WebGazeAnalyzer: a system for capturing and analyzing web reading behavior using eye gaze.

2 authors:

David Beymer

IBM

89 PUBLICATIONS   **3,977** CITATIONS

SEE PROFILE

Daniel M. Russell

Google Inc.

116 PUBLICATIONS   **2,288** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Data preparation, data science, and visual analytics tools for business View project

The MIT Vision Machine Project View project

WebGazeAnalyzer: A System for Capturing and Analyzing Web Reading Behavior Using Eye Gaze

David Beymer and Daniel M. Russell
IBM Almaden Research Center
650 Harry Rd, San Jose, CA 95120
{ beymer, daniel2 } @us.ibm.com

ABSTRACT

Capturing and analyzing the detailed eye movements of a user while reading a web page can reveal much about the ways in which web reading occurs. The WebGazeAnalyzer system described here is a remote-camera system, requiring no invasive head-mounted apparatus, giving test subjects a normal web use experience when performing web-based tasks. While many such systems have been used in the past to collect eye gaze data, WebGazeAnalyzer brings together several techniques for efficiently collecting, analyzing and re-analyzing eye gaze data. We briefly describe techniques for overcoming the inherent inaccuracies of such apparatus, illustrating how we do capture and analysis of eye gaze data for commercial web design problems. Techniques developed here include automated Document Object Model (DOM) parsing to extract page content, methods to group fixations along lines of text, and reading analysis to measure reading speed, regressions, and coverage of web page text. We illustrate the system with a brief example of eye gaze analysis, showing that highlighting text on a web page increases reading time of the text, not by causing re-reads of the text but instead by slowing the reader down.

Author Keywords

Eye gaze; reading detection; gaze registration; eye tracking data analysis methods.

ACM Classification Keywords

H.5.2. User Interfaces: GUI, Screen Design; evaluation/methodology, screen design, style guides; H.5.4 Information Interfaces and Presentation; H.1.2 User/Machine systems: Human information processing.

INTRODUCTION

There are many ways to watch people as they use web pages for specific tasks. Most commonly, user studies track how often a user visits a particular page, how long they stay on a page, and generally the number, kind and organization of their web movement behavior [1][2].

In addition, there are a growing number of researchers using eye gaze systems to watch the micro-behavior of users as they read web pages, performing various tasks. For example, researchers at PARC [3][5][4] have used information foraging theory to develop a method for analyzing the web search task using the "information scent" of a link. The Poynter Institute and Eyetools [6] have studied eye movements while subjects browsed a news web site. Hornof and colleagues have studied the usefulness of web banner ads [7] and how hyperlink text color can guide a web search [8].

Eye gaze tracking systems have been used for some time to monitor reader's behavior [20][21], and have now evolved into two main types of systems. Eye trackers can be head-mounted (e.g. EyeLink II [9]), or they can be "remote camera" systems that track head movements and eye movements by fairly inconspicuous equipment that is neither obvious nor infringing on a user's personal space. (Examples of remote camera systems are LC Technologies [10] and Tobii [11].)

Each type of system makes a tradeoff between invasiveness and eye gaze accuracy. There is some evidence to indicate that remote camera gaze tracking results in behavior that is more "normal" than systems requiring extensive head and neck motion interference [19].

The Problem and our Contribution

To gain insights into how users actually behave in web reading and web-based learning tasks, we have developed a tool called WebGazeAnalyzer. Our tool records a user's eye gaze and entire web browsing session, and then it analyzes the user's reading behavior on the recorded web pages. The recording system records a rich data stream, including a screen capture movie, the parsed Document Object Model (DOM) of each web page, the eye gaze itself, and Windows events such as scrolling and mouse clicks. Reading analysis automatically detects fixations, matches groups of fixations to text lines on the web page, and finally

measures statistics such as the time spent reading, what material was read, the reading speed, and regressions.

While there are similar systems for recording eye gaze while web browsing [12] [13] [24] [4], our system makes three main contributions. First, the DOM forms the basis of text extraction from the web pages and drives the reading analysis. The DOM is a dynamic, hierarchical representation of HTML content that is automatically generated by web browsers each time a web page is loaded. From the DOM, we extract the text on a page and its location down to the word level. Lines of DOM text are a key building block for our reading analysis, and the experimenter can specify content-of-interest by selecting a subset of the DOM. This content-based method for specifying regions of interest is superior to using bounding boxes alone, which is prevalent in today's systems.

Second, we have developed a reading analysis system suitable for a remote eye gaze tracker with error of 0.5 deg. Most reading analysis to date has concentrated on head-mounted eye trackers to get high accuracy. Since the obtrusiveness of these systems may affect reading behavior, it is desirable to use a remote tracker. Our reading analysis deals with inherent noise and drift problems in remote trackers – fixations are grouped along horizontal scans and matched against the text DOM lines in a robust manner.

Finally, existing tools for recording and browsing eye gaze lack interactivity. The experimenter may visualize the eye gaze and its interpretation, but he or she may not annotate or change it. In our system, the experimenter inspects the automatic analysis and can add annotations to correct mistakes. The annotations are immediately fed back into the analysis and are displayed. This encourages the experimenter to inspect and validate the eye tracking data.

SYSTEM OVERVIEW

WebGazeAnalyzer is the system we built to capture and analyze web reading behavior. It records the subject's eye gaze data during an experimental run. Afterwards, the data is automatically cleaned-up, aligning the subject's eye movements on the screen with the actual positions of text and figures as they appear on the monitor. In addition, analyses of reading behavior are performed, populating an extensive database of reading/scanning eye gaze behaviors with page changes, mouse movements and screen updates. After the analysis is complete, WebGazeAnalyzer can be run as an inquiry tool, letting the experimenter examine the captured data in fine detail. Further, the system allows the experimenter to identify regions of web pages as being of particular interest, and then a new analysis can be quickly re-run to test hypotheses and speculations about the captured behavior.

In the following text, we first describe the WebGazeAnalyzer system, giving a brief overview by focusing first on data capture methods, then data structuring

methods, followed by a synopsis of our reading detection algorithm and a short case study of the tool in use.

PART 1: THE RECORDING SYSTEM

Since our experiments focus on web reading behavior, we designed a recording system to record the participant's eye gaze and entire web browsing session. A number of data sources are recorded, providing a rich data set that can be visually browsed or analyzed by a batch system. A key requirement of our system is to handle the dynamics of web browsing, including all web pages visited, the start and finish of each page visit, scrolling events, mouse clicks, etc. While some commercial systems also perform similar functionality, one key additional source of information we record is the web browser DOM, which is the basis of our reading analysis.

Hardware Setup

Our system hardware consists of the Tobii 1750 eye tracker plus two IBM T41 laptop computers, one of which is a gaze server machine, and the second is the participant machine. The Tobii 1750 is a flat panel monitor with an eye tracking camera and infrared LEDs mounted inside the monitor bezel. The camera field of view is wide enough to allow head motion of 30 x 15 x 20 cm (width, height, depth), so the participant does not have to remain uncomfortably still during the experiment. The Tobii reports eye gaze at 50 Hz and with an accuracy of approximately 0.5 deg [11].

The gaze server machine runs the eye tracking system and is the experimenter's console machine. The participant machine runs an instrumented web browser for the participant and thus drives the Tobii monitor. Both machines have their clocks synchronized using the Network Time Protocol (NTP) to a local time server.

We also use a camcorder to take video footage of the participant during the experiment. This provides additional context in interpreting the eye gaze and screen recording information – is the participant looking away from the screen? Are they frustrated, happy, or confused? Are they saying anything? This video is synchronized to the gaze and web bowser data streams by manually specifying one time offset, which makes all subsequent playback and analysis time-coordinated between collected eye data and camcorder video.

Recording Software

The recording software consists of three main components, one of which runs on the gaze server, and two on the participant machine. On the gaze server runs the experimenter's control system, which displays the gaze tracking status and controls the experimental design and recording. Gaze is shown in screen coordinates, allowing the experimenter to monitor the participant's progress; also, the eye positions are displayed in camera coordinates, allowing the experimenter to monitor head drift out of the camera's field of view.

On the participant's machine runs two software systems, a screen recorder and an instrumented web browser, that we call IE*. IE* is a version of Microsoft's Internet Explorer that is augmented to record the DOM of every web page visited, along with a number of web browser events such as scrolling and page loading. All of these software systems record the following data streams, which are time stamped and recorded to disk:

1. *Gaze points*. On the gaze server, the experimenter's console system receives the eye-tracking stream from Tobii's TETServer program and saves it to disk.

2. *Event-driven screen recording*. We have developed an innovative technique for recording the Windows screen in an efficient, event-driven manner based on a modified version of the Virtual Network Computing (VNC) system (please see the details below).

3. *HTML content / DOM*. IE* saves a parsed version of the DOM for each web page visited. Using the DOM, all graphics and all text are extracted from the web page, and Microsoft extensions to the DOM give us bounding boxes for all text – we compute these down to the word level. In addition to extracting the DOM from each page, the URL address, page loading times, and scrolling events are also stored. The next section on reading analysis discusses how we use the DOM in detail.

4. *Windows events*. Using Windows hooks, we record mouse, keyboard, window positioning, and window resizing events. The system also monitors the creation of IE-related popups (e.g. a list box popup from clicking on a combo box) or new browsers (and starts a DOM recording for them).

Why develop a new screen recording technique? Many screen recording systems [12, 14] record at a fixed frequency, say at 10 Hz. When there is no or little screen activity, this is wasteful of system storage and CPU cycles. On the other hand, if there is lots of screen activity, then a fixed recording rate may miss visual events. In our opinion, it is better to have the screen recording synched with Windows painting events. Since we only record a screen rectangle that is actually changing, it is efficient. Since the rectangle capture is event-driven, it won't miss events.

We have modified the VNC remote desktop system to achieve these goals in a screen recording system. Under normal operation, the *winvnc* server sends rectangle screen updates to a *vncviewer* client on a separate machine over a socket connection. To gather realtime screen data, we modified the VNC server to add a timestamp to each rectangle update and redirect the socket output to a local file. This creates a file of timestamped screen updates. After the experiment is over, image keyframes are added to the file to make it easily browseable in our playback and analysis system.

PART 2: READING ANALYSIS
Given the recorded eye gaze from an experimental run, can we analyze it to tell when and what the participant was reading? Eye movement patterns characteristic of reading, extensively studied in the psychology literature [15], open the door for an automated analysis of reading. While readers intuitively may think that their eye gaze follows a continuous left-to-right motion, eye tracking studies show that eye motion advances in discrete chunks across the page. A reader's eyes will actually stop, or fixate, on a set of characters for about 250 ms. This *fixation* is followed by a *saccade*, an eye movement of about 10 characters to the
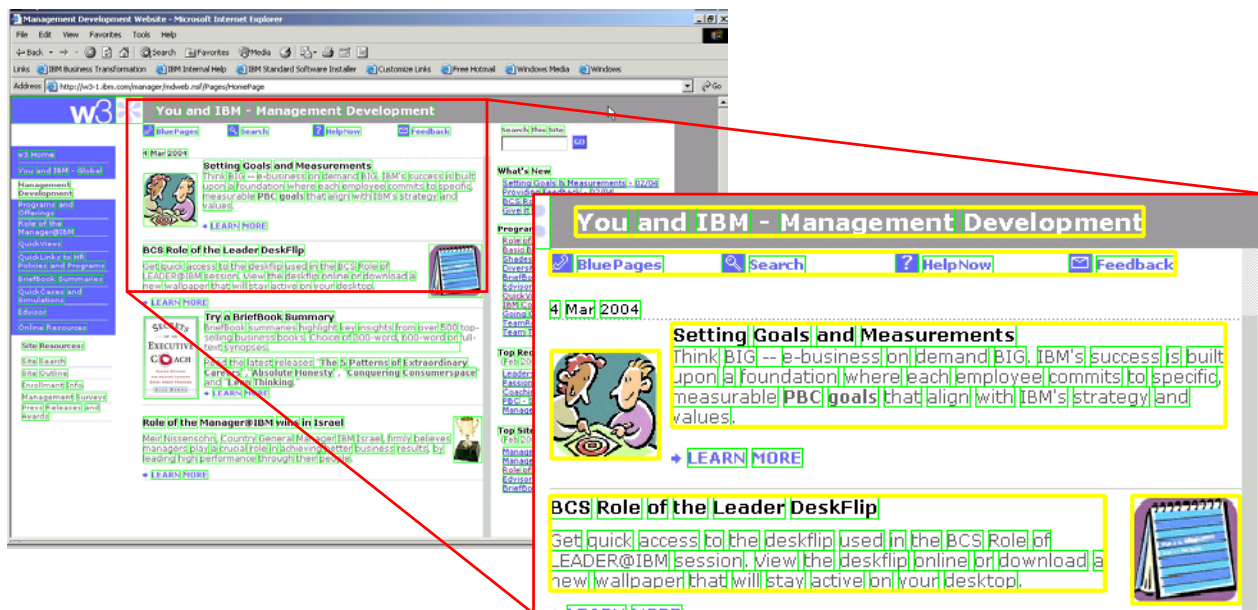

Figure 1: Parsed DOM words and pictures (left), with magnified region (right) showing content tagged for analysis. The bounding boxes are not shown to users, but simply illustrate DOM analysis and content tagging with B-tags.

right, where the eyes will stop at the next fixation. A *regression*, or backwards eye movement in the text, is a sign that the reader is having difficulty understanding the material.

This characteristic pattern of fixations and saccades can be automatically detected from the recorded eye gaze. Through reading analysis, we can tell if the participant is reading or scanning the web page, what the reading speed is, and even detect cues on reading difficulty from regressions.

DOM as a Representation of Content

While the pattern of eye fixations and saccades can tell us when the participant is reading, it reports the reading location in screen pixel coordinates. Intersecting these locations with the DOM text tells us *what content* the participant is reading.

As mentioned previously, the DOM is a dynamic, hierarchical representation of HTML content that web browsers create whenever a new URL is loaded. During the experimental recording, IE* traverses the DOM for each new web page loaded, extracting all the text and graphics. As shown in Figure 1 (left), text is broken down into words, and the location of all words and graphics are stored. These bounding boxes are not shown to study participants, but simply illustrate where we have found items.

While knowing the bounding boxes of individual words allows us to analyze eye gaze down to the word level, we would also like to aggregate content into larger chunks for analysis. Depending on the analysis task at hand, we can group the DOM content into "content-of-interest" tags that we call "B-tags". These "B-tag" can group any subset of DOM content, such as paragraphs, navigation bars, individual words, etc. Figure 1 (right) shows yellow bounding boxes around some B-tags defined for later analysis.

B-tags are a content-based way of defining areas of interest, which are typically defined in today's eye gaze analysis systems using bounding boxes or polygonal regions. But because B-tags are defined in terms of the content itself (i.e. ASCII strings), they are intrinsically more flexible.
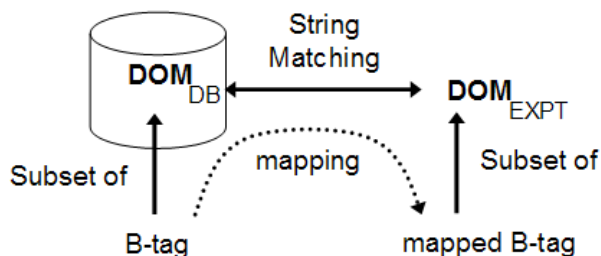
Figure 2 String matching is used to map a B-tag from an annotated database DOM_{DB} to recorded DOM_{EXPT}.

Consider the process of defining and using B-tags. B-tags are defined for a given URL, using IE* as an annotation tool. IE* loads the URL, parses the DOM, and then the experimenter/annotator selects HTML content using the normal Windows "click and drag" selection mechanism. The selection can be mapped to a subset of the parsed DOM, which defines the B-tag. This is repeated for all URLs for which one wants to define B-tags, creating a database of DOMs, DOM_{DB}, and their associated B-tags.

Given the recorded DOM from an experimental run, DOM_{EXPT}, the system can map the B-tags of a database DOM, DOM_{DB}, to DOM_{EXPT} using simple string matching (see Figure 2). If the B-tag content is matched across the DOMs, then the B-tag can be transferred from DOM_{DB} to the matched content in DOM_{EXPT}. Now the mapped B-tag can be used as a content-of-interest region in the experimental run.

We have found this to be a very robust method for defining content-of-interest, working even if portions of the URL are changed – it depends on the string matching to work. This is less fragile than an area-of-interest defined by a bounding box alone. It works if the browser if resized and the screen layout changes. It works if the screen content is dynamic (e.g. rotating touts). Also, one can match more than one DOM_{DB} against DOM_{EXPT}, which makes tagging repetitive content (e.g. a header bar) simple.

Fixation Grouping and Matching

To assign eye gaze fixations to precise DOM words and lines, one needs to handle the accuracy problems eye gaze trackers have with noise and drift. Head-mounted eye trackers are quite accurate, so mapping gaze fixations to text lines is straightforward. However, encumbering the participant with an uncomfortable head-mounted unit may skew their reading behavior. Remote trackers such as the Tobii are non-obtrusive, but noise and drift makes the matching of gaze fixations to DOM text lines ambiguous at times. However, while mapping a single fixation to the text lines may be ambiguous, the matching of *groups* of fixations to chunks of the DOM can resolve this ambiguity. Note that we do this with respect to the data, rather than requiring fixed calibration points [17].

Our strategy is to match gaze fixations to the DOM text in groups. Since in reading, a natural grouping is horizontal lines, we first group fixations into horizontal scans, or "gaze lines". Similarly with the DOM, lines of DOM words are computed using the word bounding boxes, giving us a set of DOM text lines. Since the y coordinate drift can still affect an entire gaze line, matching will match groups of gaze lines against groups of DOM lines. The ambiguity in matching goes down as the group size increases. Of greatest ambiguity are long, fill-formatted paragraphs, since the lines are not distinguished from one another. Short
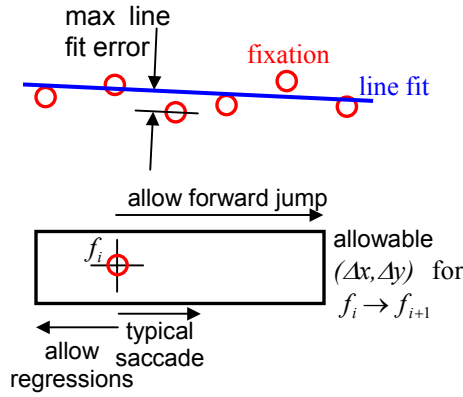
Figure 3: The top shows the line error metric for growing gaze lines, and the bottom shows the acceptable range of positions for fixation $i+1$, showing a bias for a typical saccade.

paragraphs, especially involving lines with unequal length, form natural line groups for the matcher.

At this point we are interested only in aligning gaze lines with DOM lines; details like detecting regressions and reading speed will be discussed in the next section. The remainder of this section will discuss fixation detection, grouping fixations into gaze lines, and finally matching gaze lines against DOM lines.

Fixations are detected using the dispersion-based method described in [16]. In the dispersion-based method, a fixation is a portion, or time window, of eye gaze where the spatial dispersion on (x, y) coordinates is below a threshold. Typically, one starts with a time window of minimum length (e.g. 100 msec) and grows the window to include successive gaze points while the dispersion remains under threshold. Fixations are represented by their mean (x, y) location and their start and stop time.

Next, fixations are grouped into horizontal "gaze lines" in preparation for the line-based matcher. Detecting gaze lines is driven by line fitting and uses a growing technique that is similar in spirit to the dispersion-based fixation detection. The detector grows a group of consecutive fixations by fitting a line to the fixation centers; the group is grown as long as the maximum fixation to line error is below a threshold (see Figure 3). Also, the $(\Delta x, \Delta y)$ between consecutive fixations is required to be consistent with reading behavior (again, see Figure 3).

Finally, the matcher matches groups of gaze lines against groups of DOM lines. Each such group begins with a hypothesized seed match. For a given gaze line, we gather potentially matching DOM lines under an estimate of current gaze tracker drift, including a mean correction $(\Delta x, \Delta y)$ and its variance. If the match is unique—i.e. only one DOM line found—then we accept the match, continue with the next gaze line and update the drift model. If the matching is ambiguous, then we form a group hypothesis

for each seed match. Each group updates its own drift model based on the alignment of seed gaze-DOM lines. Next, we try to extend each group by adding successive gaze lines, hence growing the group size. There are three potential outcomes:

1. If only one group hypothesis survives, then we accept the matches in the surviving group. The next unmatched gaze line is used to seed the next group.

2. If more than one hypothesis survives, then continue growing the group hypotheses with the next gaze line (thus increasing the group size).

3. If no hypotheses survive, then the current seed gaze line is not matched, and we restart the seed group with the next gaze line.

All of these detected events—fixations, gaze lines, the matching results—can be visually inspected using the interface described in the GUI section. An important new capability of our system is that the experimenter can add manual annotations in a timeline view to correct mistakes made by the automatic system just described.

Measuring Reading Statistics

Now that the line matching algorithm has assigned fixations to DOM lines, we re-examine the fixations to compute reading statistics. For a particular DOM line, we measure

1. *Reading coverage*. What is read, what is re-read?

2. *Regressions*. These are eye movements backwards in x coordinate in the text.

3. *Reading speed*. We can estimate an instantaneous measure of speed or aggregate over multiple lines.

To estimate these reading statistics, we parse the fixations for a given DOM line into *forward-reads*. A forward-read is a set of consecutive fixations moving forward along a single text line with typical forward saccades. It corresponds with intuition about when the eye is actually reading the underlying text. The backwards motion in a regression will break the reading of a line into multiple forward reads, and thus will not count towards reading. Likewise, eye movements between DOM lines will also
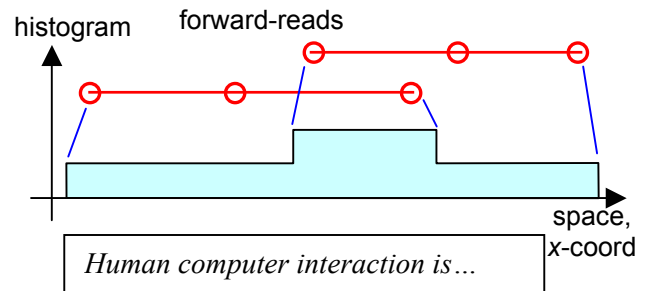
Figure 4 Mapping forward-reads to reading coverage. Note that the word *interaction* is read twice.

cause a break in the current forward-read. A visualization of reading coverage can be created from forward-reads by forming a histogram of all the forward-reads on a line (see Figure 4).

What if we are interested in analyzing reading behavior for a particular word on a line? At first, the discrete nature of fixations on the DOM line seems to complicate the analysis. Parsing the fixations along forward-reads, however, gives us the opportunity to smooth out the discrete spatial jumps at fixations, allowing interpolation. The discontinuous jumps in space are replaced by line fitting, as shown in Figure 5. If a word-of-interest is contained inside a forward-read, we map the x-coordinate of the start and end of the word through the graph to the time dimension. This provides a rough estimate of the amount of time spent reading the word. In Figure 5, the spatial distances d_0 and d_1 are related to the perceptual span – the number of characters taken in to the left and right of a fixation [15]. In addition to reading time, we can report when regressions occur near a word or when multiple forward-reads map to the same word.

In addition to estimating reading coverage and reading times, reading speed can be estimated over multiple time frames. It can be measured instantaneously by computing the slope of the space-time graph for a forward-read. Speed can be reported as averaged over a forward-read – total distance / total time. Speed can be measured over B-tags by appropriately averaging the speeds of the forward-reads that cover DOM lines contained by the B-tag.

PART 3: DATA AND ANALYSIS BROWSER
Given the data recorded from a participant experiment and the subsequent reading analysis, we have developed a browser for visualizing this data and analysis, allowing for exploration of the time-synchronized streams of data (i.e., eye gaze, overall video, analyzed reading tracks, time spent in B-tagged regions, etc.).

Compared to other systems for displaying eye gaze recordings, the main novelty of our system is the ability to interactively annotate and make corrections to the reading analysis. This encourages the experimenter to inspect and validate the eye tracking data, as he or she can actually *do* something if an error is found.

Video Playback Window
Figure 6 shows the screen recording movie, with overlays for eye gaze, fixations, gaze lines, and the DOM. There is a time slider in a dialog control that allows the experimenter to scrub back and forth in time. At every point the timeline crosses over a URL boundary (the recording typically contains many web pages), the DOM of the new page is loaded and the reading analysis performed for the new page becoming visible.
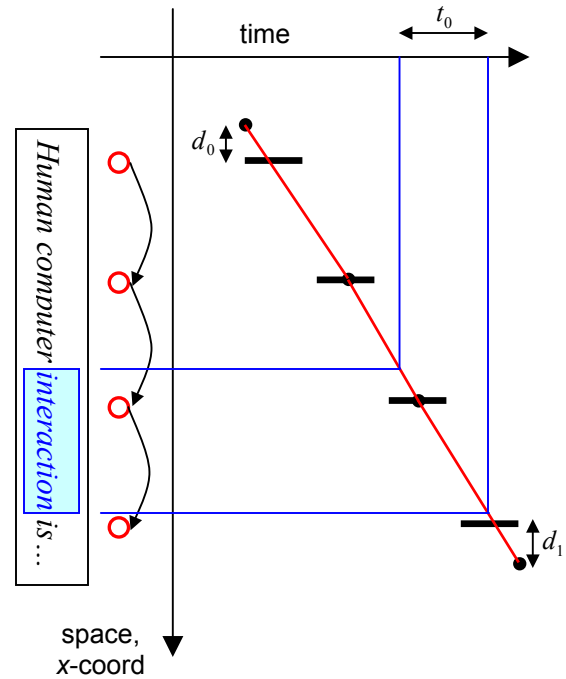

Figure 5 Our interpolation scheme for a forward-read (in red) to make the mapping between space and time continuous. The word *interaction* is mapped through the graph to estimate a reading time of t_0 .

Timeline Window
To allow the visualization of eye gaze and other events along a timeline, the browser has a timeline window. In the timeline, we find it useful to plot gaze velocity *(dx, dy)*, as fixations and saccades are readily recognizable, and problems with eye tracking noise can be seen (see Figure 6). Also in the timeline we display elements from the reading analysis – fixations, gaze lines, match groups – by showing them as time segments. The top row of time segments (in red) shows the presence/absence of eye tracking data. The following rows of segments show, respectively, the fixations (green), gaze lines (blue), and match groups (green). All these items are selectable with the mouse, which results in highlighting the selected item in the video playback window and the page window (to be described). When in playback mode, one typically does not select any items, and the system instead automatically highlights items in a 500 ms window in the center of the timeline.

Annotated time segments are an important innovation of our system, and the GUI for them is in the bottom portion of the timeline view. First, the experimenter sees an event that he or she would like to annotate or correct. Perhaps a gaze line has been missed or incorrectly detected. Then the experimenter uses Shift-mouse-drag to drag out a rectangular region defining the segment in time. A separate event dialog allows the experimenter to enter that the annotation is a gaze line. A "reanalyze" button in the GUI
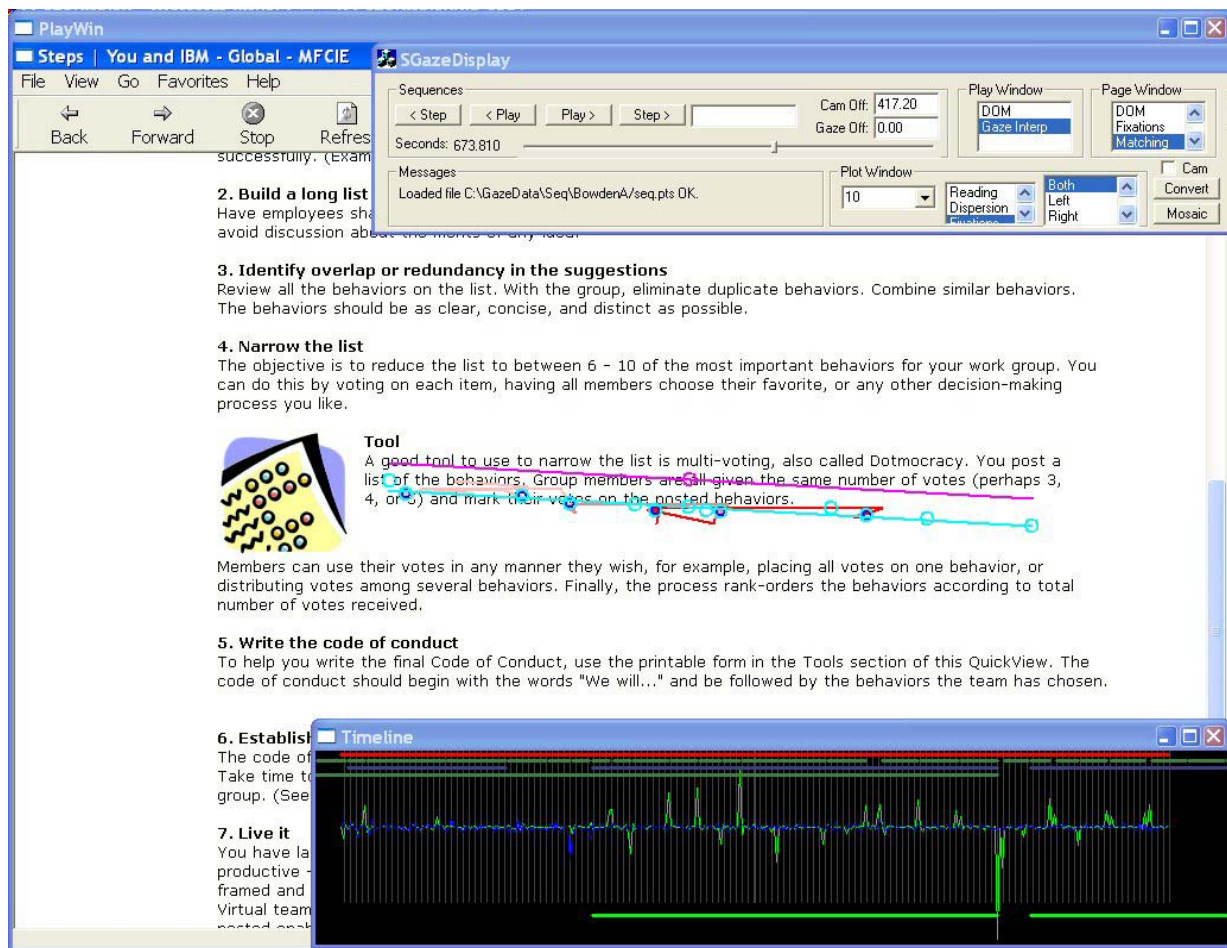
Figure 6 Browser playback and timeline windows. The playback window shows a sample gaze line (aqua) with its constituent fixations. This gaze line is matched to the DOM line beginning with "list of the behaviors", and the mapping of the gaze line under the gaze tracker drift modal is shown in magenta. In the timeline, horizontal velocity is shown in green, vertical in blue. Please refer to the text for a description of the time segments.

tells the system to re-run the reading analysis starting with the manual annotations, and this updated analysis is immediately visible to the experimenter. If the results are not acceptable, the annotation can be deleted. Incorrect gaze-DOM line matches can also be corrected by specifying which DOM line the gaze line annotation should be matched to.

To see events happening at different time scales, the window scale can be adjusted to show between 3 seconds and 20 minutes of data. Fine-scale time scrubbing can be performed in this window by dragging with the mouse.

Page Window
The page window shows the data in a content-centric view for the currently loaded URL. Over the backdrop of a clean image of the URL with scrolling unwrapped, the experimenter can show fixations, gaze lines, match groups, and DOM lines. To get a quick overview of browsing activity on the page, the experimenter can choose to show all fixations on the page. Furthermore, the fixation display can be modified to show only *reading* fixations or only

non-reading fixations, which is useful in visualizing reading and validating the reading analysis.

For the DOM lines on the web page, the reading coverage histograms from forward-reads, as first shown in Figure 4, can be plotted over the DOM lines as a heat plot (see Figure 7). This provides a quick visualization reading and re-reading behavior, showing which text on a web page the participant was drawn to or had trouble reading. If the experimenter clicks on a DOM line in the page window, then the timeline and playback windows are automatically shifted in time to the first time that DOM line was read.

B-tags showing special content-of-interest for a given experiment are also shown in the page view as bounding

Figure 7 Heat plot from the page window showing that the word "dotmocracy" -- which is highlighted in yellow in the text – is read twice by one subject.

boxes around the tagged DOM content. Recall that B-tags are automatically mapped from the DOM/B-tag database to the currently loaded DOM. Reading statistics such reading coverage and reading times are displayed next to the bounding box for the DOM lines inside the B-tag.

EXPERIMENT

As an application of our recording and analysis system, we posed a general question regarding web site reading behavior: *How does highlighting affect reading behavior?* Does it increase the time spent reading the highlighted material? Does it alter the reading speed? Does it increase regressions? Does it increase the retention of the content, as measured by a post test of the material? The specific client we were working with designs educational web sites for IBM managers, so our motivation was to see if highlighting may have an impact on online educational material.

Method: Our experimental web page describes the creation of a "code of conduct" for a manager's work group, and it is a real page from IBM training course materials. We modified it for two different highlighting conditions:

    A. No highlighting on page.

   B. Highlighting in 5 different spots on the page.

Subjects were volunteers from a group of new managers attending management training school, so the web material was very relevant to their background, motivations and interests; each subject had extensive practice with these course materials before our test.

Each subject read through 2 practice pages, and had to answer several basic data collection questions (e.g., background, handedness, etc.) This practice session familiarized them with the physical setup of the Tobii eyetracker, the particulars of the mouse and our online experiment presentation system.

Subjects were told there would be a post study test of their retention of the content and that they should study the materials in anticipation of the test just as they had during their normal course of instruction. (To get to this point in the management class, our subjects had to work through 12 web-based modules, each averaging 15 pages of content. They were well practiced with this material.)

There were 9 subjects in condition A (4 male, 5 female) and 7 subjects (4 male, 3 female) in condition B. Our recording system captured the screen recording, DOM, and eye gaze for each participant.

For this reading analysis, we chose to concentrate on the
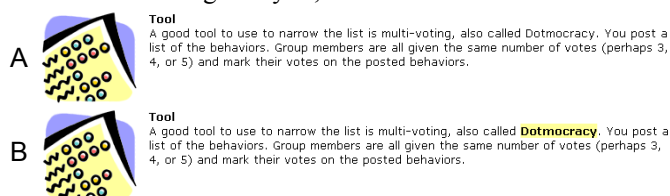
Figure 8 Highlighting test paragraphs for conditions A and B.

highlighted (or non-highlighted) word "dotmocracy" in the reading material. ("Dotmacracy" is a neologism in the course materials and describes a particular voting technique using small sticky dots. This is the first time our subjects would have encountered the term.) The paragraph containing "dotmocracy" is shown for both conditions in Figure 8 – it is highlighted with a bright yellow highlight in condition B but not in A. All subjects had normal color vision and reasonable visual acuity.

The post test contained the target question about dotmocracy: "To narrow a long list of items to a fewer number, team members are each given the same number of votes, and they mark their votes on one or more of their choices. This process is called _____." Participants were asked to fill in the blank and had to recall the term. (We graded answers strictly: obvious misspellings of the word were graded as correct, but correct answers had to include "dot" and a close variant of "-mocracy.")

|	Condition A No highlight	Condition B Highlighted text
N subjects	9	7
Ave time reading	0.8 sec	1.43 sec
Number of reads	1.86 reads	1.5 reads
Ave speed	203 pixels/sec	103 pixels/sec
Retention test	1/9 correct	3/7 correct

Table 1 Reading analysis results for highlighted text;

Using WebGazeAnalyzer, we measured total reading time, retention in the post test, the number of times the word was read, and the average reading speed. The word "dotmocracy" was labeled with a B-tag, making it very simple to get the following data shown in Table 1.

RESULTS

The differences between average reading time for the single word "dotmocracy" are significant at the 0.06 level (one-sided t-test). While reporting reading speed in pixels/sec in non-traditional, the key point here is the comparison across conditions. We could also use the DOM word bounding boxes to map pixels to words and report the answer in words/minute at a very fine grain of behavioral analysis. We can also determine the number of words that were actually read vs. those skipped.

As can be seen in Figure 9, the longer eye gaze time on the
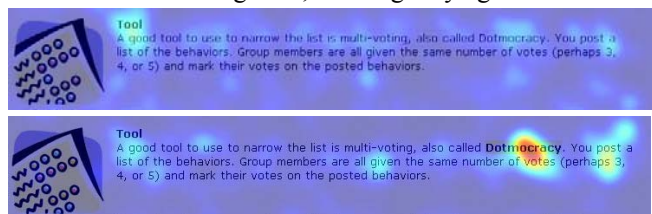
Figure 9 Both A and B conditions shown as heat plots summarizing all subjects in each condition. Color shift toward red indicates longer cumulative fixations, clearly showing the longer dwell time on the word dotmocracy.

word "dotmocracy" on the right hand side of the page clearly shows up. Intriguingly, when we examine other low-frequency words that appear on this page, there are no such lingering dwell times, perhaps because those words are not flagged by highlights as being salient to the impending post-test task.

DISCUSSION
It is well known that highlighted text can dramatically affect overall reading performance and that inappropriate highlighting (that is, text that has been highlighted for some other task) degrades overall reading comprehension [18]. Contrariwise, appropriate highlighting of text can significantly improve task performance, especially when the highlighting is task supportive at a detailed level (such as highlighting a search term in a field visual field of very similar objects) or supporting reading behavior for specific tasks [22].

So in a sense, these results are not surprising. However, we show here that highlighting has increased reading time, but it has done so by *slowing the reader down* while reading the body of the text, instead of causing additional re-reads of the text. Initially, we expected to see an increased number of reads for condition B, so this result surprises us. But the speed and reading time numbers balance each other out, so the data is self-consistent.

The increased retention for condition B is not surprising, but it is a nice result. If an educational web site wishes to increase retention for a few key points, these results imply that highlighting may be an effective tool. In condition B, the highlighting was fairly sparse and highly correlated to the task, even though the reader did not know the task (i.e., the test questions) in advance. Since the page was fairly long, highlighting was not overused. Other studies have shown that highlighting can be abused, as in the extreme case when virtually every sentence is highlighted [22] [23]. In such cases, distracting highlights cause targets to be less salient, with correlated reductions in overall comprehension and task performance.

CONCLUSION
We have presented WebGazeAnalyzer, a system for recording and analyzing web reading behavior. The WebGazeAnalyzer has several key capabilities:

A. It uses the web page DOM to represent text content organization and layout,

B. The gaze interpretation is interactive, open to detailed user walkthrough and annotation,

C. It provides an annotation capability for corrections, feeding back into the analysis itself,

D. With B-tags that can be assigned at any time, different hypotheses can be quickly explored, and post-data gathering re-analysis of the data.

Experience with the analysis tool shows that typical behaviors are immediately apparent. Visual search patterns indicative of different reader tasks (such as focused comparison tasks or open-ended scanning behavior) are obvious by their distinctive patterns of eye movement.

In future work we plan on exploring these visual behavior patterns of behavior that seem to become evident when exploring data with the WebGazeAnalyzer. We believe that it might be possible to automatically determine by eye gaze behavior if the web reader is actively pursuing a task goal, and to tell some of these goals apart. By using the tool, we hope to be able to tightly characterize these and other behaviors that we see across a wide population of web readers.

REFERENCES
1. Heer, J. Capturing and Analyzing the Web Experience. In *Proc. CHI 2002*, ACM Press (2002).

2. Chi, E.H., Rosien, A., and Heer, J. LumberJack: Intelligent Discovery and Analysis of Web User Traffic Composition. In *ACM-SIGKDD Workshop on Web Mining for Usage Patterns and User Profiles*, (2002).

3. Card, S.K., Pirolli, P., Van Der Wege, M., Morrison, J.B., Reeder, R.W., Schraedley, P.K., Boshart, J. Information Scent as a Driver of Web Behavior Graphs: Results of a Protocol Analysis Method for Web Usability. In *Proc. CHI 2001*, ACM Press (2001), pp. 498-505.

4. Reeder, R. W., Pirolli, P., Card, S. K. WebEyeMapper and WebLogger: tools for analyzing eye tracking data collected in web-use studies. In *Proc. CHI 2001*. Extended abstracts on Human factors in computing systems. (2001)

5. Chi, E.H., Rosien, A., Supattanasiri, G., Williams, A., Royer, C., Chow, C., Robles, E., Dalal, B., Chen, J. and Cousins, S. The Bloodhound Project: Automating Discovery of Web Usability Issues using the InfoScent Simulator. In *Proc. CHI 2003*, ACM Press (2003).

6. Poynter Institute and Eyetools, Inc. Eyetrack III: Online News Consumer Behavior in the Age of Multimedia.
http://www.poynterextra.org/eyetrack2004/index.htm

7. Burke, M., Gorman, N., Nilsen, E., Hornof, A. Banner Ads Hinder Visual Search and Are Forgotten. In *Proc. CHI 2004*, ACM Press (2004).

8. Halverson, T., and Hornof, A. Link Colors Guide a Search. In *Proc. CHI 2004*, ACM Press (2004).

9. EyeLink II eye tracker, SR Research. http://www.eyelinkinfo.com/

10. Eyegaze System, LC Technologies. http://www.eyegaze.com/

11. Tobii 1750 Eye-tracker, Tobii Technology, http://www.tobii.se/

12. Clearview Eye Gaze Analysis Software, Tobii Technology, http://www.tobii.se/

13. Eyetools, http://www.eyetools.com

14. Camtasia Studio, TechSmith Corporation, http://www.techsmith.com

15. Rayner, K. and Pollatsek, A. The Psychology of Reading. Lawrence Erlbaum Associates, Publishers. Hillsdale, NJ. 1989.

16. Salvucci, D.D. and Goldberg, J.H. Identifying Fixations and Saccades in Eye-Tracking Protocols. *Proc. of the Symposium on Eye Tracking Research & Applications* (ETRA), 2000.

17. Hornof, A.J., Halverson T., Cleaning up systematic error in eye-tracking data by using required fixation locations. Behavior Research Methods, Instruments & Computers 1 November 2002, vol. 34, no. 4, pp. 592-604(13).

18. Silvers, V.L., Kreiner, D.S. (1997). The Effects of Pre-Existing Inappropriate Highlighting on Reading Comprehension. Reading Research and Instruction, 36(3), pp. 217-23.

19. Collewijn, H., Steinman, R.M., Erkelens, C.J., Pizlo, Z., & van der Steen, J., Effect of freeing the head on eye movement characteristics during three-dimensional shifts of gaze and tracking. Chapter 64 in The Head-Neck Sensory Motor System, Berthoz, A., Graf, W., & Vidal, P.P., Eds. Oxford University Press, 1992.

20. Ware, Colin and Mikaelian H.T. "An Evaluation of an Eye Tracker as a Device for Computer Input" Proceedings of ACM CHI+GI'87, 183-188 (1987).

21. Jacob, R. J.K. The Use of Eye Movements in Human-Computer Interaction Techniques. ACM Transactions on Information Systems 9, 3, 1991, 152-169.

22. Wu, J., Yuan, Y. Improving searching and reading performance: the effect of highlighting and text color coding. Information & Management, 40, pp. 617–637, 2003.

23. Fisher, D.L., & Tan, K.C. (1989). Visual displays: The highlighting paradox. Human Factors, 31, 17-30.

24. Lankford, C., Gazetracker: software designed to facilitate eye movement analysis. *Proc. of the Symposium on Eye Tracking Research & Applications* (ETRA), 2000.