

TopBlend: An Efficient Implementation of HtmlDiff in Java

Yih-Farn Chen, Fred Dougliis, Huale Huang, and Kiem-Phong Vo
AT&T Labs-Research
Florham Park, NJ USA
{chen,dougliis,huale,kpv}@research.att.com

Abstract: The World Wide Web is growing rapidly with new and changing web content. Detecting changes in web pages is crucial for website masters who care about website integrity. It is also convenient for web surfers who are constantly looking for new products, services, or information. This paper describes TopBlend, a new HTML differencing tool implemented in Java. TopBlend uses the fast Jacobson-Vo algorithm, which solves the Heaviest Common Subsequence problem, for page comparison. Performance results indicate that TopBlend significantly outperforms a previous HTML differencing tool in most time-consuming jobs, often by 1-2 orders of magnitude. TopBlend allows comparisons to be performed either on the server or client side, and can present the results in either a merged HTML view or a more convenient side-by-side view for web pages with complex graphics designs.

Introduction

Viewing the differences between web pages can be useful in many contexts. A web site maintainer might compare the new version of a page with the one available on the public site, in order to identify what has changed. By scanning just the changes, one can be sure the page is consistent with any policies for content or format without having to review the entire page at the same level of detail. A user of the web might identify the way a particular page has changed since it was last viewed (Dougliis et al. 1998). For example, consumers can use the technology to monitor new products, services or auctions on E-commerce websites.

HtmlDiff (Ball and Dougliis 1996) is a program to display the differences between pages. An initial version of HtmlDiff was written in SML of NJ, using the LCS comparison algorithm that was described by (Hirschberg 1977) and is used in the well-known UNIX `diff` program. Unfortunately, it had some shortcomings:

- It proved to be computationally expensive, for example taking over 3 minutes on a 200MHz Sun Ultra 1 to compare two similar 100-Kbyte files with about 800 anchors and 4000 words (Dougliis et al. 1998).
- It displayed the differences between two HTML files as a "merged" page, marking up the newer page in place with font changes to highlight the differences from the older page. While this worked well in some cases, it often resulted in an aesthetically unpleasing display. Also, as a "derivative work" based on the newer page, it required that whoever generated the differences hold the copyright to the work. Side-by-side display of the old and new pages provides an alternative to this approach.
- It ran only on a single machine on which SML and HtmlDiff were installed, so it was not scalable to a large community of users. In contrast, a Java applet could be shipped to execute on an individual user's machine.

We therefore chose to reimplement HtmlDiff using (1) Java, (2) a more recent comparison algorithm developed by (Jacobson and Vo 1992) and (3) side-by-side display of differences. We refer to the new implementation as *TopBlend* [1]. In this paper we report the design and implementation of TopBlend, and an empirical analysis of its performance by comparison to the SML version of HtmlDiff. An extended version of this paper (Chen et al. 2000) provides additional information that has been omitted due to space constraints.

Background

This work is part of the AT&T Internet Difference Engine (AIDE), which is a tool for tracking and viewing changes on the web (Dougliis et al. 1998). AIDE permits users to provide URLs of pages of interest, which it then checks for changes. It archives versions of pages users see, either upon an explicit request or more commonly by saving each new version as it becomes available and then recording which versions a user actually sees. By default, a user is shown the differences between the current version of a page and the last version seen by the user.

[1] The name TopBlend was selected in homage to Peet's coffee, a prominent venue in Berkeley, California, where two of the authors did graduate studies. It is selected in keeping with the common practice of naming Java-based systems in a coffee genre, while recognizing the effect of blending two versions of a page together.

As a result of AIDE's automatic archival, we have a repository of about 750 Mbytes of web pages, consisting of 7600 URLs over a period of 4 years. Of these URLs, about 1700 are currently being tracked for changes. The mean number of versions per page is 22. It is important to note that these pages are not representative of the web as a whole. Most of these pages have been selected explicitly by one or more users within AT&T Labs-Research for tracking; therefore, they are skewed toward academic research pages, conferences, personal home pages, and the like. However, a sizable fraction of the pages in the repository were archived by a project that performed random sampling of the web, using URLs that were obtained at random from the AltaVista search engine. Most of these pages are regular HTML pages and are suitable for comparison with TopBlend or HtmlDiff.

Given that the pages in the archive *are* representative of the set of pages HtmlDiff compares when used within AIDE, indications that TopBlend executes more efficiently than "HtmlDiff classic" suggest that AIDE users can benefit considerably from improved performance.

Related Work

A number of systems are now available to track when pages are modified ((URL-minder 1996) appears to be the most widely known example), but most of them do not tell in detail *how* a particular page has changed. We are only aware of a few systems that support the comparison of web pages at the HTML level. One is AIDE; another is WebBeholder (Saeyor & Ishizuka 1998), which also uses an HtmlDiff-like tool and an agent community to find and display changes on the World Wide Web; and the last is WebIntegrity (Mortice Kern Systems 1997), which displays the differences between two versions of a page on a site, using frames to display the old and new versions side-by-side. Elliot Berk of Princeton University reimplemented HtmlDiff in Java (Berk 1996), again using Hirschberg's algorithm, but the prototype was not fully functional (Douglis et al. 1998). Like Hirschberg's algorithm, the Jacobson-Vo algorithm we adopted is worst-case quadratic time, but the Jacobson-Vo algorithm has the nice property that the worst cases are very unusual and it typically will run in linear time.

Human-readable comparisons of HTML pages are one way to represent the differences between versions. Another way is a binary-level comparison, which will typically run faster and encode more efficiently. In this study, we use Vo's *vdelta* program (Korn & Vo 1995) to compute a delta-encoding of two versions of a page for the purpose of comparing TopBlend's performance with a best-case binary-level comparison.

Architecture Of Topblend

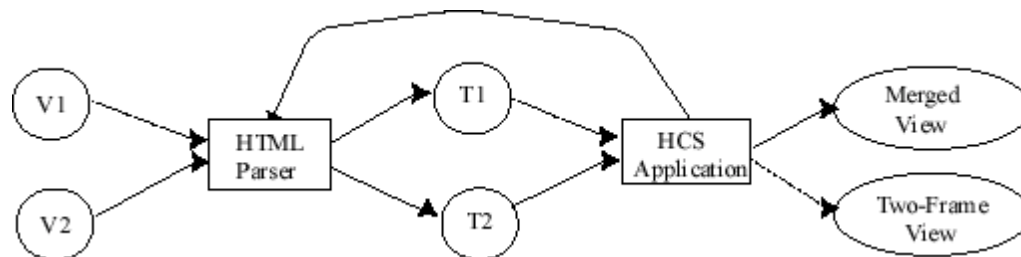


Figure 1: Architecture of TopBlend

TopBlend has two major components (Fig. 1), an HTML parser and an application of the Jacobson-Vo algorithm for the HCS (Heaviest Increasing/Common Subsequence) problem (Jacobson & Vo 1992). The parser breaks down each HTML file into a set of blocks according to HTML tags and emits a token sequence. The HCS algorithm compares the two token sequences generated by the parser and outputs the difference information in either a single merged HTML file or a frame view for side-by-side comparison.

HTML Parser

One of the main goals of TopBlend is to respect the HTML structure as much as possible so that it does not compare text across structure boundaries. For example, consider an example (Fig. 2) that consists of an old table and a new table. The latter adds a new row to the former.

If a tool simply divides the HTML file into a sequence of tokens without considering the table structure, then it may mistakenly mark the insertion as crossing two rows, as shown in the left table of the next example (Fig. 3). This is because the first "Denver" token is mistakenly matched across the first old row and the new row. On the other hand, if a tool matches by table rows before matching within rows, then we get the correct table shown in the right side (Fig. 3). This presentation, which TopBlend produces, is more consistent with a user's intuition.

City	ACCESS#	V.90
Denver	(303)824-0003	Yes
Denver	(303)824-0091	Yes

City	ACCESS#	V.90
Denver	(303)824-1037	Yes
Denver	(303)824-0003	Yes
Denver	(303)824-0091	Yes

Figure 2: WorldNet Access Numbers for Denver: Old and New

City	ACCESS#	V.90
Denver	▶(303)824-1037	▶Yes
▶Denver	(303)824-0003	Yes
Denver	(303)824-0091	Yes

City	ACCESS#	V.90
▶Denver	▶(303)824-1037	▶Yes
Denver	(303)824-0003	Yes
Denver	(303)824-0091	Yes

Figure 3: Comparison Results: Misleading Output and More Intuitive Output

In order to preserve the original HTML structure, the HTML parser classifies the HTML tags into three types:

Intact tag: The content between the starting tag and the ending tag of this category is considered a single unit even if there are other tags inside the block. Examples include table tags like `<tr>` and `</tr>` and list tags like `` and ``.

Ignored tag: The content between the starting tag and the ending tag of this category will not be compared or broken down further. Examples include (a) an HTML comment, (b) HTML content that cannot be compared visually, such as choice menus (`<select>`), (c) Non-HTML code that the parser cannot handle, such as scripts and Java applets.

Simple tag: All other tags. Plain text between such a tag and the next tag is considered a unit for comparison.

The HCS algorithm

The heart of the TopBlend system is a Java implementation of the Jacobson-Vo algorithm that solves a generalization of the longest common subsequence (LCS) problem. The line-matching algorithm of the well-known Unix command `diff` is based on computing an LCS between the sequences of lines of text of two given files. Unmatched lines are insertions and deletions. Since LCS is based only on matching symbols, the `diff`'s line matching algorithm cannot take advantage of line lengths or approximately matched lines when determining line insertions and deletions. (Jacobson & Vo 1992) discussed how to include weights when matching symbols. In the most general case, weights can be associated to both symbols and their locations in the sequences being matched. Therefore, the Jacobson-Vo algorithm is capable of solving the *heaviest* common subsequence (HCS) problem.

In TopBlend, there are three stages in applying the HCS algorithm. At every stage, the checksum of each token (computed from its string content), rather than the string itself, is used for comparison to improve performance.

The HTML parser first breaks the two HTML files into sequences of blocks divided by the tags of the immediate children of the `<BODY>` tag. A token at this stage is either a first-level tag or a block between two first-level tags. All these tags share the same weight, which is the size of the html file. The weight of each block is the block length. The HCS algorithm is then used to find the heaviest common subsequence between these two sequences. This allows us to quickly identify chunks of identical text within the HTML first-level structure.

After that, it applies the HTML parser again, step by step, to break up each block (if there is any) between every pair of consecutive, matched tokens in the first stage. The parser breaks each block into a set of tokens, with each token representing either an intact block, an ignored block, or a simple block. The HCS algorithm is then applied to detect the longest common sequence of these block tokens based on their checksums.

Finally, it applies the HTML parser again, step by step, to each section of simple tokens (words, tags, etc.) between every pair of consecutive, matched block tokens in the second stage. At this point, the parser simply breaks up the intact blocks and simple blocks into a stream of simple tokens, while leaving each ignored block as it is. TopBlend then invokes the HCS algorithm again on these subsequences.

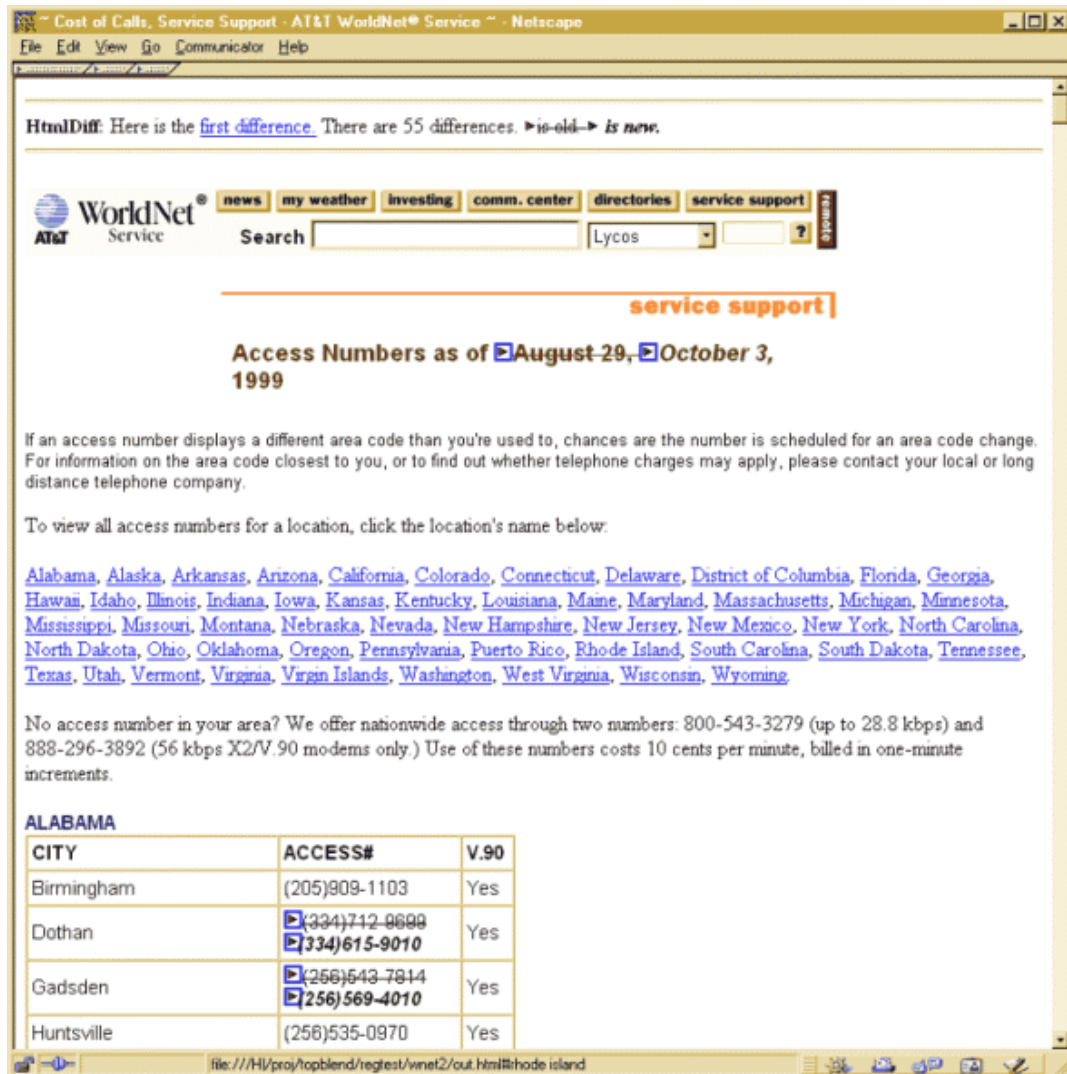


Figure 4: Merged View: Differences between Two WorldNet Pages on Access Numbers

TopBlend Views

TopBlend has a simple command line interface that takes two input HTML files and generates a specified output file in either the merged view or two-frame view. As an example (Fig. 4), we show the merged changes in the AT&T WorldNet access number list from August 29, 1999 to October 3, 1999. There are 55 differences and each is marked with an anchor that takes one to the next. New or changed entries in tables such as this, organization charts, or product catalogs can best be spotted using merged views. On the other hand, when a page has complex layouts with embedded images, it is usually better to use the frame view (Fig. 5). Here, the top frame shows the list of differences, the lower left frame shows the new page with new text highlighted in pink, and the lower right frame shows the old page with the old text dimmed in gray. The frame view clearly shows some textual changes, as well as making some changes to images apparent (though they are not marked explicitly).

Performance

To evaluate the performance of TopBlend relative to HtmlDiff, we ran a series of benchmarks using the AIDE archive. We selected 500 pages for which at least five versions were available, and extracted the most recent five versions from the archive. We then did pair-wise comparisons of successive versions (<N-4,N-3>,<N-3,N-2>,<N-2,N-1>, and <N-1,N>, where N is the highest version number of the page in the archive). In each case we ran a set of comparison programs on the aforementioned Sun Ultra 1, with 256 MBytes of DRAM, and minimal other applications executing. Because many individual executions took minutes or even hours, we did not repeat identical

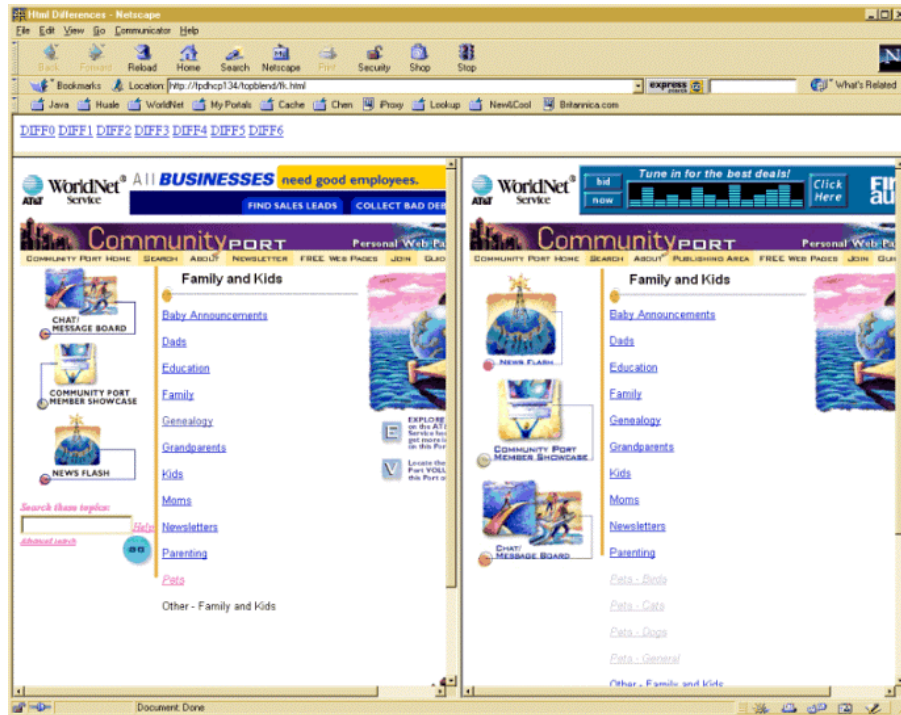


Figure 5: Two-Frame View: Differences between Two Versions of a Page on WorldNet's Community Port

inputs multiple times to improve statistical significance. Instead, we rely on the similarity of comparisons of multiple pairs of versions of the same page and the large number of samples overall. The four benchmarks were as follows:

- *HtmlDiff*. This was the original SML HtmlDiff program.
- *TopBlend*. This was run in "merged" mode, and should produce results virtually identical to HtmlDiff.
- *Vdelta*. This program computes a byte-wise comparison of the two files based on a new fast string algorithm developed by Kiem-Phong Vo. It is not sensitive to HTML markup.
- *Diff -e*. This is the UNIX `diff` program, using the LCS algorithm. It compares on a line-by-line basis.

Using the UNIX `timex` program, we compared the four programs on the basis of execution time (CPU time), mean memory size, and total memory usage (the product of memory size by execution time). We were primarily interested in the differences between HtmlDiff and TopBlend, and used the other two programs as lower bounds against which to compare any possible improvement. Due to space limitations, this paper presents only execution time comparisons; refer to the extended version of this paper (Chen et al. 2000) for additional data.

Execution Speed

We plot (Fig. 6) on a log-log scale the cumulative probability distribution function for each application, where the x-axis represents time and the y-axis represents the fraction of instances in which a given application finished within that time. In general, both `vdelta` and `diff` executed on nearly any input in a fraction of a second. Note that `diff` compares lines while both TopBlend and HtmlDiff perform detailed token comparisons. HtmlDiff executed in under a second on simple inputs but could take minutes on more complicated ones; and TopBlend took about a second on simple inputs (partially due to the start-up time of the Java virtual machine) but ran significantly faster than HtmlDiff on the more complicated ones. The TopBlend and HtmlDiff lines cross at about the 1.7-seconds point; i.e., a greater fraction of HtmlDiff executions took more than 1.7 seconds than did TopBlend executions. Furthermore, while 98.8% of TopBlend executions finished within 1 minute, only 96.7% of HtmlDiff executions did, while the 99th percentile for HtmlDiff was nearly 7 minutes. Since about 99% of the pages can be compared within a minute by TopBlend, the program can be used for on-the-fly comparisons.

Conclusions

As information explosion occurs on the web, the need to digest information quickly will increase. The web age differencing technology offered by TopBlend provides relief by helping web users to identify new or changed

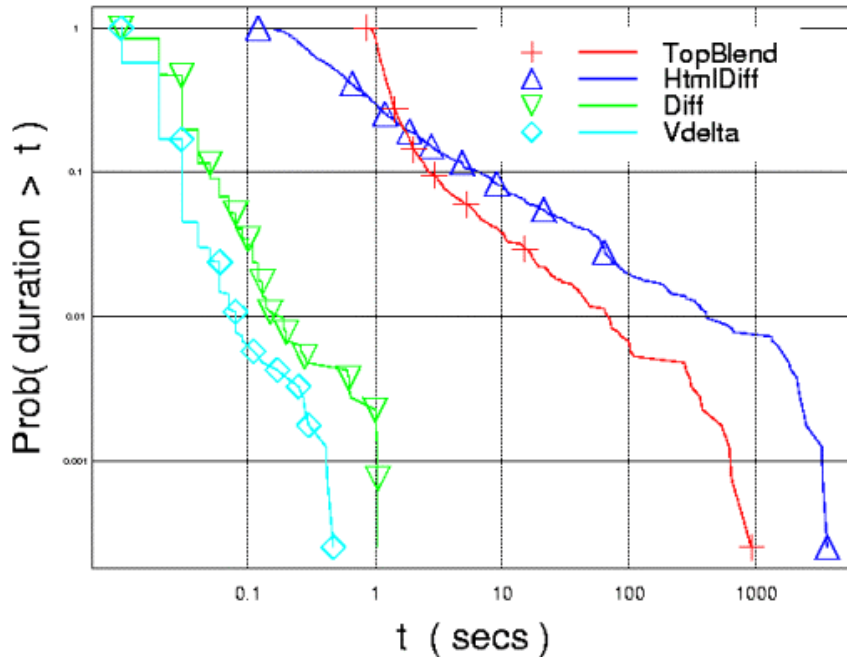


Figure 6: The Cumulative Probability Distribution Function of execution times

web content on their favorite websites. By adopting a fast algorithm, TopBlend significantly outperforms the old HtmlDiff in the most time-consuming jobs and finishes most typical jobs in a few seconds. Furthermore, by migrating to Java, TopBlend allows processing to be performed on increasingly powerful clients rather than on a busy server. For websites with complex graphics designs, the new two-frame view offered by TopBlend avoids clutter by highlighting differences using a side-by-side comparison.

Acknowledgments

Xiaoning Zhang implemented an initial version of TopBlend during an internship at AT&T. Anja Feldmann provided assistance with the statistical analysis. Several anonymous reviewers provided valuable comments.

References

- Ball, T. & Douglass, F. (1996). Tracking and viewing changes on the Web, *Proc. 1996 USENIX Technical Conf.*
- Berk, E. (1996). HtmlDiff: A Differencing Tool for HTML Documents, *Student Project*, Princeton University.
- Chen, Y.-F et al. (2000). TopBlend: An Efficient Implementation of HtmlDiff in Java, *AT&T Labs - Research Technical Report 00.5.1.*
- Douglass, F et al. (1998). The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web, *World Wide Web*, 1(1).
- Hirschberg, D. (1977). Algorithms for the longest common subsequence problem, *Journal of the ACM*, 24(4).
- Jacobson, G. & Vo, K.-P. (1992). Heaviest Increasing/Common Subsequence Problems, *Proc. 3rd Annual Symp. of Combinatorial Pattern Matching*, Vol. 64.
- Korn, D.G. & Vo, K.-P. (1995). Vdelta: Differencing and Compression, in *Practical Reusable UNIX Software*, B. Krishnamurthy, ed., John Wiley & Sons, New York.
- Mortice Kern Systems Inc. (1997). Web Integrity, <http://www.mks.com/products/wi/>, 1997.
- Saeyor, S. & Ishizuka, M. (1998). WebBeholder: A Revolution in Tracking and Viewing Changes on The Web by Agent Community, *Proc. WebNet98.*
- Url-minder (1996). <http://www.netmind.com/URL-minder/URL-minder.html>.