Boston University Computer Science Technical Report No. 2006-007

Web Mediators for Accessible Browsing

Benjamin N. Waber, John J. Magee, and Margrit Betke*
Computer Science Department
Boston University

Abstract

We present a highly accurate method for classifying web pages based on link percentage, which is the percentage of text characters that are parts of links normalized by the number of all text characters on a web page. K-means clustering is used to create unique thresholds to differentiate index pages and article pages on individual web sites. Index pages contain mostly links to articles and other indices, while article pages contain mostly text. We also present a novel link grouping algorithm using agglomerative hierarchical clustering that groups links in the same spatial neighborhood together while preserving link structure. Grouping allows users with severe disabilities to use a scan-based mechanism to tab through a web page and select items. In experiments, we saw up to a 40-fold reduction in the number of commands needed to click on a link with a scan-based interface, which shows that we can vastly improve the rate of communication for users with disabilities. We used web page classification and link grouping to alter web page display on an accessible web browser that we developed to make a usable browsing interface for users with disabilities. Our classification method consistently outperformed a baseline classifier even when using minimal data to generate article and index clusters, and achieved classification accuracy of 94.0% on web sites with well-formed or slightly malformed HTML, compared with 80.1% accuracy for the baseline classifier.

1 Introduction

People who cannot physically use a mouse, for example because of quadriplegia, often rely on an assistive device that moves the mouse pointer by tracking the user's head or eyes. Computer access

*Contact information: `bwaber@media.mit.edu` and `{mageejo, betke}@cs.bu.edu` This material is based upon work supported by the National Science Foundation, grants IIS-0308213, IIS-039009, IIS-0093367, P200A01031, and EIA-0202067.

with such devices is difficult because they typically do not provide the same selection accuracy as a mouse pointer. Moreover, since the user cannot type with a physical keyboard, text entry, for example of a web address, requires the use of an on-screen keyboard. Selection of a letter on this keyboard or a small text link in a web page may be particularly difficult on traditional browsers for users who experience tremors or other unintentional movements that prevent them from holding the mouse pointer still. One possible solution is to change (1) the display of a web page and (2) how the interface navigates the information based on its content. A page can be rendered and navigated differently depending on the "type" of page. Such classification allows us to create a variety of customizations to occur in interaction mode and display depending on the intended application and user. We could also allow the user to select not just a single link, but a *group* of links. This would allow the web browser to either enlarge the single group of links on the page or, for users who only have control of a binary interface, allow them to scroll through individual links within that group.

The principal technical contributions of this paper are a clustering method to accurately determine the type of a web page based on a technique that examines the text characters on a page and a link grouping method that respects the structure of the web page while providing groupings that substantially increase the effectiveness of browsing.

The clustering method computes the *link percentage*, the percentage of text characters that are parts of links as compared to all text characters on a web page. We posit that there are only two types of pages – articles and index pages – on web sites that deal with news media: article pages contain mostly text and index pages contain mostly links to articles and other indices. This classification may be helpful to allow people with disabilities to browse the web in an effective and efficient fashion. After determining the content of a web page as an index or article, our method can render the page to meet the needs of users with disabilities, for example, by increasing the size of links on index pages. This makes the links easier to read, but more importantly, it makes them easier to select. All text on article pages is enlarged to increase readability.

We also present a novel link grouping algorithm that preserves link structure to enable disabled users to browse web pages orders of magnitude faster than current systems allow.[1] Our link grouping method proceeds in two stages: first it builds a *link tree*. The leaves of the link tree are the links on a web page. The parents of these leaves are the first common parent between different links in the HTML Document Model (DOM) tree [34]. Our method then leverages this structure by moving up from the leaves of the link tree, attempting to group links at their parent node. If all links could not be merged subject to the constraint that the sum-of-squared differences (SSD) error of the new grouping is less than a thresholding function.

Our method for classifying web pages based on link percentage is highly accurate. We used k-means clustering [9] to automatically create unique thresholds to differentiate index pages and article pages on individual web sites. We also used web page classification to alter web page display on an accessible web browser that we developed to make a usable browsing interface for users with disabilities. Our method consistently outperformed a baseline classifier even when using minimal data to generate article and index clusters, and achieved classification accuracy of 94.0% on web sites with slightly malformed HTML, compared with 80.1% accuracy for the baseline classifier.

[1]The link grouping algorithm is implemented in Javascript and can run on any web browser that supports the Document Object Model.

The rest of this paper is organized as follows: section 2 reviews related work in the area of web page classification and point clustering, section 3 describes our accessible web browser, section 4 describes our methodology for detecting web context, while section 5 examines our link grouping algorithm. In section 6 we describe our experiments, section 7 analyzes our results, and section 8 discusses possible avenues for future work. Finally, section 9 summarizes our findings.

2 Related Work

Previous work on classification of web pages into specific types has been limited. It is hoped that this paper will spur interest in the use of web context and computer context in general and to improve accessibility in particular. In the research community, "context" has too often referred to the *physical* context of the user, such as location. Little research has been done on the context of the *computer* environment, which is crucial to understand so that users with disabilities and mobile users can effectively utilize applications. For example, references [24] and [33] discussed this notion of computer context. A notable exception to this is the AVANTI web browser of Stephanidis et al. [31], which utilizes user profiles to modify web pages for individual users. It also incorporated a link review and selection acceleration features, which are particularly relevant to our work.

For mobile users, computer context may be as simple as the currently open applications, while for a person with disabilities computer context includes their disability, the human-computer interface system they are using, the applications they normally use, and numerous other factors. Harnessing the power of context in the web, when we have detailed information on the current state of the application, is a good place to start the investigation of this concept.

We define *web context* as the type of web page the user is currently viewing. Extending this concept to include pages that the user previously browsed in the current session is beyond the scope of this paper, but was examined by Milic-Frayling et al. [20]. Methods for determining context or content of web pages vary widely. Cimiano et al. [6, 7] proposed a system called PANKOW (pattern-based annotation through knowledge on the Web) and its derivative C-PANKOW (context-driven PANKOW). Plessers et al. [25] examined web semantics by proposing a web design method combined with the Dante approach to aid visually impaired users through web annotation. The HearSay system of Ramakrishnan et al. [26], in contrast, uses semantic and structural analysis to provide an audio-based web browser to users with visual impairments. Larson and Gips [18] created a web browser for people with quadriplegia that reads web page text and provides other accessibility options for people with disabilities. Gupta et al. [12] determined web site context (in their case genre, such as news or shopping) in order to facilitate content extraction.

Kim et al. [17] described a method for segmenting topics in discussion boards in order to help blind users more effectively browse the web. Particularly important is that the authors also identified navigational context as an important cue for web browsing, especially for users with disabilities. Milic-Frayling et al. [20] also explored the notion of context by utilizing page importance to implement a web browsing feature called SmartBack. SmartBack functions similarly to the common "Back" button on traditional web browsers except that it attempts to skip to pages of high importance in the browsing history rather than simply the previously visited web page.

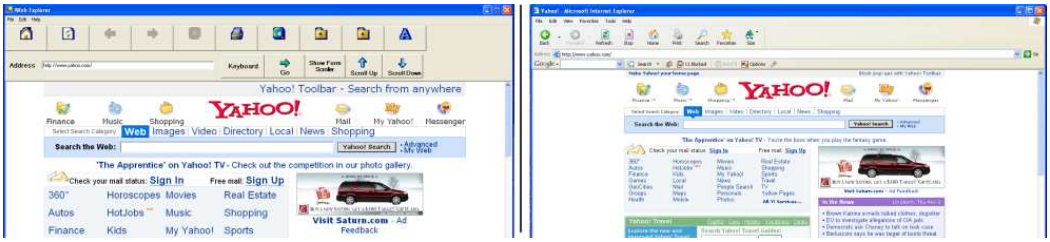

Figure 1: Left: The rendering of an index page in our web browser. Notice that the link text is enlarged relative to the plain text to address the problem of mouse clicks generated by dwell time in mouse substitution devices. Right: The rendering of the same page in Internet Explorer.

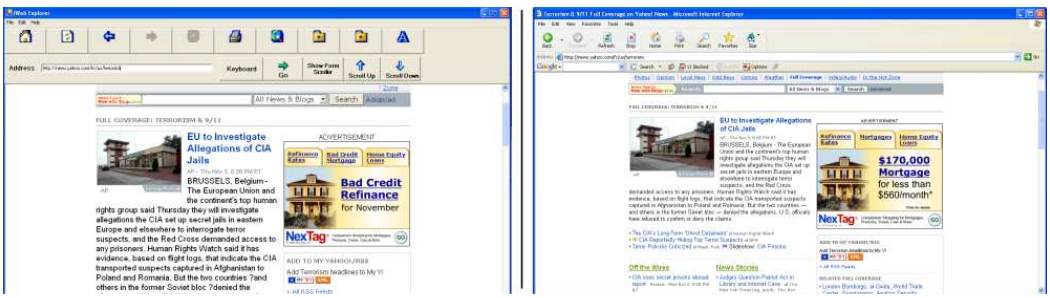

Figure 2: Left: The rendering of an article page in our web browser. All text has been enlarged to enhance readability. Right: The rendering of the same page in Internet Explorer.

Mobile devices usually have small screens and therefore have difficulty fully displaying traditional web pages. Classification of web pages could assist methods that use text summarization to facilitate web browsing on these devices ([3] and [23]) or aid text summarization methods that drive user interfaces for people with disabilities [23]. Index pages, for example, typically display text snippets that are summaries of larger articles, and thus further summarizing these snippets is probably unnecessary. In addition, for text extraction and news delivery purposes knowing the type of page is necessary and an accurate classification method would further enhance the accuracy of these systems. Reis et al. [27] clustered pages by layout features to attempt to distinguish between "section pages" and article pages to facilitate news extraction. Henzinger et al. [15] described a system that suggests news articles on the web based on broadcast news. It is evident that classification of a web page as an article would aid this methodology.

While the web page modification methods that we present here preserve all web page content, Gupta and Kaiser [11] extracted content from web pages in order to render them more accessible. Clearly, our methods could be combined with this extraction to allow for even simpler browsing for users with disabilities.

Various previous work has been performed in re-rendering web pages for mobile devices. Buyukkokten et al. [3] presented a number of text summarization methods for display on personal data assistants (PDAs) or mobile phones. The web pages are broken into segments of text that can be displayed or hidden. Summaries are constructed from keyword extraction or a determination of significant sentences.

Chen et al. [4] detect and organize a web page into a two-level hierarchy. Each section of a page is displayed as a thumbnail that the user can zoom in to view more closely. For pages that are not able to be split, an intelligent block scrolling method is used to present the web page. In a similar fashion, Robbins et al. [29] presented a method for navigating a map interface on Smartphone devices by segmenting a map into nine squares corresponding to keys on the phone. Hornbæk et al. [16] analyzed the effectiveness of these zoomable interfaces for the user's navigation experience. While this work shows results for navigating a map interface, similar conclusions may be drawn for navigating a zoomable web page display interface. This result supports the technique that we chose to alter web page display. Our method essentially also "zooms-in" on a web page.

Many alternative user interfaces limit the number of ways the user can interact with a computer. Various mouse substitution devices, for example, have been developed both for people with disabilities and for other purposes. The EagleEyes [8] project uses electrodes placed around a user's eyes to detect eye movements and translate them into mouse pointer movements on a screen. The Camera Mouse [1] tracks a user's face or other body part to control the mouse based on the user's movements. These interfaces have proven very successful with many users with severe disabilities, however fine "pinpoint" control of the mouse is difficult. Generating a mouse click requires the user to dwell the mouse pointer over the item to be selected for a short period of time. Given the small size of a link in a regular web page, users may have difficulty navigating web sites as they are normally presented.

The accessibility problem for web pages comes with the openness of the web. Web designers are generally free to present their information in whatever layout they find appealing. Drop down menu bars or clickable image maps may aid in the navigation of a web site with the traditional user interface of a mouse, but may hinder the usefulness of the web site for alternative accessibility interfaces. Sullivan and Matson [32] surveyed accessibility on some of the web's most popular sites, while Chi et al. [5] presented a method that automatically generates a web site usability report.

Richards and Hanson [28] examined web accessibility and how user interfaces should be built to facilitate interaction for users with disabilities and the elderly. Harper et al. [13, 14] developed mobility heuristics for visually impaired web surfers and incorporate this into a web browser plug-in to aid navigation. They also investigated the notion of web context as it applies to link anchor text, which is often misleading. They posited that in order for web browsing to be more efficient, some notion of preview and context of the linked page is necessary.

Link analysis has been employed in many different contexts. Lu et al. [19] leverage the structure of the web graph to remove web pages of low *in-degree* from consideration in the PageRank algorithm and achieve high speed up with low decrease in accuracy. Fogaras and Rácz [10], in contrast, use links to facilitate searches of the web graph using graph analysis methods. Bharat et. al [2] similarly use the web graph to study the evolving structure of the web.

Duda et al. [9] summarize a number of point clustering algorithms and implementation techniques. Particularly relevant to this work is their description of agglomerative hierarchical clustering and clustering in the presence of unknown data structure. Agglomerative hierarchical clustering creates clusters by merging the closest clusters together until the desired number of clusters is reached, thus giving the result a minimum variance flavor. They also describe methods for evaluating the validity of cluster splitting by examining the behavior of a fitness function as the number of clusters are increased, stopping splitting only if the split results in a fitness function increase that falls below that found by a thresholding function.

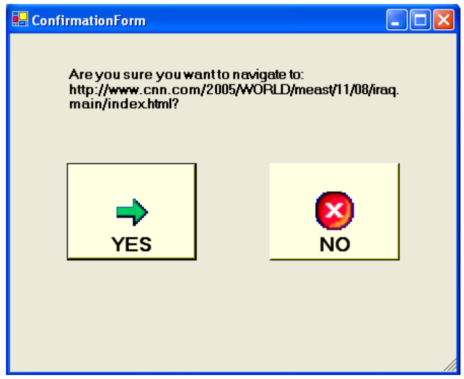

Figure 3: Interface window to confirm that the user wanted to select a link.

3 Web Browser

In our IWeb Explorer web browser [22], we addressed seven problems that users with disabilities had with traditional web browsers such as Microsoft Internet Explorer and on-screen keyboards:

1. The buttons in the toolbar and scrollbar are too small.
2. There is no way to type in the URL address without using the keyboard.
3. The links are all very close together in each of the pages.
4. It is easy to unintentionally end up following the wrong link since there is no check to make sure that the user has not mistakenly selected a link.
5. Each user has different abilities and preferences and so the button sizes and colors in the keyboard and interface windows should not be fixed.
6. The favorites menu is not easily accessible and the links again are small and are difficult to select for people with disabilities.
7. It is difficult to reach keys and links on the far corners of the computer screen. Therefore a scrolling device is needed to scroll around the web browser interface.

Screenshots of our browser for both an article and index page compared to the rendering provided by Internet Explorer are shown in figures 1 and 2. The link text clearly stands out on index pages much more in our browser since it is enlarged to address the problem of mouse clicks generated by dwell time in mouse substitution devices, while article text is enlarged to enhance readability. This is done automatically using our web context recognition method, which is described below.

Perhaps the most glaring problem of conventional web browsers is the lack of an opportunity for the user to confirm that a link was selected correctly. Our confirmation window, shown in figure 3, has yielded a positive response in preliminary tests with users with disabilities, and we hope to further improve this browser by allowing users and their caretakers to change the way that different pages are displayed to suit their individual needs.

When people with disabilities used this web browser without our web context component, effective web browsing was not attainable [22]. It therefore became clear that to make web browsing applications viable for all users, web context needed to be leveraged.

4 Web Context Recognition

4.1 Page Classification

The key observation of our technique is that by examining link percentage we can accurately determine the "type" of a web page. The link percentage is the percentage of text characters that are parts of links as compared to all text characters on a web page. We posit that on sites dealing with news media that there are only two types of pages: articles and indices, where articles contain mostly text and indices contain mostly links to articles and other indices. Below we refer to pages as "dynamic" if their contents change from day to day.

While the idea to classify web pages based on link percentage seems intuitive, the question is, is it actually feasible to break down pages into categories by this one-dimensional characteristic? Figure 4 gives an example where a dynamic index page and multiple article pages are clearly separable over time. Does this mean that a single threshold on link percentage will be an accurate classifier for all web sites? From the graphs of the link percentage of article and index pages drawn from four popular web sites over a period of two weeks shown in figure 5, it is clear that a single threshold does not suffice. Index pages have a higher link percentage than articles in most cases, but a single threshold cannot separate these two types of pages across the web.

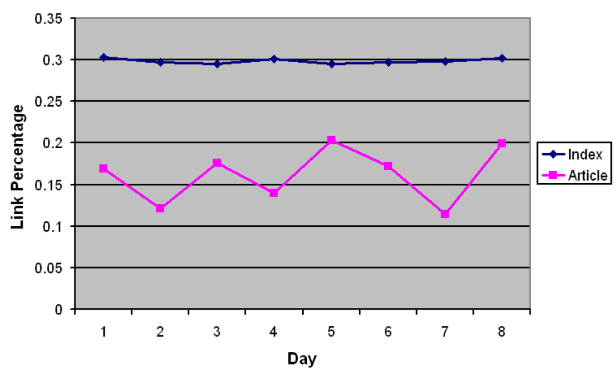

Figure 4: The link percentage the BBC web site over a period of 8 days. The index page that was chosen was the same, but different articles were chosen at random every day. Observe that there is a clear distinction in link percentage between the index page and the article pages.

For the 50 web sites that we tested, the average ratio of link percentage in index pages to link percentage in article pages was approximately 3, including malformed HTML characters (for the interested reader, the web site that had the highest link percentage ratio in our corpus was that of the Real Madrid football team, with an average index page link percentage of 95.3% and an average article page link percentage of 0.001%). Some examples from our web site corpus are shown in figure 6.

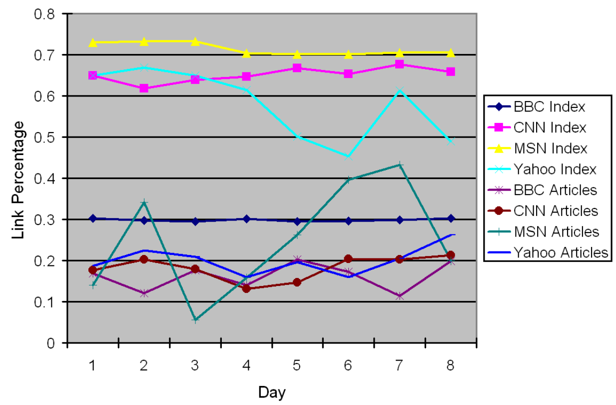

Figure 5: The link percentage for various sites over a period of 8 days. The examined index pages are dynamic and typically change every day. Different articles at these sites were chosen at random every day. While it is apparent that index pages have higher link percentages than articles, there is not one threshold that can separate these two types of pages for all web sites, as shown by the MSN article link percentage rising above the link percentage for the BBC index page. It is also important to notice that, except for the Yahoo index page, the dynamic index pages' link percentages do not change by more than 3%. The behavior of the Yahoo index page's link percentage is due to malformed HTML.

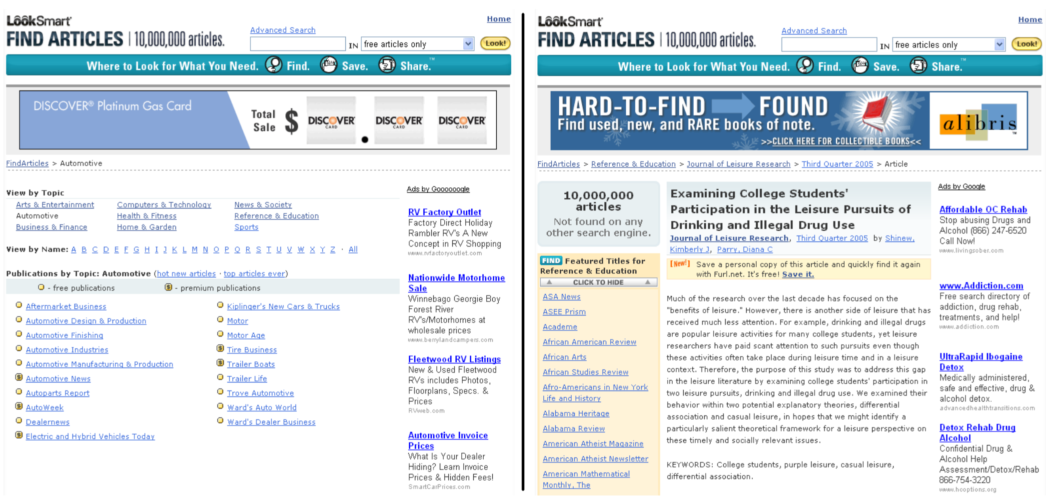

Figure 6: Two web pages included in our corpus as rendered in Microsoft Internet Explorer. Left: The index page for the Looksmart FindArticles site (link percentage: 60.2). Right: An article on the Looksmart FindArtilces site (link percentage: 11.5).

To find these link percentages, we use an HTML parsing mechanism that works on most web pages with well-formed HTML. HTML is often not well-formed, however, so we perform some further processing after the link text and plain text has been extracted from the HTML code by removing as many extraneous tags as possible. Extra spaces are also collapsed to a single character to better represent the actual amount of text that is within a page.

An important issue for our algorithm is the notion of character codes. In particular, for characters such as '*&*' and '<', the character codes used will be longer than the actual single character value. Thus we replace these codes in our extracted text strings with their character value.

The substitution does not apply to pages in languages with different character sets that use character codes to represent text. As long as the language is consistent across a page the link percentage distinction remains intact. This is demonstrated by our results from some Japanese news web sites, shown in figure 7. The index and article pages are still separable, implying that our methodology works correctly here as well.

Once we have parsed a web page's HTML code and determined the link percentage, the issue of determining a proper threshold arises. If the user has not visited other web pages from this site, then we can only use generic thresholds to determine the type of a page. An initial threshold of 0.4 was used in experiments and found to perform reasonably well. After the user has visited at least one page of each type, however, we can begin to discover how the link percentage values of index pages and article pages are clustered. Using the k-means clustering algorithm for each web site, with $k = 2$, one cluster for article pages and one cluster for index pages, we can accurately classify future web pages from this site. We use as the initial mean points of the cluster the pages with the lowest and highest link percentage for the article and index clusters, respectively. We then run the k-means algorithm to determine the final clusters. Using these clusters, we choose as our decision threshold the value midway between the link percentage of the page with the highest link percentage in the article cluster and the link percentage of the page with the lowest link percentage in the index cluster. An example where clusters are separated by a threshold computed in this way is given in figure 8. The k-means algorithm can be viewed as a method to approximate the maximum-likelihood estimates for the means of the clusters.

We observed that even dynamic web pages' link percentages do not fluctuate very much over short periods of time. We studied four popular websites over the course of seven months and found that the link percentages for the same index pages had a total range of less than eight percent. Therefore, once a page is visited its link percentage is stored in a database and is retrieved if the page is visited again and no HTML parsing is performed. This saves computational effort and can easily be overridden by the user if the classification results falter because of a change in the web site structure.

4.1.1 Database Storage

Each user's database of visited web pages and web sites is stored on the user's computer in a hash table which is processed by the web context program prior to browsing. We could, alternatively, have stored the link percentages of pages and web site thresholds in an online database. In this scheme, users would first query an online database system with the URL of the page that they will download. The database would either respond with that page's classification or state that

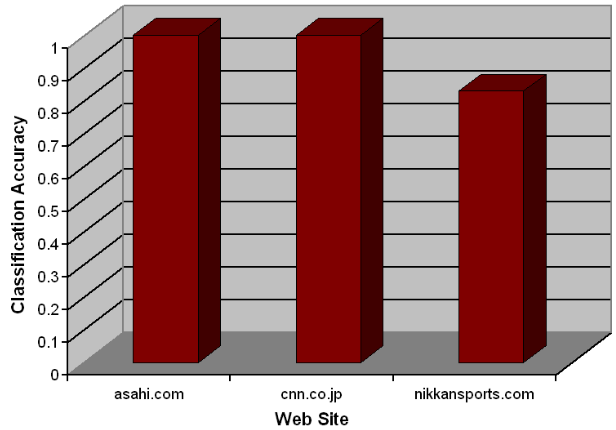

Figure 7: The classification accuracy of our method on three Japanese web sites. Six pages, three indices and three articles, were used for testing, while a single index page and a single article page were used to create the two clusters.

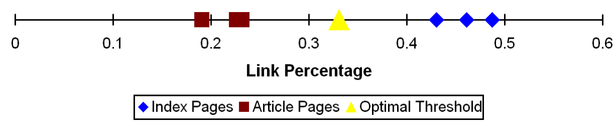

Figure 8: The optimal threshold for link percentage on the Yahoo Sports web site given the link percentages of three index pages and three article pages.

the page is not in the database. If the page was not in the database, the user could then upload the web page's link percentage after the page is loaded.

Clearly, there are problems with this approach. Privacy concerns rank chief among these, since the user would be informing the database of their browsing behavior at all times. In addition, user-defined behavior, such as changing the classification of a certain web page for individual reasons, played a factor in our decision to keep the database on the local machine. Finally, an online database would introduce wasted computational effort employed in searching such a large database since most users visit a narrow band of web sites during the course of normal browsing [21].

Clusters for individual web sites are also stored locally for similar reasons, and since clusters are computed extremely quickly and only when the user ends their browsing session this imposes a minimal computational burden on the user's system.

4.1.2 URL Analysis Feature

We initially attempted to use analysis of the web page URL as a fast, effective indicator of page type. This method evaluates a series of conditions to classify a web page (a more sophisticated URL analysis method is presented in [30]). If the page URL satisfies any of the conditions, then it is classified as an index. Otherwise, it is classified as an article. The conditions are:

1. The web page has the same URL as the root of the URL (for example, the root of www.cnn.com/WORLD/ is www.cnn.com).

2. The URL ends with a "/", indicating that it is the index of some subdirectory.

3. The URL ends with "index.*", where "*" is a valid file name extension (such as .html, .jsp, .psp).

This approach yields fairly good results in certain cases, but it fails in a number of situations, since URL naming conventions appear to differ widely across the web. Particularly troublesome are articles where the URL satisfied the third constraint since the web page represented a summary of all news pertaining to a certain event, such as the URL for an article on a political story on CNN.com: http://www.cnn.com/ ... /delay.indiantribes.ap/index.html. Even if we remove step 3, however, numerous index pages were still falsely classified. Thus it was determined that the far more accurate link percentage analysis was preferable, since the pages that the URL analysis method faltered on were the very pages that often presented trouble for the k-means clustering method.

4.2 Customized Page Display

Our web browser enlarged link text on index pages to support mouse substitution interfaces that use dwell time to generate mouse clicks. The browser also enlarged plain text on article pages to enhance readability. This is just one possible display modification, and for certain web pages this may or may not be useful. The user can undo these display modifications if they wish, putting the ultimate decision of the page's display in their hands (see figures 1 and 2 for a comparison of our browser's rendering of a web page using web context to that of Internet Explorer).

We could further enhance the user's interaction experience by changing the way that keyboard or mouse commands are issued depending on the type of web page that the user is viewing in order to facilitate navigation. Moreover, we could provide the user with tools to create their own rules for modifying web page display based on page type. This kind of control is extremely crucial for people with disabilities. With many of the currently available assistive interface systems web browsing is still difficult. For example, using an interface system such as the Camera Mouse [1] to select links or scroll down a page is hard even for users without disabilities. With our method, these users could scroll down an article web page merely by moving the mouse pointer to the left half of the screen, or iteratively cycle through links by performing the same action on an index page. It is our hope that this method can alleviate some of the accessibility problems that people with disabilities have with current interface systems. We also hope that future work will place more emphasis on the context of their actions to enrich the interactive experience and make it more effective and efficient.

Another aspect of customized page display is the use of manual corrections if a user has a preference for a different display than provided by our method or if the web page was misclassified, which occurs when pages were created with malformed HTML (see below). We decided that the best course of action is to simply omit the page in question from the clustering algorithm altogether.

5 Link Grouping

Link grouping allows users with severe disabilities to use a scan-based mechanism to tab through a web page and select item. This can substantially improve the user's rate of communication, and could be applied to mouse substitution interfaces by allowing users to click on a group of links so they can more easily select the desired single link.

5.1 Link Tree Creation

The link tree creation step of our link grouping algorithm creates the framework under which we can cluster points according to their location on the web page as well as their location in the HTML code. Essentially our method leverages the structure of the HTML code, which can be quite nicely represented in tree form by allowing an element to be a node in a tree and the elements that it encapsulates to be its children. An example link tree along with the HTML code that it was created from is picture in figure 9. The tree representation allows us to employ a divide-and-conquer method from grouping as described below. Our method can also use this tree to constrain link grouping so that groups do not span inappropriately across the web page. Note that it is not that such across-page grouping is incorrect per se, rather that it would create a very unnatural grouping consisting of circles of links that did not respect the structure of the web page. In addition, this would leave us with a very unconstrained clustering problem, and we certainly prefer an approach that can apply the divide-and-conquer paradigm to the grouping problem.

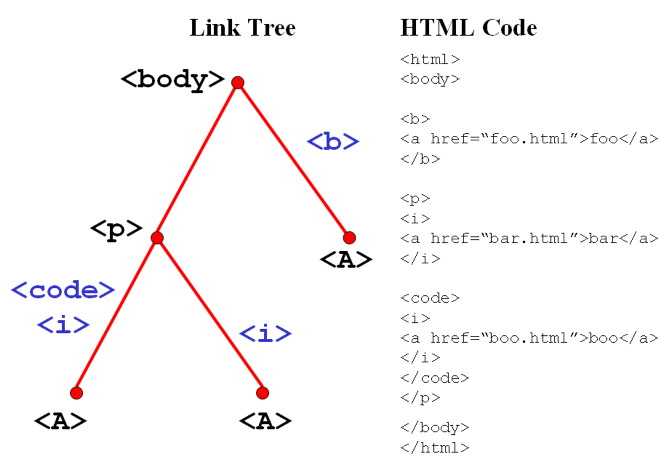

Figure 9: The left figure represents the link tree for the HTML code on the right. At the leaves are the link nodes, and the internal nodes are the first common parent of its children in the DOM tree. The parents of each node in the DOM tree that are not represented as internal nodes of the link tree are shown in blue. This example can best be understood by examining the HTML code and viewing the structure of the DOM tree. Observe that the first common parent of the link to "bar.html" and the link to "boo.html" is the "<P>" node. Therefore this is their parent in the link tree. Next, note that the first common parent of the link to "foo.html" and the "<P>" node is the "<BODY>" element, which is the root of the tree.

The pseudocode for link tree creation is shown in algorithm 1. Our algorithm, *CreateLinkTree*, takes as input an HTML DOM tree and outputs the link tree. The algorithm starts at the links of the DOM tree and traverses it until it reaches the root node of the document, marking every node that it visits along the traversal (line 9). If, however, it comes across an already marked node, our algorithm converts that node into an internal node in the link tree, connected to the link nodes that already marked it (lines 5 and 6). If that link node has already been incorporated into a subtree of the link tree, that subtree becomes the current node's child. Traversal of the DOM tree then stops. The last case that can occur is that a link traversal will arrive at a node that is already an internal node in the link tree. Here the link will simply add itself as another child to that node and stop traversal.

In the worst case, the links do not intersect until the "<BODY>" node of the DOM tree (since this node must encapsulate all links). Then, if there are n links, and the height of the DOM tree from the link nodes is $log(q)$, where q is the number of elements in the DOM tree, it is clear that the link tree creation algorithm is bounded by $O(nlog(q))$, since our method must make n traversals of length $log(q)$. Note that typically $n \ll q$.

Algorithm 1 CreateLinkTree: *Input:* HTML DOM Tree T, *Output:* Root of the Link tree.

```
1:  for each link a in T
2:      traverser = a
3:      while traverser.parent != root
4:          traverser.parent.child = a
5:          if marked(traverser.parent)
6:              make_node(traverser.parent)
7:              break
8:          end if
9:          mark(traverser.parent)
10:         traverser = traverser.parent
11:     end while
12: end for
13: return root
```

5.2 Link Group Creation

The previous step of our algorithm, that of link tree creation, has now given our method the machinery to perform link grouping. For all of the clustering steps below, our algorithm, GroupLinks, uses a set of points defined by the position of the links on the rendered web page in Cartesian coordinates. In general, our algorithm traverses the link tree from the top down and attempts to merge the clusters of link points of one of its child nodes with those of another child node if each child has only one link cluster. The optimal number of clusters is chosen by the criterion function defined below. Pseudocode for the link grouping algorithm is given in algorithm 2.

More specifically, using the link tree as a structural guide, the algorithm recursively breaks down the clustering problem into that of merging the link point clusters of children nodes together, starting at the root node, by a criterion function. If our method cannot merge all of the children of some internal node together, then, intuitively, these link groups should be excluded from merging with other groups at higher levels in the tree. As stated earlier, doing so would violate our constraint of respecting the structure of the web page.

At the current node, call it node i, the method first runs the link grouping algorithm on all of its children (line 4). If node i has no children, it is a link and thus simply returns its position to its parent as a single cluster (line 2). Otherwise, the algorithm checks which of node i's children *can* be merged (line 7), which is the case if each child returned only a single cluster of points. For all of node i's children that can be merged, our method runs the agglomerative hierarchical clustering algorithm on the mean points of each of its children's clusters (line 8). This algorithm essentially merges the closest clusters at every step. As stated above it is for this reason that we can expect the clusters to have low variance. Assuming that there are c children of node i, this gives our algorithm a total of c clusters before i is processed.

The next step is to determine the optimal number of clusters. We do this using the equations (due to [9]):

$$\frac{J(k+1)}{J(k)} < 1 - \frac{2}{d\pi} - \alpha\sqrt{\frac{2(1-8/\pi^2 d)}{nd}} \tag{1}$$

where $J(k)$ is the SSD error for k clusters, d is the distance between the means of the clusters that were split to create $b+1$ clusters from b clusters, and n is the number of points in all clusters. The parameter α is determined by solving the equation:

$$\alpha = \sqrt{2} \times \mathrm{erf}^{-1}(1 - \frac{p}{2}) \tag{2}$$

where erf is the Gauss error function, defined by the equation:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{3}$$

Here, p is the significance level at which we believe that we have at least $k+1$ clusters. Our algorithm starts with $k = 1$, stopping once the inequality in equation 1 is violated. It then returns the resulting clusters.

Naturally, having $k+1$ clusters will yield a lower SSD error than k clusters. Essentially, what equation 1 does is model the clusters as k different normal distributions and check that the error reduction that we see is not due to chance at the p significance level, since if there actually were only k clusters we would expect any other clusters that formed to be formed by chance.

Once the final clusters have been returned to the root node, we can modify the web page to make it more accessible to users with disabilities. While this modification mechanism can be accomplished by various parameterized functions, in our implementation we choose to use color to identify links in the same group, using different colors for different groups. This is shown in figures 10 and 11. Other options include link enlargement when the mouse cursor hovers over a link group, making all links in a group lead to a page where the links are made very large for easy navigation, and many other possibilities which we will explore further in the Future Work section.

The link grouping creation algorithm is bounded by $O(n^3)$ in the worst case, where n is the number of links, since our method must find the closest group at every step of hierarchical clustering, for n steps. Note that this worst case is only realized if every link has only one common

---

Algorithm 2 GroupLinks: *Input:* Link Tree Node R, Significance p, *Output:* Link Groupings for Tree Rooted at R.

---

if isLeaf(R)
    return R.position
end if
for each child c of R
    groups(c) = **GroupLinks**(c)
end for
mergeGroups = { group(c) } s.t.
num_clusters(group(c)) = 1
create_hierarchical_clusters(mergeGroups)
current_clusters = 1
while equation 1 is not satisfied
    current_clusters++
end while
return clusters(current_clusters) +
        Σ (**num_clusters**(group(c)) | **num_clusters**(group(c)) ≠ 1)]

Background [edit]

Heimaey before the eruption

Iceland is a region of frequent volcanic activity, due to its location astride the Mid-Atlantic Ridge, where the North American and Eurasian Plates are moving apart, and also over the Iceland hotspot, which greatly enhances the volcanic activity. It is estimated that a third of all the basaltic lava erupted in the world in recorded history has been produced by Icelandic eruptions.

The Vestmannaeyjar (Icelandic for Westman islands) archipelago lies off the south coast of Iceland, and consists of several small islands, all formed by eruptions in the Holocene epoch. Heimaey, the largest island in the group and the only inhabited one, also contains some material from the Pleistocene era. The most prominent feature on Heimaey before 1973 was Helgafell, a 200 m (650 ft) high volcanic cone formed in an eruption about 5,000 years ago.

The Vestmannaeyjar archipelago was settled in about 874 AD, originally by escaped Irish slaves belonging to Norse settlers on the mainland. These settlers gave the islands their name, Ireland being west of mainland Scandinavia. Although plagued by poor water supplies and piracy during much of its history, Heimaey became the most important centre of the Icelandic fishing industry, having one of the few good harbours on the southern side of the country, and being situated in very rich fishing grounds.

Since the settlement, no eruptions had been known to occur on the islands until 1963, when a new member of the archipelago, Surtsey, was formed by a four year eruption which began offshore about 20 km (12 mi) south-west of Heimaey. However, offshore eruptions may have taken place in 1637 and 1896. Scientists have speculated that volcanic activity in the archipelago may be increasing due to the southward propagation of the rift zone which crosses Iceland.

Figure 10: A Wikipedia (http://www.wikipedia.org) web page that has been processed by our link grouping algorithm. Note that the different link groups are in different colors so that the user can easily pick out different groups. In this page, shown only partially here, 66 clusters were found.

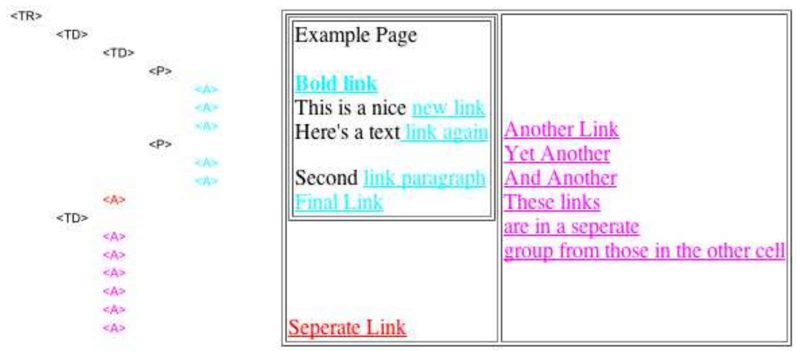

Figure 11: Left: The link tree for the web page on the right. The links in the tree have the color that they were given in the web page after they were processed by our link grouping algorithm. Right: An example web page processed by the link grouping algorithm.

parent at the root of the link tree, which does not happen in practice since that would require essentially no other content on the web page. Since the link tree exhibits the structure of a tree with branching factor b in the average, the algorithm is bounded by $O(b^2 log_b(n))$, since the height of the link tree is $log_b(n)$ and the algorithm needs to create b clusters at each level of the tree. If no links can be merged, the algorithm stops at the leaves of the link tree and is bounded by $O(b^2)$. In practice, we measured a runtime that falls somewhere in between the last two bounds that we derived, since typically we can merge only some of the link point clusters at each node. Note that the runtime depends heavily upon the p used in equation 2, since p essentially bounds the size of a cluster in 2D space. The complete algorithm, including the creation of the link tree, then, for the bound on the link grouping algorithm of $O(b^2 log_b(n))$, is simply $O(nlog(q))$, as long as we have $b \ll n$, which is again normally the case.

Suppose that our user is using a simple tab interface which requires the user to press the tab button once to move to the next link and enter to click on the link. Using our grouping algorithm, this user could select a specific group and then an individual link. Assuming that there are c final clusters and that each group has an average of s links, the average number of tabs required to click on a link would be $\frac{c+s}{2}$. In comparison, the tab-based interface that is currently employed on commercial web browsers has an average number of tabs of $\frac{n}{2}$. Therefore, our method improves the communication rate of users by a factor of $n/(c+s)$. Note that if s is too large, we could simply split each cluster into subgroups in order to minimize the number of tabs required and maximize $n/(c+s)$, but that is left to future work. As a precaution, if only one cluster is created then the grouping is not used, since this would result in requiring the user to press the tab button once just to be able to perform the original selection task. Clearly, in the worst case, where the number of clusters equals the number of links, our method performs as well as the current tab-based implementations on web browsers, and due to the trivial computational cost of our grouping algorithm (the highest number of links in our web corpus of 300 pages was 125, and the link grouping program ran nearly instantaneously), our method could be an integral component for an accessible web browser, or any web browser in general.

6 Experiments

To test our web context method we used it to classify web pages from a corpus of the top 25 news and top 25 sports web sites as rated by Alexa Web Search (URL: www.alexa.com). For each web site, three index pages and three article pages, as categorized by a human observer, were used for testing for a total of 300 web pages. One index page and one article page from each web site were randomly chosen to create an index cluster and an article cluster in the web context program, and these pages were not included in the test set.

We compared our method to a static threshold technique, which used a predetermined global threshold (a link percentage of 0.4) to differentiate between index pages and articles. We treat this classifier as the baseline in our discussion of results, since it is the simplest solution to the classification problem. We also compared our method to an "optimal" classifier, which chooses the best possible classification threshold for each web site. This is merely a theoretical classifier; given complete knowledge of what the correct classifications are, it finds the optimal classification threshold.

There were many web pages that contained severely malformed HTML, as evidenced by the fact that even the optimal classifier did not generate 100% accuracy on every web site. The results are shown in figures 12, 13, and 14, broken down by site category (news or sports) and combined. The average classification accuracy for each method is shown in table 1, and a graph comparing the static threshold method and the clustering method is shown in figure 15. In figures 12, 13, 14, and 16 accuracy of 100% implies that all pages in the test set for a particular web site were classified correctly, while 83% implies that five out of the six pages in the test set were classified correctly, and so on.

To evaluate our link grouping method, we tested it on a number of web pages from our original corpus using a significance level $p = 0.001$, which kept the link groups to a reasonable size. We discuss the results of these experiments below.

Table 1: Classification accuracy of three classifiers as the fraction of the number of correctly labeled web pages out of 300 test web pages.

	Static Threshold	Clustering	Optimal Classifier
News Pages	0.734	0.840	0.946
Sports Pages	0.700	0.760	0.866
All Pages	0.717	0.800	0.906

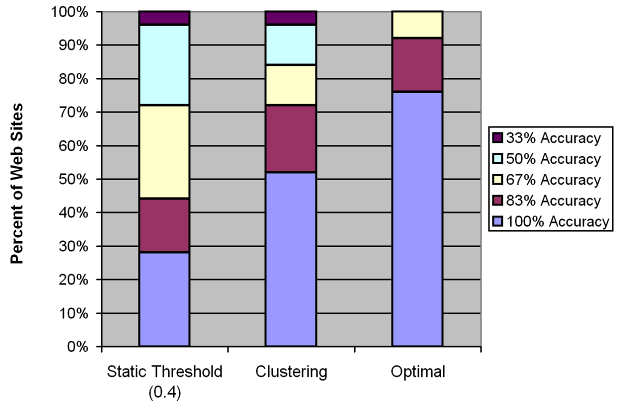

Figure 12: The classification accuracies for the three different classifiers on 25 news web sites, containing a total of 150 web pages.

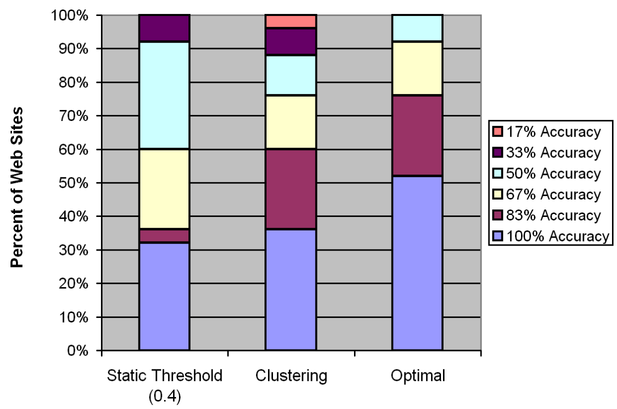

Figure 13: The classification accuracies for the three different classifiers on 25 sports web sites, containing a total of 150 web pages.

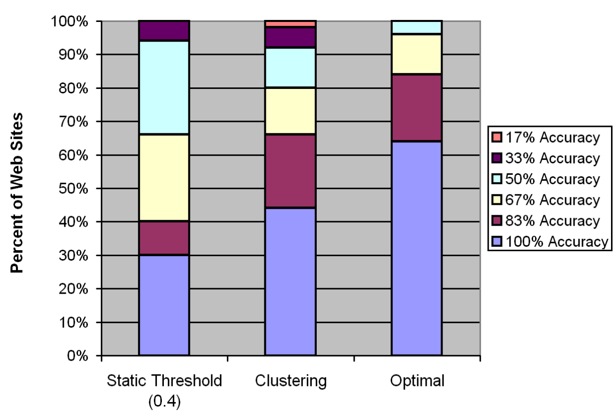

Figure 14: The classification accuracies for the three different classifiers on all 50 web sites containing a total of 300 web pages.

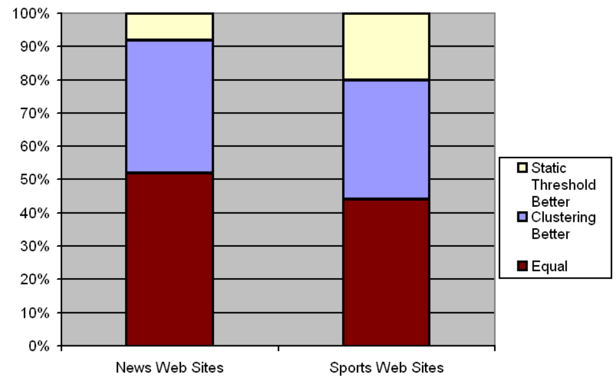

Figure 15: A comparison of the static threshold classifier and the clustering method. As the HTML parsing improves, evidenced by the higher accuracy of the optimal classifier for the news web site data set, so does the performance of our clustering method.

7 Results and Discussion

7.1 Web Context

In our experiments, the clustering method gave higher performance than the static threshold method, correctly classifying 80.0% of the test pages compared to the static threshold's 71.7%. Our method's accuracy, however, is clearly not equivalent to the optimal classifier, which classified 90.6% of the pages correctly. The other 9.4% of pages had highly malformed HTML code, since when the experimenter hand labeled the link and plain text, these pages were found to cluster in the expected fashion and often the hand labeled link percentages differed from those given by the parsing program by over 40%.

It is also interesting that our method achieved better results than the static threshold 38% of the time, while only having poorer performance in 14% of our experiments. This result is encouraging because it shows that we are consistently outperforming the baseline classifier and reaching a level of performance that is close to that of the optimal classifier.

It also makes sense to ask how the static threshold and clustering methods compare when all web sites where the optimal classifier did not achieve 100% accuracy are removed. Then we see the clustering method achieved 94.0% accuracy, while the static threshold technique rose to only 80.1% accuracy. These results are shown in figure 16. The clustering method performed better than the static threshold on 47% of the web sites, and the static threshold method performed better on only 9.4% of sites. This indicates that as HTML parsing accuracy increases, our clustering method gets closer to 100% accuracy. Indeed, there are still some parsing errors left over in this group of sites, only less of them and of smaller magnitude. It seems clear that combining our method with an "HTML cleaning technique" such as that introduced in [35] would yield extremely high levels of accuracy.

The strength of our method is that it finds the proper threshold between index and article pages in a web site given little data. The fact that we do not know *a priori* which web pages are indices and which are articles poses a problem if the user visits only pages of a single type. The algorithm will then assume that one of the pages is in fact an index and thus erroneous clusters

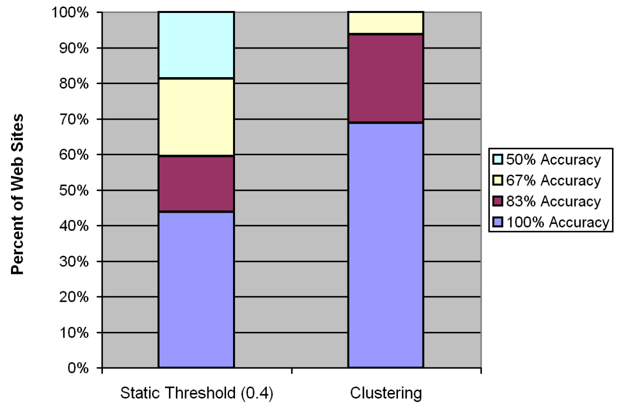

Figure 16: The classification accuracy for the static threshold and clustering methods on sites for which the optimal classifier achieved 100% accuracy. Our clustering method correctly classified 94% of web pages, while the static threshold method only classified 80% correctly.

will emerge until the user visits a page of the other type. This is not a problem if the user can turn off our web page classification method for certain sites that do not have different kinds of pages. It is important to make the user aware of this caveat, lest they prematurely turn off the algorithm for sites where it would work appropriately if it had more information.

With regards to our clustering algorithm, it may be desirable for manual corrections to force the initial mean points to be the mean of the web pages that were previously classified as articles. This comes with the danger of an initial poor classification that was not corrected corrupting future classification attempts, so we decided against this modification.

Note that, in general, users will collect more data from each site over the course of normal browsing, so we would expect even better results than those reported here. Our experiments are meant to show merely the bare minimum of what our method is capable of.

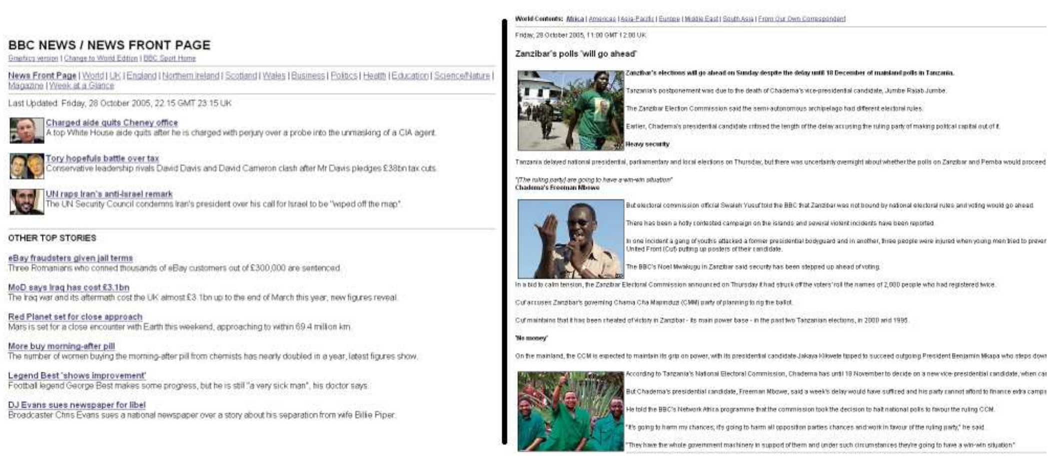

Figure 17: Left: The index page for the BBC News site. Right: An article on the BBC News site. Both web pages have three images, so a classification algorithm based solely on number of images would not work in this case.

We acknowledge that we could have used other features such as periods, punctuation character percentage, or word rather than character link percentage. But these do not capture the page type data as accurately as link percentage. Period and punctuation percentages perform poorly because they are often used as delimiters within a page, and occasionally index pages have many small text snippets, such as when small summaries for different articles are present, giving these pages more punctuation characters than an article page. Word link percentage, or the percentage of words that were part of links divided by the total number of words, on the other hand, treats words such as "a" and "interestingly" as equally weighted words, when clearly "interestingly" takes up more space than "a" on the rendered page and so contributes more to its type. The two page types appeared less separable using this feature, so we opted in favor of the (character) link percentage cue, although admittedly using word link percentage may blunt the influence of malformed HTML.

It also may be unsettling that we ignore images, display markup, and position information. While we realize that these are important parts of a page's content, it is difficult to develop rules that would generalize to the entire web, since some articles have many images while index pages have very few (see, for example, figure 17). Some images, however, are used as links in place of text. It is unclear exactly how influence should be computed for these images, but these images may prove useful as an additional cue in a future extension of our system, although currently they are ignored during processing.

In addition, text markup is used in many different ways with rather loose rules governing their use, and given that our classification performance is extremely high, it does not seem that the extra processing required would generate large gains in accuracy.

7.2 Link Grouping

The results for link grouping varied widely, with the reduction in tabbing by the factor of $n/(c+s)$, spanning from a factor of 40 to an improvement of only 1.5, with a mean of 12. In addition, our algorithm on average took 0.2 seconds to run, with the longest time at 0.8 seconds. Thus our method clearly runs in real time and provides real performance gains for the user.

Our link grouping algorithm has shown itself to be quite versatile and effective, working across web pages in multiple languages with a myriad of layouts. While is difficult to state what the "correct" link grouping would be in an objective manner, personal experience with the algorithm has shown that it does indeed respect the layout of the page and provides very intuitive groupings on most pages. As a bonus, our method is easy to plug in to any web browser, since it was built entirely in code that can be inserted directly into a web page by a browser.

Actual modification of web page display, however, did not receive as much attention as the algorithm itself. There are innumerable possibilities for modifications, and these vary drastically with the intended audience. We will explore some of these possibilities below.

8 Future Work

Extracted HTML text characters can clearly be used to form a very accurate classification algorithm, but in order to push accuracy higher we may need to use other cues. Using rendering data to weight text according to its centrality in the displayed page (i.e. weighting text that is closer to the top and middle higher than text that is more towards the sides and bottom) appears to be an attractive extension, although it is not clear if a general rule can be developed that works for a broad segment of web sites.

We may also wish to handle pages that contain a high volume of images used as links rather than text links. While not encountered in our test corpus, handling of such pages is crucial, and perhaps assigning a default weight to each image-link would further improve results.

Detecting web pages and sites that are merely Java applets or Flash programs is also important, since we can no longer determine the optimal mode of interaction with the web page. If, however, we allow the user to specify what mode of interaction to use when they visit this page, then we can contact the interface system each time this page is visited again and instruct it to output commands according to the user's specification. This is a very useful feature that will likely be implemented in our web browser in future work.

It may also be useful to segment the rendered page into different regions using a decomposition method such as that introduced by Chen et al. [4] and then classify each of the regions using our method. The classification of the entire page could come from a weighted sum of the classifications of each region. This decomposition could also aid in interaction modification, since we can imagine displaying regions of different types in different ways and changing the user interaction method if they select a particular region. This would incur a higher computational cost, but it may be a necessary extension to further utilize web context on PDA-class devices.

We could extend our approach to web page classification into the image processing realm by using a bitmap image of the rendered page to classify text regions and other regions using pixel information only. This would be insulated from many of the problems of parsing malformed HTML, but this type of algorithm would be computationally expensive and require the page to be rendered before it is altered, placing further burden on the user. It may be useful to combine this image-based method with our current classification scheme, however, to yield a more robust estimate of page type.

One issue that we touched upon earlier was sites that have only one page type or web sites where only pages of one type are visited. To handle this case automatically it may be necessary to first attempt to fit a single cluster to all web pages on a site and then see if the fit is acceptable. If not, then the regular algorithm can resume. Characterizing what constitutes a "good" fit may prove troublesome and complicates this technique.

There are also web sites which do not fall within the domain of sports or news web sites that may have multiple types of pages. A major component of future work is to identify these page types and examine if they generalize across a wide range of web sites as the article and index types do. If it appears that new page types provide a nice fit for a wide range of web sites, incorporating these types into a future algorithm would be a definite possibility. It may be, however, that beyond the basic article-index distinction different interaction modes and display modifications are not useful. This issue clearly demands further research.

We also wish to extend our web browser to give more control to the user in displaying different types of web pages. We are experimenting with various ways to offer this functionality, and it will surely create a greatly enhanced interaction experience for the user. Detailed experiments on how page display modification and changes in the interaction mode positively impact the user interface experience will also be performed to further validate our results. This is particularly relevant to our link grouping algorithm, which could allow users to click on a link group to enlarge that group of links, or to highlight a link group in a different color as it's selected. What options are most user-friendly and intuitive would make for interesting future research.

Our work is a preliminary step into the larger investigation of computer context. In later work we would also like examine other types of computer context and investigate whether or not extending web context to include previously browsed pages is feasible. Work in this area has only just begun.

9 Conclusion

We have presented a highly accurate method for classifying web pages based on link percentage. Our k-means clustering method created unique thresholds to differentiate index pages and article pages on individual web sites. Accuracy increased when we removed web sites from the corpus that had extremely malformed HTML, and it is expected that more robust HTML parsing will yield even more accurate results. Our method consistently outperformed a baseline classifier even when using minimal data to generate initial article and index clusters.

Our link tree creation algorithm and link grouping method have been shown to be quite effective and guaranteed to outperform or at least stay at the same level as the functionality offered by current web browsers, leading to a possible improved communication rate for users with disabilities. This method is fast and is portable to nearly all available web browsers, giving it promise to become an integral tool for web accessibility.

We also used web page classification and link grouping to alter web page display on a web browser, and future work will center around giving the user more control in determining how different types of web pages are displayed and choosing intuitive ways to change interaction modes of an interface system based on web page classification and link grouping.

References

[1] M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10, Mar. 2002.

[2] K. Bharat, B. Chang, M. Henzinger, and M. Ruhl. Who links to whom: Mining linkage between web sites. In *International Conference on Data Mining (ICDM)*, pages 51–58, 2001.

[3] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: text summarization for web browsing on handheld devices. In *Proceedings of the 10th International World Wide Web Conference (WWW)*, pages 652–662, Hong Kong, 2001.

[4] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 225–233, Budapest, Hungary, 2003.

[5] E. H.-H. Chi, A. Rosien, G. Supattanasiri, A. Williams, C. Royer, C. Chow, E. Robles, B. Dalal, J. Chen, and S. Cousins. The bloodhound project: automating discovery of web usability issues using the infoscent simulator. In *Computer-Human Interaction 2003 Conference on Human Factors in Computing Systems (CHI)*, pages 505–512, 2003.

[6] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 462–471, New York City, 2004.

[7] P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: Context-driven automatic semantic annotation with C-PANKOW. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, pages 332–341, Chiba, Japan, 2005.

[8] P. DiMattia, F. X. Curran, and J. Gips. *An Eye Control Teaching Device for Students without Language Expressive Capacity – EagleEyes*. The Edwin Mellen Press, 2001. See also http://www.bc.edu/eagleeyes.

[9] R. O. Duda, R. E. Hart, and D. G. Stork. *Pattern Classification, Second Edition*. John Wiley & Sons, New York, 2001.

[10] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, pages 641–650, Chiba, Japan, 2005.

[11] S. Gupta and G. Kaiser. Extracting content from accessible web pages. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, pages 26–30, Chiba, Japan, 2005.

[12] S. Gupta, G. Kaiser, and S. Stolfo. Extracting context to improve accuracy for html content extraction. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, pages 1114–1115, Chiba, Japan, 2005.

[13] S. Harper, C. Goble, and R. Stevens. Traversing the web: Mobility heuristics for visually impaired surfers. In *4th International Conference on Web Information Systems Engineering (WISE 2003)*, pages 200–208, Rome, Italy, Dec. 2003. IEEE Computer Society.

[14] S. Harper, Y. Yesilada, C. Goble, and R. Stevens. How much is too much in a hypertext link?: investigating context and preview – a formative evaluation. In *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 116–125. ACM Press, 2004.

[15] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 1–10. ACM Press, 2003.

[16] K. Hornbæk, B. B. Bederson, and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Human-Computer Interaction*, 9(4):362–389, Dec. 2002.

[17] J. W. Kim, K. S. Candan, and M. E. Dőnderler. Topic segmentation of message hierarchies for indexing and navigation support. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, pages 322–331, Chiba, Japan, 2005.

[18] H. Larson and J. Gips. A web browser for people with quadriplegia. In C. Stephanidis, editor, *Universal Access in HCI: Inclusive Design in the Information Society*, pages 226–230. Lawrence Erlbaum Associates, Mahwah, NJ, 2003.

[19] Y. Lu, B. Zhang, W. Xi, Z. Chen, Y. Liu, M. Lyu, and W.-Y. Ma. The powerrank web link analysis algorithm. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 254–255, New York City, 2004.

[20] N. Milic-Frayling, R. Jones, K. Rodden, G. Smyth, A. Blackwell, and R. Sommerer. SmartBack: Supporting users in back navigation. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 63–71, New York City, 2004.

[21] A. Montogmery and C. Faloutsos. Indentifying web browsing trends and patterns. *Computer*, pages 94–95, 2001.

[22] M. Paquette, M. Betke, and J. Magee. IWeb Explorer: A web browser designed for use with an eye controlled mouse device. In *Boston University Computer Science MA Thesis Report*, 2005.

[23] B. Parmanto, R. Ferrydiansyah, A. Saptono, L. Song, I. W. Sugiantara, and S. Hackett. AcceSS: Accessibility through simplification & summarization. In *Proceedings of the Second International Cross-Disciplinary Workshop on Web Accessibility (W4A2005)*, pages 18–25, Chiba, Japan, 2005.

[24] J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, Sept. 2002.

[25] P. Plessers, S. Casteleyn, Y. Yesilada, O. D. Troyer, R. Stevens, S. Harper, and C. Goble. Accessibility: a web engineering approach. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 353–362. ACM Press, 2005.

[26] I. V. Ramakrishnan, A. Stent, and G. Yang. Hearsay: enabling audio browsing on hypertext content. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 80–89, New York, NY, USA, 2004. ACM Press.

[27] D. Reis, P. B. Golgher, A. S. da Silva, and A. H. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 502–511, New York City, 2004.

[28] J. T. Richards and V. L. Hanson. Web accessibility: a broader view. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 72–79. ACM Press, 2004.

[29] D. C. Robbins, E. Cutrell, R. Sarin, and E. Horvitz. Zonezoom: map navigation for smartphones with recursive view segmentation. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 231–234. ACM Press, 2004.

[30] L. Shih and D. Karger. Using URLs and table layout for web classification tasks. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 193–202, New York City, 2004.

[31] C. Stephanidis, A. Paramythis, C. Karagiannidis, and A. Savidis. Supporting interface adaptation in the AVANTI web browser. In *Procedings of 3rd ERCIM Workshop on User Interfaces for All*, Alsace, France, Nov. 1997.

[32] T. Sullivan and R. Matson. Barriers to use: Usability and content accessibility on the web's most popular sites. In *Proceedings of the 2000 Conference on Universal Usability*, pages 139–144, Arlington, Virginia, USA, 2000.

[33] T. Winograd. Architectures for context. *Human-Computer Interaction*, 10(24):401–419, 2001.

[34] L. Wood, A. L. Hors,V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, G. Nicol, J. Robie, R. Sutor, and C. Wilson, editors. Document Object Model (DOM) Level 1 Specificiation Version 1.0, W3C Recommendation, http://www.w3.org/TR/REC-DOM-Level-1, 1998.

[35] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305, Washington, D.C., USA, 2003.

[36] X. Yin and W. S. Lee. Using link analysis to improve layout on mobile devices. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 338–344, New York City, 2004.