A Genre-based Clustering Approach to Content Extraction

Suhit Gupta
Columbia University
500 W. 120th Street
New York, NY 10027
001-212-939-7184
suhit@cs.columbia.edu

Hila Becker
Columbia University
500 W. 120th Street
New York, NY 10027
001-212-939-7100
hb2143@cs.columbia.edu

Gail Kaiser
Columbia University
500 W. 120th Street
New York, NY 10027
001-212-939-7081
kaiser@cs.columbia.edu

Salvatore Stolfo
Columbia University
500 W. 120th Street
New York, NY 10027
001-212-939-7080
sal@cs.columbia.edu

ABSTRACT

The *content* of a webpage is usually contained within a small body of text and images, or perhaps several articles on the same page; however, the content may be lost in the clutter (defined as cosmetic features such as animations, menus, sidebars, obtrusive banners). Automatic content extraction has many applications, including browsing on small cell phone and PDA screens, speech rendering for the visually impaired, and reducing noise for information retrieval systems. We have developed a framework, Crunch, which employs various heuristics for content extraction in the form of filters applied to the webpage's DOM tree; the filters aim to prune or transform the clutter, leaving only the content. Crunch allows users to tune what we call "settings", consisting of thresholds for applying a particular filter and/or for toggling a filter on/off, because the HTML components that characterize clutter can vary significantly from website to website. However, we have found that the same settings tend to work well across different websites of the same *genre*, e.g., news or shopping, since the designers often employ similar page layouts. In particular, Crunch could obtain the settings for a previously unknown website by automatically classifying it as sufficiently similar to a cluster of known websites with previously adjusted settings. We present our approach to clustering a large corpus of websites into genres, using their pre-extraction textual material augmented by the *snippets* generated by searching for the website's domain name in web search engines. Including these snippets increases the frequency of function words needed for clustering. We use existing Manhattan distance measure and hierarchical clustering techniques, with some modifications, to pre-classify the corpus into genres offline. Our method does not require prior knowledge of the set of genres that websites fit into, but to be useful *a priori* settings must be available for some member of each cluster or a nearby cluster (otherwise defaults are used). Crunch classifies newly encountered websites online in linear-time, and then applies the corresponding filter settings, with no noticeable delay added by our content-extracting web proxy.

Categories and Subject Descriptors

I.7.4 [**Document and Text Processing**]: Electronic Publishing; H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Web-based Services*

WWW 2006, May 22-26, 2006, Edinburgh, UK.

General Terms

Human Factors, Algorithms, Standardization.

Keywords

Website classification, clustering, content extraction, reformatting, HTML, context, accessibility, speech rendering.

1. INTRODUCTION

Webpages are often cluttered with extraneous materials, perhaps attempting to attract the user's attention or improve the user's efficiency, but they may end up *distracting* the user from the actual content. These "features" may include script and flash-driven animations, other kinds of images not directly associated with a main text body, menus and guides, links scattered around the screen, etc. The automatic extraction of heuristically-defined "content" from webpages has many applications, including enabling end-users to access the web more easily over constrained devices, providing better access to the web for the blind or otherwise disabled, producing less noisy data for information retrieval and content summarization algorithms, and so on.

We have developed a framework, Crunch [4] [5] [6], as a web proxy that employs various heuristics in the form of filters and filter "settings" to achieve content extraction via clutter reduction. Crunch passes each webpage through an HTML parser, which corrects the markup and creates a Document Object Model (DOM) tree. DOM (www.w3.org/DOM) is a standard for creating and manipulating in-memory representations of HTML (and XML) content. Crunch applies the filters to the DOM tree, according to their settings, and the resulting HTML page provided to the user client is a (relatively) clean, clutter-free page.

Crunch allows end-users or administrators to tune the settings, essentially the thresholds for applying each filter, via aggressiveness sliders and on/off checkboxes. While Crunch works extremely well on a large variety of webpages using the default settings, these settings sometimes should be manually configured to best extract content for a given site. We found that, in practice, the settings need to be adjusted only when a user moves from one major class of website to another, e.g., from news to shopping or *vice versa*. For instance, it may be appropriate to remove what (heuristically) appears to be advertising from a news webpage, but if the same is done on a shopping webpage, there may be little or nothing left!

Specifically, the link density and table/cell structure layout remained consistent among sites within news *vs.* shopping genre (and other genres discussed below).

In order to reduce human involvement in selecting the heuristic filter settings, we consider utilizing a website's *genre* classification. Crunch can then obtain some previously (manually) adjusted settings for a newly visited website by automatically classifying it as sufficiently similar to a genre-cluster of known websites, at least one of which has "known good settings" – which, we found empirically, produces better content extraction results than *any* possible one-size-fits-all default settings.

With a good webpage-oriented document clustering method, Crunch or other web applications can automatically organize a corpus into a meaningful genre hierarchy. For example, another application might be to enable efficient navigation of the corpus by genre [17]. However, performing genre analysis in real-time (online) is too computationally expensive for most web applications. Additionally, most current clustering algorithms require *a priori* knowledge of the number or specific set of clusters to properly classify a set of webpages - but given the vast quantity and enormous variety of documents posted on the web, this does not seem the best model for open-ended applications.

Our goal was to find a simple acceptable-cost *offline* algorithm for pre-clustering a large corpus of websites, which also enabled efficient *online* classification of new websites as they were encountered by Crunch users. Our approach utilizes the text results (called "snippets") generated by sending the website's domain name to several popular search engines and using those snippets together with the website's own actual text towards determining the genre (cluster) of the website. We found that exploiting snippets not only increased the frequency of function words, but those function words were, in general, highly descriptive of the "meaning" of the websites being accessed and thus especially useful in the analysis of the appropriate genre.

While we could have, in principle, summarized each webpage directly using, e.g., NLP techniques, that would have enormously increased the complexity of our own algorithms. Further, utilizing snippets means that we are *re-using* structured data that has already been conveniently compiled by search engines in advance. Crunch then uses standard techniques like Manhattan distance measure and hierarchical clustering, with some modifications, to pre-classify websites into genres. Our clustering method does not require any prior knowledge of the set of genres that websites might fit into, but instead discovers these relationships among websites. Subsequently, Crunch is able to classify newly encountered websites in linear-time, and then apply the corresponding filter settings (adopted from the closet member of the genre cluster), with no noticeable delay introduced.

In this paper, we present our method for pre-clustering a corpus of websites, and describe the ensuing classification of individual new websites and its application to selecting heuristic filter settings by apparent genre. We also show experimental results that demonstrate the substantial improvement of our content extraction system, as it is bolstered by our efficient approach to genre-based clustering of websites and re-use of previously adjusted settings.

The following sections describe a selection of the vast related work in the fields of information retrieval and document clustering, followed by details of our approach and the associated implementation. We conclude with empirical results from our experiments and a summary of our contributions.

2. RELATED WORK

Document clustering has been investigated for many domains [3] [16].The use of clustering in IR appears mostly to be driven by the cluster hypothesis, which states that "closely associated documents tend to be related to the same requests" [1]. Xu et al. [17] explain that document clustering methods can be categorized into two main types: document partitioning (flat) and agglomerative (bottom-up hierarchical) clustering. Although both types of methods have been extensively investigated for decades, accurately clustering documents without domain-dependent background information is still a challenging task. They provide a novel document partitioning method using non-negative factorization of the term-document matrix. However, we have found that web document clustering can be done with little or no background information about the domains, as long as there are enough function words available to help classify the documents.

There is a large body of related work in genre classification. Lee et al. [7] describe their method for text genre classification by using two different class sets, genre classes and subject classes, in the training data. However, their method would not work well for web documents since the number of genres needs to be identified and fixed before the categorization. This, as pointed out before, is an impediment for the vast number and incredible variety of the corpus of web documents. For the same reason, traditional partitioning methods like K-Means clustering [9] will not work. Similarly, clustering using sequential information maximization, presented by Slonim et al. [14], requires prior knowledge of the number of clusters for the data to be classified.

Spectral clustering, an approach demonstrated by Ng. et al. [10], clusters points using eigenvectors of matrices derived from the data contained in the documents. However, as with K-Mean clustering, the number of clusters is expected to be a known quantity before the process. Additionally, spectral clustering works hard at performing tight fitting of all data points within a cluster, but later experiments show that this is not a hard requirement when classifying web documents by genre. Further, we have observed that a relatively small number of documents (as few as two, as demonstrated later in this paper) are enough to define a cluster and its associated genre.

Cutting et al. [3] describe a cluster-based approach to browsing large document collections, called Scatter/Gather. They provide a mechanism for user-driven organization of data in a fixed number of clusters, but the users need to be in the loop and the computed clusters do not guarantee accuracy. Moreover, the technique is designed to work well for finding similarity between articles based on subject. We are interested in identifying broader topic genres, like news, shopping and sports, for top-level domains.

Zhang et al. [21] present BIRCH, a clustering method for extremely large databases. The main optimizations in their technique are made to reduce the number of I/O operations and to increase space efficiency. In order to truly gain the benefits from this model, multiple passes on the data are required. However, this increases the runtime complexity of the system.

Siersdorfer et al. [13] show an interesting approach, called restrictive clustering, to clustering by working on only a subset of the available data. The key element to their approach is to construct restrictive meta-methods at the moderate cost loss of uncertain samples. Their approach clusters with high accuracy, but is prone to miss recognizing failed clustering attempts.

Zamir et al. [20] describe Grouper, a clustering system that employs term-based clustering, an approach similar to ours. They use Suffix Tree Clustering, which is shown to be fast and domain independent. However, they have applied their methods only to grouping the results returned by a search engine by analyzing the produced snippets. We instead aim to automatically classify whole websites, leveraging the snippets for additional function words.

Lastly, Crammer et al. [2] describe a technique for performing classification that focuses on online additive algorithms for classification tasks. Their approach was designed to save a user from needing vast computational resources. We tend not to suffer from this problem either since we define a fixed range of words (described in Sections 3 and 4) to classify the various websites. Additionally, the system described needs a training period to produce the level of classification achieved, while even with a static set of pre-clustered data we can achieve similar results.

Finally, most of the work described here tends to be used towards analysis of documents or search engine results, but not targeted to webpages. One general problem with applying these strategies to webpages is that most pages contain extremely noisy data that may lead to incorrect clustering results. In Crunch, newly seen websites are necessarily clustered *prior* to content extraction, so the clustering approach must be reasonably resilient to such noise.

3. APPROACH

Clustering involves the grouping of similar objects, and has been practiced, consciously or unconsciously, for many thousands of years [16]. Distance coefficients, such as Euclidean distance, have been used very extensively in cluster analysis, owing to their simple geometric interpretation. We employ multiple techniques that incorporate the advantages of the previous work on clustering and information retrieval. However, in order to motivate our clustering into genres, we briefly describe our application.

3.1 Crunch

Extraction of "useful and relevant" content from webpages has many applications, including constrained screen browsing, speech rendering for the visually impaired, and reducing noise for NLP and IR systems. Our initial insight was to work with DOM trees, rather than raw HTML markup [4] [5] [6]. Crunch provides an extensible set of tunable heuristic filters for clutter reduction, and consequent content extraction, via a web-proxy architecture. The filter settings should be tuned, automatically or by the user (e.g., using the console shown in Figure 1), to the site being browsed and the content desired to see. Figure 2 shows an example of a typical CNN page without and with tuned filters applied.

Figure 1 - Crunch Control Panel

One of our goals is to *automate* the selection of filter settings for arbitrary websites by classifying each newly visited website according to genre, and then re-using previously configured settings known to work well for that genre (or a nearby genre if there are no known good settings for that genre, falling back to proxy instance-specific defaults in the worst case of no sufficiently close genre with known good settings). We have found that certain settings work very well for websites in a given genre, but sometimes poorly for other genres. This is an observed phenomenon, subject to change as website design techniques advance, but at present it appears that web designers working within the same genre tend to prescribe similar screen layouts using the same HTML elements to present those layouts. For instance, the news sites in our sample corpus share the same manually configured "best" filter settings, but those settings did not work very well for shopping and sports. Therefore, we hypothesized that detecting the webpage's genre would enable better content extraction than any "one size fits all" defaults.

Since this is an online web application, content genre identification must be done in near real-time. Assuming the content extraction proxy already has data on the existence of various genre clusters (and the corresponding filter settings that work well for those genres), then matching individual webpages to those clusters and applying the appropriate settings can be an

efficient process. Therefore, we focused on the offline identification of genre clusters from a large corpus of websites.

Figure 2 - Content Extraction results on a typical cnn.com article (original page vs. extracted page)

3.2 Clustering

We found relatively few existing classification algorithms apply directly to the clustering of web documents (see Section 2). These few algorithms, however, generally suffer from one or both of two basic problems: (i) they had high run-time complexity (cubic or higher running time), and/or (ii) they classified webpages into a fixed number of pre-determined genres rather than finding the affinity among websites and thus discover where the clusters lay. Our goal was to avoid these two limitations of previous work.

We use an external module as a preprocessor to pre-classify some wide corpus of websites, in our experiments covering genres like news (international and regional), shopping, astronomy and technical weblogs. Our key insight was to leverage the results returned by search engines when searching for each website's domain name, specifically the text "snippets" attached by the search engine to each result. We used only those snippets on the first page of the results, as likely to be most relevant, but employed several distinct search engines so as not to be overly biased by any particular engine's ranking algorithm. We found that these snippets contain words that are highly descriptive of the function of the corresponding websites. Furthermore, leveraging snippet data does not increase the complexity of our clustering algorithm, but instead simply adds the aggregate access/wait time for the search engines to the overall running time.

For each website slated for clustering, we create a word frequency map based on the textual data of the (front) webpage as well as the snippets produced by six popular search engines (Google, Yahoo, Dogpile, MSN, Altavista and Excite). For instance, searching for cnn or nypost increased the frequency of words such as "news" and "business", while searching for Amazon resulted in a frequency increase in the words "shop" and "books". From the frequency map of each document, we prune all words deemed insignificant, using a stop word list consisting of prepositions, articles, pronouns, etc., and also remove other words that may appear frequently in a document but that do not add information to the genre of the site, e.g., non-dictionary words. Our dictionary contains 23,000 words and their variations, including some common prefixes, suffixes and tenses. From the frequency maps, frequent (greater than 10 occurrences) and unique words are added to a *Word Key* vector, if they were not already added while processing a previous website. We found that each site added, on an average, six new words to the *Word Key*.

The frequency maps are then re-graphed across this *Word Key* to produce a content genre *identifier* for each of the websites (example graphs are shown in Figures 3 and 4). The next step is to find the distance of each *identifier* from all the other identifiers in the corpus. It has been pointed out that a major limitation of the Euclidean distance in the information retrieval context is that it can lead to two documents being regarded as highly similar to each other, despite the fact that they share no terms at all in common [16]. However, we found that the addition of snippet data, and the consequent increase in the frequency of genre-specific function words, effectively avoided this problem.

With all the pair-wise distances in place, the pairs are sorted from closest association to furthest. Here we employ the hierarchical clustering algorithm [16], with a slight variation (as described in the next section), to perform clustering. The clusters are then optionally (manually) tagged by the appropriate genre name. The "best" heuristic filter settings can now be associated manually with each genre, or previously saved settings configured by some administrator or user can be used for each genre, potentially individually for up to every website in a genre cluster, using some

algorithm for deconflicting competing (different) settings; further discussion of such mechanisms is outside the scope of this paper.

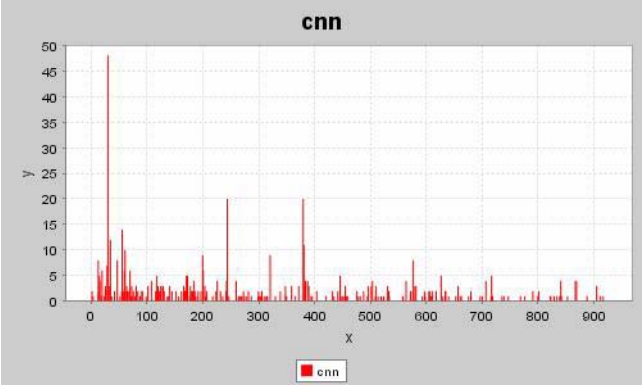

Figure 3 - Graph of CNN.com across the *Word Key*

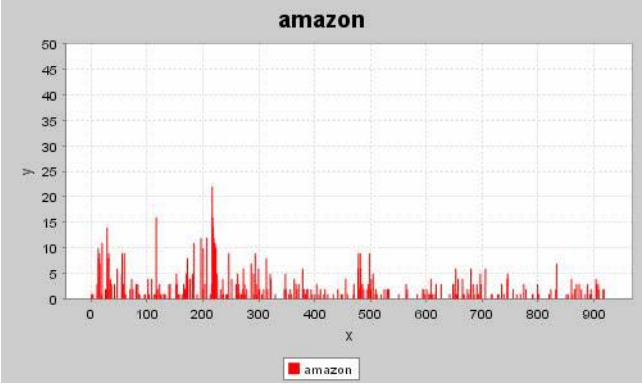

Figure 4 - Graph of Amazon.com across the *Word Key*

At run-time, when a webpage from a previously unclassified site is encountered, Crunch employs a similar process: the text from the webpage and its website's snippet data is aggregated, cleaned and graphed across the *Word Key*. Then using the same distance measurement model, Crunch finds a closest match in some pre-classified cluster. The associated filter settings are loaded and Crunch extracts the content for that webpage. (In some cases, as explained in Section 5.1, there may not be any sufficiently close cluster according to a parameterizable threshold; in that hopefully rare situation, the proxy's default settings are used.)

4. IMPLEMENTATION

We use the *Manhattan histogram distance measure algorithm* to measure the distance between the *Word Key* graphs for each pair of websites during pre-classification, and for the newly visited webpage vs. those pre-classified websites at run-time. The formula is defined as

$$D_1(h_1, h_2) = \sum_{i=0}^{n-1} | h_1[i] - h_2[i] |$$

The histogram (h_1 , h_2) is represented as a vector, where n is the number of bins in the histogram (i.e., the number of words in our Key Word Set). h_1 and h_2 must first be normalized in order to satisfy the distance function requirements. Crunch uses the settings associated with the pre-classified site whose distance is closest to the webpage being accessed (breaking ties arbitrarily).

During the preprocessing stage, calculation of the Manhattan distance for every site from every other site is an $O(n^2)$ operation. After the distances between every pair of sites has been determined, the distances are sorted in time $O(n \log n)$. We can think of this computation as producing a complete graph on n vertices, where each vertex corresponds to a website and the value of the edge between any two vertices corresponds to the distance between the websites. Clusters are formed on this graph based on the assumption that websites with similar layouts contain similar words and, a key assumption for content extraction purposes, that similar HTML elements are used to present the clutter desirable to be pruned. We do not force a specific number of clusters, but rather let clusters form naturally using a hierarchical clustering algorithm that runs in time $O(n^2)$.

Hierarchical clustering has been shown to be an effective method for clustering webpages. We briefly describe this clustering method here: At each iteration, the algorithm examines the next closest pair of vertices. One of several situations may occur:
- If neither vertex belongs to any existing cluster, they form a new cluster.
- If both vertices are assigned to different clusters, examine their distance and merge the clusters if within a predefined threshold.
- If both vertices are already assigned to the same cluster, continue.
- If vertex 'a' is assigned to cluster 'i' and vertex 'b' is not assigned to any cluster, check the level of vertex 'a' in the hierarchy with respect to cluster 'i'. If it is either on the first or second level, insert vertex b into cluster 'i', otherwise continue.

Thus, root nodes of a cluster may pull additional sites into that cluster, and any site that was pulled in directly by the root may also pull in additional sites. However, any site that was not clustered directly by the root of that cluster may not pull in any other sites, even if the algorithm encounters a site not yet clustered. This restriction is imposed to prevent chaining, ultimately preventing all the clusters from merging into one gigantic one. We found that sites further than one link from the root of a given cluster are often far enough away to be potentially closer to a different cluster. In many cases, sites that were initially rejected are still pulled into the cluster, as the algorithm proceeds, by a site that is directly connected to the root.

The threshold for the hierarchy level within clusters as well as the threshold for merging clusters was determined empirically. This was necessary since these thresholds are a consequence of the range of distances and the correlation among the different sites as it is reflected in our distance measurement. The algorithm halts when the distance between the next pair of sites exceeds the preset threshold or when all possible pairs of sites have been examined. The thresholds are used in order to prevent extremely unrelated sites from contaminating existing clusters.

The overall complexity of the offline clustering module is $O(n^2)$. However, when the genre of a previously unvisited website needs

to be determined at run-time, the complexity drops to O(n): the time to create the frequency graph is a constant, and then it needs to be distance-matched to the n different pre-clustered sites to find the closest cluster to which to add this website.

5. EXPERIMENTS & RESULTS

5.1 Clustering

We chose 171 unique websites as the corpus on which to test our implementation. The sites chosen were those visited by our group on a regular basis (admittedly a limited sample). Table 1 shows results consistent with our primary hypothesis – that using search engine snippets towards clustering produces better and tighter results. Table 2 summarizes the top six clusters produced by our clustering system. The search engines used to extract snippets were Google, Yahoo, Dogpile, MSN, Altavista and Excite.

	Using Snippets	Without Snippets
No. of Sites	171	171
No. Search Engines	6	0
No. Clusters Found	14	5
Max. Cluster Size	71	159
Min. Cluster Size	2	2
Avg. Cluster Size	12.21	34.2
Time to cluster (min)	25	11

Table 1 - Cluster Results

In Table 1, we see that the number of clusters found when using snippets far exceeds those from the system without using snippets. While the run-time more than doubled, the increase was due only to the access time and data gathering from the six more sites per website being clustered (i.e., accessing the search engines). Upon manual inspection, we found that the sites clustered by the first experiment, i.e., the one using snippets, categorized sites more accurately: When we manually clustered the same websites into genres, we came up with 16 distinct clusters; using snippets identified 87.5% of those clusters while only 31% of those clusters were identified when snippets were not used.

	Cluster Type	# of Sites (w/snippets)	# of Sites (no snippets)	# of Sites (manual)
Cluster #1	International News	71	159	65
Cluster #2	Shopping	27	-	25
Cluster #3	Regional News	10	-	13
Cluster #4	Tech News	7	-	10
Cluster #5	Tech Blogs	7	-	8
Cluster #6	Astronomy	7	6	7

Table 2 – Typical Cluster Examples

Table 2 summarizes some of the clusters that were created by our preferred approach, as well as the comparable approach without using snippets and our (subjective) manual clustering. International news sites were the most common websites in the corpus used in our experiments, and those made their way into one cluster. Other notable genres included shopping, regional news (from India), and astronomy sites. An interesting observation regards the separation of technical news sites vs. technical blogs. They apparently clustered into separate genres because blogs tend to be updated more often than general tech news sites, so the topics were different (and more timely in the blog case). Overall, using essentially the same algorithms but without snippets produces results that are far worse than the approach with snippets.

Figures 5-6 show some graphs demonstrating the general similarity (wrt our *Word Key* classifications) of various websites that were determined to be part of a cluster. Figure 5 depicts some international news websites (the news genre), including cnn.com, drudgereport.com, washingtontimes.com and chicagotribune.com. For such sites, Crunch's heuristic settings should be far more text oriented and thus should aggressively remove links and advertisements. Figure 6 shows popular shopping sites (the shopping genre): amazon.com, ebay.com, pricewatch.com and streetprices.com. In these cases, the settings should be more accepting of link-heavy elements and HTML forms.

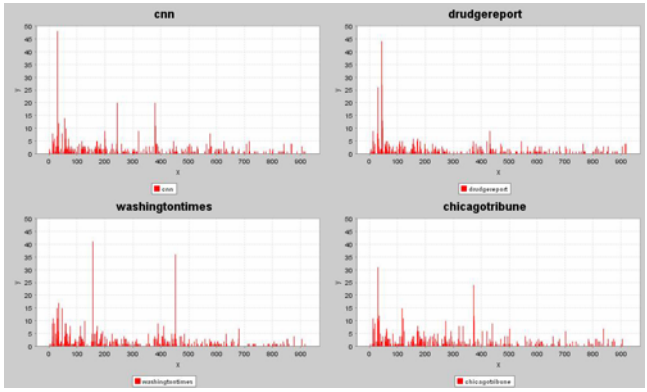

Figure 5 - Cluster containing CNN and other news sites

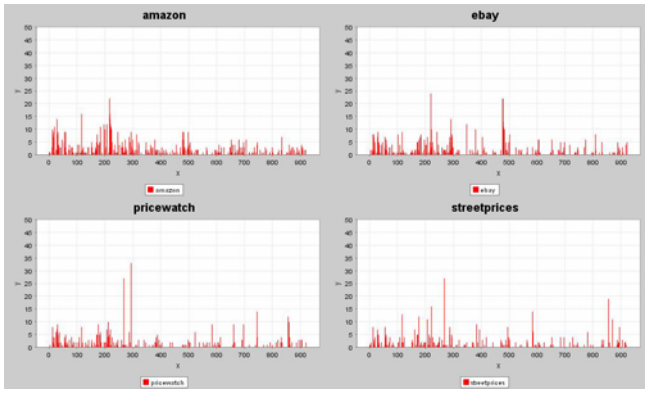

Figure 6 - Cluster containing Amazon and other shopping sites

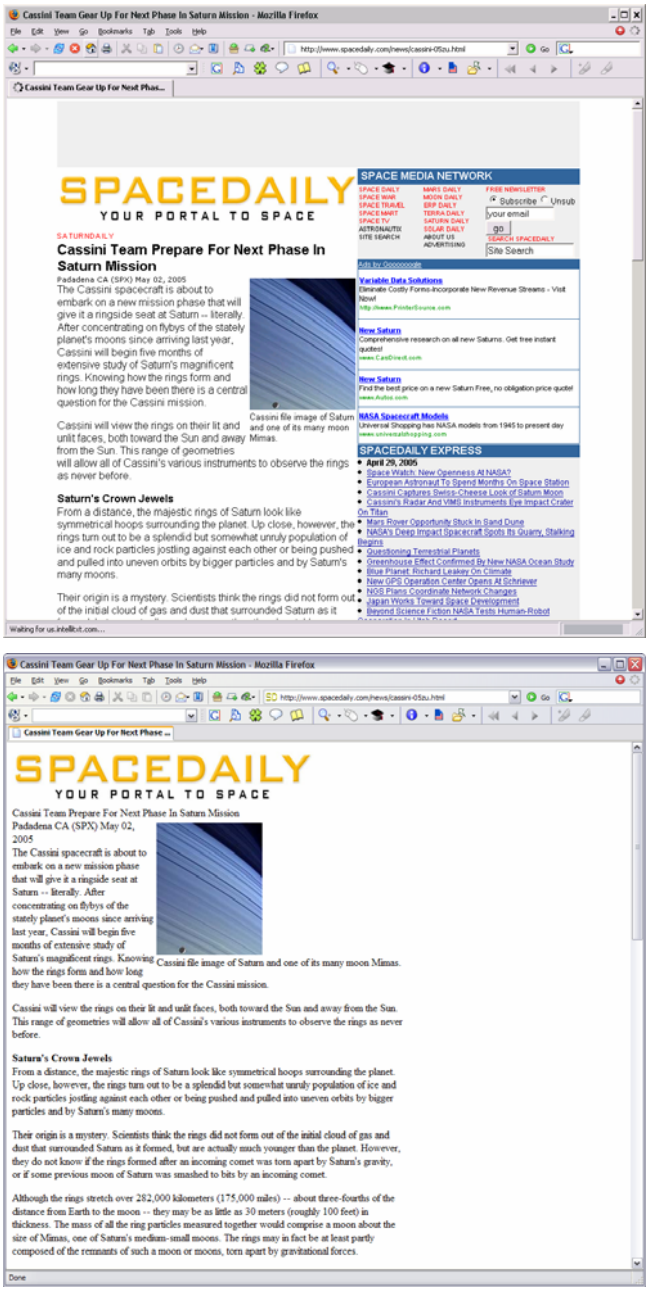

Figure 7 – Before and after extraction of content for Spacer.com, with CNN-like news settings

Figure 7 shows another example news site, spacer.com. We see the original webpage and the content-extracted webpage when news settings tuned for cnn.com are loaded, since spacer was identified as a site in the same cluster as cnn. We observed similar results for other genres such as shopping, for example, Amazon.com's settings were successfully used on eBay.com.

While our approach to website genre classification is able to cluster similar sites, it is also able to clearly distinguish singular sites that do not fit into any cluster, both in the pre-classification stage as well as in the subsequent genre matching phase. Skinheadz.com is one such a site, and its frequency graph along the *Word Key* is shown in Figure 8. Similarly, sec.gov, whose frequency graph is shown in Figure 9, is a site that remained unmatched to the identified genres in our corpus. While both of these may be exemplars of certain genres, those particular genres did not occur in our sample corpus. A broader range of website pre-classification awaits future work

Our algorithm was also successfully able to cluster websites whose frequency graphs look visually different but whose genre is similar. Examples are shown in Figures 9 and 10 in the Appendix.

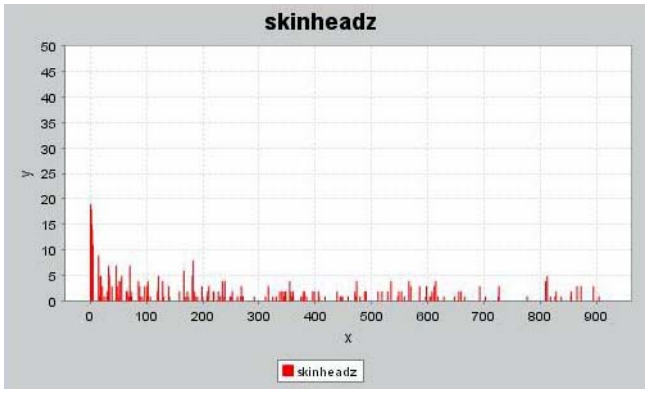

Figure 8 - Graph of skinheadz.com across the *Word Key*

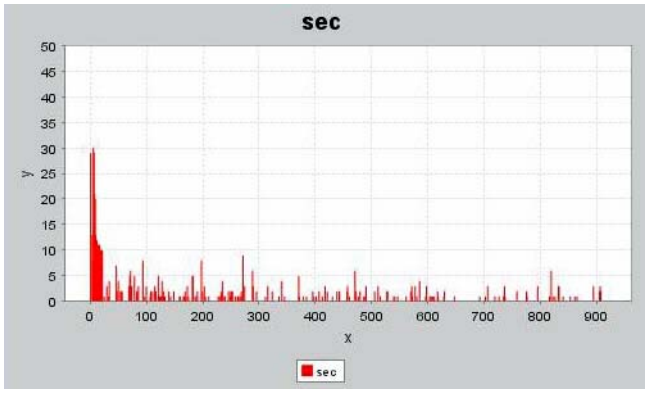

Figure 9 - Graph of sec.gov across the *Word Key*

We have also found that using snippets aids resiliency to changes in the structure of clusters over time. Even though the content on the various websites may change, the snippets tend to remain nearly the same. Therefore, the frequency of the most frequent words remains similar over time causing little fluctuation.

5.2 Speech rendering experiment

To evaluate the effectiveness of our clustering approach and the subsequent automatic application of genre-based filter settings, we conducted experiments to test Crunch's benefits in two areas where it could be particularly useful – content extraction that facilitates speech rendering for the visually impaired, and webpage rendering on devices with constrained resources.

The goal of our first experiment was to consider the "readability", by conventional screen reader software, of the extractions produced by Crunch compared to the raw webpages. We

measured the length of time, in seconds, that it took the screen reader to render (speak) each variant of a given page, considering a variety of webpages. We ran trials with both the "free demo" version of JAWS (www.freedomscientific.com) and a licensed copy of Home Page Reader (www.ibm.com), which vary slightly in how they preprocess the raw HTML – although the measured time differences were under 3 seconds in all cases, so we average the results in our reporting below. The idea was to determine the amount of time a visually disabled user might save by using our content extraction technology. The notion of "content" is inherently subjective and our determination of what is the content vs. non-content was performed by visual and auditory inspection.

For this experiment, we chose 11 websites that represent a variety of layout formats. We included websites from all the major genres that appeared in our corpus (e.g., news, shopping, tech news, astronomy), but were also careful to cover different structures (columns, single-body articles, portal-based and blog-style sites), as well as W3C-compliant "accessible" sites vs. non-compliant sites. We passed the original webpages vs. the Crunch outputs through the screen readers. We then measured the time it took, in minutes, to read (speech render) the entire webpages.

Table 3 – Speech rendering results

Site (accessed on May 18th, 2005)	Read original webpage (minutes)	Read page produced by Crunch genre-based settings (minutes)
CNN.com front page	10 :09	1 :08
CNN.com subsidiary page	7:35	2:44
Slashdot.org front page	25:20	17:53
Slashdot.org article page	14 :13	6 :15
MSNBC front page	10 :47	2 :40
MSNBC article page	11:12	3:43
Yahoo News front page	25:15	16:39
Yahoo News article page	14:08	5:13
NASA Ames front page	2 :18	1 :48
NASA Ames Research page	1:57	1:17
Amazon.com front page	13:28	7:42

From these tests and from the anecdotal accounts of visually impaired users (e.g., attendees at the 2005 W4A meeting), it is clear that blind web users typically spend tens of minutes listening to nearly any single webpage using a commercial screen reader alone - and this is absolutely unacceptable! We found that using Crunch together with such a screen reader reduces by 10-80% the time spent in reading the page while the content on the webpage remains qualitatively accurate. The least significant improvement (< 10% speedup in reading) using Crunch was on the main page of a given site, where the settings preserved a larger percentage of navigation links (Crunch's heuristics distinguish between front and auxiliary pages since front pages are often intended to operate as portals). The greatest improvement noticed was on subsidiary pages of websites, usually containing contentful articles.

On a side note: one of the Crunch heuristic plug-ins detects the front page vs. a subsidiary page of a website based on URL analysis and (sometimes) web browser error messages, and differentiates the website settings accordingly. We added this capability to Crunch because the kinds of HTML elements that should be treated as "useful and relevant" content vs. clutter tends to change depending on whether we are looking at the front page of a site or some subsidiary page. For example, the article is typically the main focus of the content on a subsidiary page of a news site; however, the main page of that same news site often has tiny bits of detail about several news stories as well as numerous links that lead the user to other webpages on the site.

A final point is that our trials on a website fully compliant with the W3C's Web Accessibility Initiative accessibility guidelines, NASA Ames Research, still show an improvement in reading time - albeit small. Thus we believe that Crunch, and its genre analysis presented in this paper, are valuable tools for content extraction even on websites that are compatible with the WAI accessibility guidelines. However, further research is needed in this area.

5.3 Constrained screen testing

We also evaluated how well Crunch compared to other content extraction and webpage reformatting technologies designed for devices with limited screen real-estate. We used the same samples as for the speech rendering tests above, and displayed both the original webpages and the pages output by Crunch on various combinations of handheld devices and browsers. We tested the system on the Toshiba e805 and HP iPaq 2215 PDAs running Microsoft's PocketPC OS, with Pocket Internet Explorer and BitStream's Thunderbird browsers, respectively. We measured the amount of content on the first screenful at both 320x240 and 640x480 resolutions. We also used a Blackberry 7100t running a proprietary Blackberry browser and a Microsoft Smartphone i600 running Internet Explorer and Opera Mobile Browser.

The purpose of these tests was to demonstrate the increase in "relevant and useful" content displayed on a small screen when using Crunch vs. not using Crunch. We would like to reiterate that, in the general-purpose case absent any model of the author's or reader's intents, content is subjective. For this experiment, we define content as the number of relevant words (whether displayed in text or images) shown on the screen, measured by visual inspection.

Table 4 - Constrained device testing results

Number of words (PDA 320x240)		Number of words (PDA 640x480)		Number of words (Blackberry)		Number of words (Opera on Smartphone)	
I	II	I	II	I	II	I	II
29	38	102	217	17	30	10	32
29	185	158	338	17	68	13	59
49	154	134	270	48	68	43	63
45	80	215	215	48	68	43	63
56	56	111	111	27	27	27	27
123	123	370	370	14	53	13	48
20	34	20	75	27	33	25	29
20	93	20	93	27	51	25	45
7	34	185	185	3	19	3	19

15	112	112	112	12	30	12	28
28	35	247	247	12	34	43	43

I – without Crunch, i.e., original webpage.
II – with Crunch, i.e., the page is passed through Crunch, with automatic genre-based settings.

From the data presented in Table 4, we see that Crunch with genre-based automatic selection of filter settings is very useful towards maximizing the amount of content displayed on constrained devices. The most significant difference was on a 320x240 resolution PDA screen, where there was on average a 215% increase in the amount of content displayed on the screen. This increase jumped dramatically up to a 750% when considering only news articles. With 640x480 resolution, we found an average increase of 133% of content on the first screenful. Several of the pages tested were able to fully render within that screenful. When testing with the cell phone browsers, we found the results to be almost identical, almost 185% improvement in both cases, presumably because of the very similar screen sizes. The main difference was due to the Opera browser's default behavior of jumping to the "middle" of the page where it found the largest concentration of text attempting to skip over anticipated non-content, which was lacking on the Blackberry. In none of our trials did a webpage rendered using Crunch display less content on the first screenful than the original page rendered on the same constrained device. However, the Opera comparison is somewhat problematic, not always counting the same words, due to Opera's skip-to-the-middle heuristic.

We also tested the sample websites using ZoomText, a leading low-vision screen magnifier. We found that with their standard magnification, the number of words displayed in a conventional web browser on a desktop running at 800x600 resolution was comparable - within 95-105% in both the with and without Crunch cases - to the number of words displayed on a 320x240 resolution PDA. Thus Crunch's improvements are applicable to those mildly to moderately visually impaired users who prefer screen magnifiers to speech rendering (although the two technologies can be used in combination with the same result).

6. SUMMARY OF CONTRIBUTIONS

In this paper, we consider the problem of clustering websites according to their genre, as applied to selecting the most appropriate amongst previously adjusted settings for an online content-extraction web application. Webpage classification is much more difficult than pure-text classification due to the noisy information embedded in webpages [12] - which reminds of our original motivation for content extraction. Utilizing snippets produced by search engine searches for the domain name of each website being classified, we are able to improve the frequency of the function words used to classify that site. With these snippets as well as the textual content on the site itself, we use existing simple and proven techniques - Manhattan distance and hierarchical clustering - to successfully pre-cluster a large number of websites in an efficient manner. This pre-clustering allows the system to classify individual new webpages not already classified in linear time, by comparing them to the existing clusters. Our content extraction web proxy, Crunch, uses this information to produce better results. In addition, our approach to identifying webpage genres may be beneficial to other applications unrelated to content extraction, e.g., to support browsing of search engine results by genre [7].

7. ACKNOWLEDGEMENTS

The Programming Systems Laboratory is funded in part by National Science Foundation grants CNS-0426623, CCR-0203876 and EIA-0202063, and in part by Microsoft Research. Part of the work reported in this paper was conducted in collaboration with the Columbia Intrusion Detection Systems lab, which has been supported by grants form NSF and HS ARPA.

8. REFERENCES

[1] Javed Aslam, Ekaterina Pelehov, Daniela Rus, "A Star Clustering Algorithm for Static and Dynamic Information Organization", Journal of Graph Algorithms and Applications, vol. 8, no. 1, 2004

[2] Koby Crammer, Jaz Kandola, Yoram Singer, "Online Classification on a Budget", Seventeenth Annual Conference on Neural Information Processing Systems, 2003

[3] Douglass Cutting, David Karger, Jan Pedersen, John Tukey, "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collection", Proceedings of SIGIR '92, 15th ACM International Conference on Research and Development in Information Retrieval, 1992

[4] Suhit Gupta, Gail Kaiser, David Neistadt, Peter Grimm, "DOM-based Content Extraction of HTML Documents", 12th International World Wide Web Conference, May 2003

[5] Suhit Gupta; Gail E Kaiser, Peter Grimm, Michael F Chiang, Justin Starren, "Automating Content Extraction of HTML Documents" Submitted to the World Wide Web Journal, January 2004

[6] Suhit Gupta, Gail Kaiser, "CRUNCH - Web-based Collaboration for Persons with Disabilities", W3C Web Accessibility Initiative, Teleconference on Making Collaboration Technologies Accessible for Persons with Disabilities, Apr 2003

[7] Yong-Bae Lee, Sung Hyon Myaeng, "Text Genre Classification with Genre-Revealing and Subject-Revealing Features", Proceedings of SIGIR '02, 25th ACM International Conference on Research and Development in Information Retrieval, 2002

[8] Tao Li, Sheng Ma, Mitsunori Ogihara, "Document Clustering via Adaptive Subspace Iteration", Proceedings of SIGIR '04, 27th ACM International Conference on Research and Development in Information Retrieval, 2004

[9] Tom Mitchell, "Machine Learning", McGraw-Hill Science/Engineering/Math, March, 1997

[10] Andrew Ng, Michael Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm", In Advances in Neural Information Processing Systems, 2001

[11] Fabrizio Sebastiani, "Text categorization", In Alessandro Zanasi (ed.), Text Mining and its Applications, WIT Press, Southampton, UK, 2005

[12] Dou Shen, Zheng Chen, Qiang Yang, Hua-Jun Zeng, Benyu Zhang, Yuchang Lu, Wei-Ying Ma, "Web-page Classification through Summarization", Proceedings of SIGIR '04, 27th ACM International Conference on Research and Development in Information Retrieval, 2004

[13] Stefan Siersdorfer, Sergej Sizov, "Restrictive Clustering and Metaclustering for Self-Organizing Document Collections", Proceedings of SIGIR '04, 27th ACM International Conference on Research and Development in Information Retrieval, 2004

[14] Noam Slonim, Nir Friedman, Naftali Tishby, "Unsupervised Document Classification using Sequential Information Maximization", Proceedings of SIGIR '02, 25th ACM International Conference on Research and Development in Information Retrieval, 2002

[15] C.J. van Rijsbergen, "Information Retrieval", Butterworths, London, 2nd ed., 1979

[16] Peter Willett, "Recent Trends in Hierarchic Document Clustering: A Critical Review", Journal Information Processing and Management, 1988

[17] Wei Xu, Xin Liu, Yihong Gong, " Document Clustering Based on Non-negative Matrix Factorization", Proceedings of SIGIR '03, 26th ACM International Conference on Research and Development in Information Retrieval, 2003

[18] Yiming Yang, Jian Zhang, Bryan Kisiel, "A scalability analysis of classifiers in text categorization", Proceedings of SIGIR '03, 26th ACM International Conference on Research and Development in Information Retrieval, 2003

[19] Oren Zamir, Oren Etzioni, "A Dynamic Clustering Interface to Web Search Results", Proceedings of Eighth World Wide Web Conference, 1999

[20] Oren Zamir, Oren Etzioni, "Web Document Clustering: A Feasibility Demonstration", Proceedings of SIGIR '98, 21st ACM International Conference on Research and Development in Information Retrieval, 1998

[21] Tian Zhang and Raghu Ramakrishnan and Miron Livny, "BIRCH: an efficient data clustering method for very large databases", ACM SIGMOD International Conference on Management of Data, 1996

9. APPENDIX

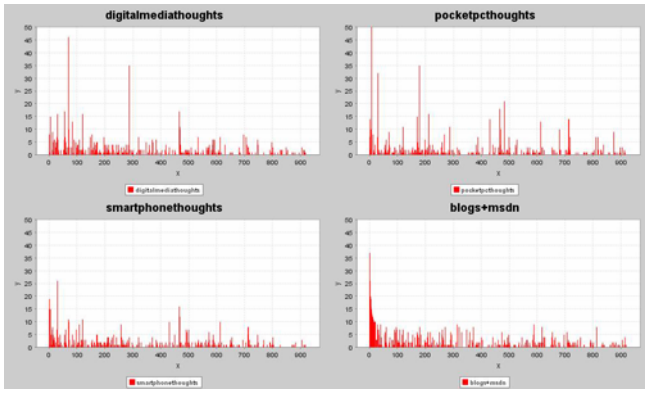

Figure 10 - Cluster containing tech blog sites including DigitalMediaThoughts, PocketPCThoughts, SmartphoneThoughts and MSDN Blog

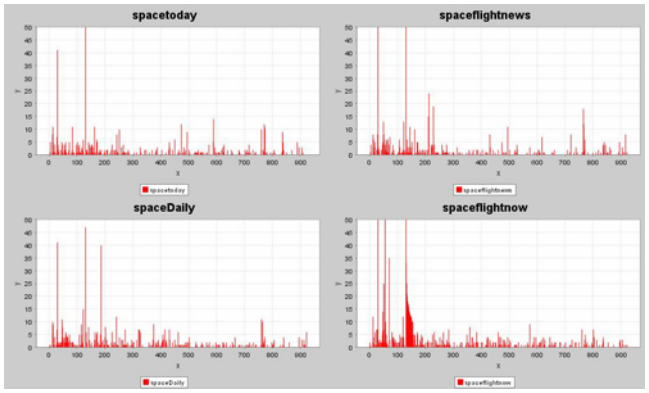

Figure 11 - Cluster containing several astronomy-related sites including spacetoday.com, spaceflightnews.com, spacedaily.com and spaceflightnow.com