpha = 0.7$ was chosen empirically. The DOM text at the head of the resulting list is what is returned as a snippet. If it is longer than 30 words, the 30-word-long window which contains as many query terms as possible is what is returned. Figure 4 illustrates this process.

5. EVALUATION

To evaluate the use of the DomGraph and DomRank for extracting important page cues, we performed a Web-based user study[6] comparing the two snippet-extraction algorithms. Users were asked to enter a query of their choice. They were then shown a pair of snippets, derived as above with $n = 1$ initially, and asked to click the one they thought was more helpful, or skip the result if neither was the clear winner. When the user rated either of the two snippets or clicked "skip," he or she was shown a new pair of snippets with $n \leftarrow n + 1$ (i.e. from the next page in the result set). Pages for which both methods produced the same snippet were skipped. At any point the user wished, he or she could enter a new query or end the experiment. A screenshot of the experiment in action is shown in Figure 3. The average length of a snippet returned from the baseline method was 27 ± 3 words, or 149 ± 10 characters; the DOM-based

[6]Indiana University IRB #07-11684

method's snippets were significantly shorter, at 12 ± 2 words or 63 ± 9 characters.

The experiment was active for approximately 1 week, and garnered around 40 participants who rated 528 snippets all together. The number of times a snippet created by each method was chosen over the other is shown in Figure 5.

Thus, the snippets extracted from the DOM were preferred approximately 50% more than those extracted by the baseline method. We are encouraged by these results, pointing to the DomGraph as a useful tool in Web applications.

6. DISCUSSION AND FUTURE WORK

The method presented here for extracting snippets directly from the DOM tree of Web documents does not rely on any heuristics or semantic information about the text, but solely on graph topological information. Our experimental evaluation shows it to outperform a simple and reasonable method that relies on textual cues alone. We do not present this as an argument that DOM-based snippet extraction would clearly outperform a state-of-the-art method, but rather to support the conclusion that the topology of the DOM contains semantic information that should be exploited for information retrieval.

Future work should deal with optimizing the computation of DomRank for the large graph which results from expansion of each page's DOM, as well as the size of the index involved. Among the techniques that might be explored are trimming each page's DOM to remove irrelevant nodes (i.e. nodes that do not affect the DomRank of any text nodes), and the possibility that the DomRank may be computed in some optimized way from the PageRank of the corresponding page graph.

An obvious candidate for application of techniques of this type is Web page summarization. This has been previously explored [4], but our method is unique in that it does not rely on page- or page-genre specific heuristics. Question answering applications could also benefit from this method's ability to select important bits of text from a list of candidates, based on the structure of the Web page from which the snippets are derived. More in general, applications such as clustering and classification could be improved by this technique's ability to select important items (and thus remove noise) from Web page data. DomRank even applies to non-textual data appearing in HTML pages, such as images and other media.

7. ACKNOWLEDGMENTS

The authors wish to thank the members of the NaN group at Indiana University for helpful discussions. They would also like to thank the participants in the user study, including the WEBIR and SIGIR mailing list members. This work is supported in part by the Indiana University School of Informatics in Bloomington, Indiana, and the Institute for Scientific Interchange Foundation in Turin, Italy.

8. REFERENCES

[1] Bharat, K., Chang, B.-W., Henzinger, M. R., and Ruhl, M. Who links to whom: Mining linkage between web sites. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 51–58.

[2] Boldi, P., and Vigna, S. The webgraph framework i: compression techniques. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM, pp. 595–602.

[3] Brin, S., and Page, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117.

[4] Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. Accordion summarization for end-game browsing on PDAs and cellular phones. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2001), ACM, pp. 213–220.

[5] Can, L., Qian, Z., Xiaofeng, M., and Wenyin, L. Postal address detection from Web documents. In *WIRI '05: Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 40–45.

[6] Erkan, G., and Radev, D. R. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research 22* (2004), 457–479.

[7] Fortunato, S., Boguñá, M., Flammini, A., and Menczer, F. On local estimations of PageRank: A mean field approach. *Internet Mathematics*. (to appear).

[8] Google Inc. Detecting query-specific duplicate documents. US Patent no. 6615209, 2003.

[9] Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. DOM-based content extraction of HTML documents. In *WWW '03: Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), ACM, pp. 207–214.

[10] Hearst, M. A. TextTiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist. 23*, 1 (1997), 33–64.

[11] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*, 5 (1999), 604–632.

[12] Meiss, M. R., Menczer, F., Fortunato, S., Flammini, A., and Vespignani, A. Ranking web sites with real user traffic. In *WSDM '08: Proceedings of the international conference on Web search and Web data mining* (New York, NY, USA, 2008), ACM, pp. 65–76.

[13] Pant, G., and Menczer, F. Topical crawling for business intelligence. In *ECDL '03: Proc. 7th European Conference on Research and Advanced Technology for Digital Libraries* (Trondheim, Norway, 2003).

[14] Turpin, A., Tsegay, Y., Hawking, D., and Williams, H. E. Fast generation of result snippets in web search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 127–134.