ChangeDetector™: A Site-Level Monitoring Tool for the WWW

Vijay Boyapati, Kristie Chevrier, Avi Finkel, Natalie Glance,
Tom Pierce, Robert Stockton, Chip Whitmer
WhizBang! Labs
4616 Henry St., Pittsburgh, PA 15217 USA
1-412-683-8596
{vboyapati|kristie|afinkel|nglance|tpierce|rstock|cwhitmer}@whizbang.com

ABSTRACT

This paper presents a new challenge for Web monitoring tools: to build a system that can monitor entire web sites effectively. Such a system could potentially be used to discover "silent news" hidden within corporate web sites. Examples of silent news include reorganizations in the executive team of a company or in the retirement of a product line. ChangeDetector, an implemented prototype, addresses this challenge by incorporating a number of machine learning techniques. The principal backend components of ChangeDetector all rely on machine learning: intelligent crawling, page classification and entity-based change detection. Intelligent crawling enables ChangeDetector to selectively crawl the most relevant pages of very large sites. Classification allows change detection to be filtered by topic. Entity extraction over changed pages permits change detection to be filtered by semantic concepts, such as person names, dates, addresses, and phone numbers. Finally, the front end presents a flexible way for subscribers to interact with the database of detected changes to pinpoint those changes most likely to be of interest.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web based services.
G.3 [**Probability and Statistics**]: statistical computing.

General Terms

Algorithms

Keywords

Machine learning, classification, information extraction, intelligent crawling, URL monitoring.

1. INTRODUCTION

Web monitoring tools [Acke97, Dedi98, Doug98, Fish97, Glan01, NetM, Spy, WebS] have played an important role since the early days of the WWW. By automatically tracking changes in web pages, these tools allow users of the Web to more easily keep up to date with dynamic content. With the advent of the mobile Web, monitoring tools have found another niche: notification via mobile phone or PDA of changes in stock prices, weather alerts, news about specific companies or markets, the availability of a product, and much more. Current Web monitoring tools have been tailored for these kinds of applications and are able to filter out some uninteresting changes, such as timestamp changes or changes in ad banners. Some tools are also able to selectively monitor sections of a page, changes that include certain keywords, and structured changes (for sites with a database backend).

One very important application of Web monitoring is for competitive intelligence. Large companies typically have an individual or team responsible for tracking competitor developments. Competitor web sites are an important source of "silent news": company news that goes unreported in the press, or perhaps is reported, but with a delay of days or weeks. Typically, individuals responsible for competitive intelligence will spend a large fraction of their time scouring company web sites for this type of information. Current monitoring tools can help, but are designed to monitor individual web pages, not entire sites. Furthermore, only a small number of changes are likely to be of interest. For example, an individual responsible for competitive intelligence may want to know when there is a change in the executive team of a competitor (especially an unreported one!) or when a new product is released or an old one is retired, but may not wish to be alerted whenever a product specification is updated.

The challenge, then, is to build a system that can monitor entire web sites and selectively report on changes relevant to a particular user. In this paper, we report on a system called ChangeDetector, which uses machine learning techniques for intelligent crawling, page classification and entity extraction to filter detected web site changes and report them to the user. Subscribers to ChangeDetector are able to query the system in complex ways. An example query is: "Return all changes from *the past week* in *health industry* sites that appear on *executive biography* pages and include *person names* or *job titles*."

The novelty of ChangeDetector is the generic application of intelligent crawling, classification and extraction for the filtering of changes to Web sites (in particular, company Web sites). Also, to the best of our knowledge, this is the first report of a WWW monitoring tool capable of monitoring the textual content of sites up to an arbitrary depth and that can filter changes according to both extracted entities and page classification.

Furthermore, in contrast to previous monitoring tools, ChangeDetector provides customizable filters for specifying which changes are of interest. For example, the user can filter changes by topic of interest, by type of extracted entity in the changed region, and by time range when the change occurred. The ability to filter changes selectively is paramount when monitoring entire web sites, since it is common to see thousands of changes occurring over hundreds of pages on a weekly basis on large sites. Without mechanisms to select which changes are of interest, users would be swamped by a large number of uninteresting changes.

WWW 2002, May 7–11, 2002, Honolulu, Hawaii, USA.
1-58113-449-5/02/0005.

Current Web monitoring systems typically can only monitor a fixed number of specified pages from a site and are limited to reporting on changes to entire pages, subsections, or around keywords. Exceptions include monitoring systems that work directly with sites that have database back-ends.

The paper is organized as follows. First we provide a system overview, describing the high level architecture of ChangeDetector. Then, in Section 3, we describe the pipeline of machine learning tools used to filter changes: intelligent crawling, page classification, and entity-based change detection. In Section 4, we discuss how the user interacts with ChangeDetector to pinpoint changes likely to be relevant. Related work in Web monitoring is presented in Section 5. Finally, we conclude in Section 6 and discuss future directions for this work.

2. ChangeDetector OVERVIEW

An overview of the ChangeDetector architecture is shown in Figure 1 below. Each component is summarized below, while detailed descriptions are provided in Sections 3 and 4.

The back-end components are: crawling, classification, entity-based differencing, verification and e-mail notification. The front-end components are the client user interface and the administrative interface. The components communicate with each other through data sharing via a database.

The schema also shows the flow of processes. At the back-end, a set of sites is submitted to the crawler. All pages crawled are then classified into a taxonomy of categories. Classification is incremental, so that only newly crawled pages are classified. Entity-based change detection is then run on all sites that have been crawled at least twice. Entity-based change detection looks for changes in web pages that include semantic entities, such as person names, job titles, dates, and company names. The changes can then be run through an optional verification phase, in which people assess the significance of the detected changes. Currently, support for this process has not been implemented, as our current goal is to achieve a zero human-intervention system. Finally, an e-mail process is triggered which sends reports by e-mail to those subscribers who have requested to be informed by e-mail.

The automatic generation of site maps proceeds in parallel to the crawling and classification processes. Site maps for each site are automatically built using the lists of URLs crawled and their page classification. The site maps are used to learn an intelligent crawl strategy for subsequent crawls in order to focus on areas of interest to subscribers.

The front-end user interfaces are comprised of the client UI and the administrative UI. The client UI provides three main functions: subscription; reporting; and search. Subscribers have access to a rich set of properties for customizing their report. They can access current change reports as well as previous change reports. Finally, they can search over all changes to all sites according to page category and entity type. The administrative user interface allows the administrator to create and modify new user accounts, to monitor crawl behavior and to modify crawl strategies.

Currently, we have a working prototype capable of monitoring two thousand sites weekly (on a Pentium III with 1 Gigabyte of RAM). Screenshots of the working system are shown in Section 4.

3. MACHINE LEARNING TOOLS FOR CHANGE DETECTION

In this section we will describe the principal backend components of ChangeDetector: crawling, classification and entity-based change detection. All three components use machine learning techniques. Intelligent crawling depends on the automatic creation of site maps from classified crawled pages. Classification of pages uses known algorithms for classifier training along with new twists in feature engineering. Entity-based change detection combines XML file differencing with entity extraction to yield a fundamentally new way to filter changes in Web pages.

3.1 Intelligent Crawling

During development, we have been testing ChangeDetector against Fortune 1000 Web sites in addition to a number of selected companies from certain industries. The default crawling policy is to crawl up to 1000 pages for each site using a breadth-first search strategy. Crawl statistics show that the number of pages crawled per site follows a Zipf-like distribution, with a large percentage of sites with very few pages and a small number of sites with many pages (the log of the distribution is linear). The Zipf distribution fails to hold near 1000 due to the page limitation set on the crawler. The average number of pages per site in our development set is 300, but the median is much lower: 50% of the sites have less than 109 pages. Thus, for the majority of sites, the default crawling strategy is sufficient.

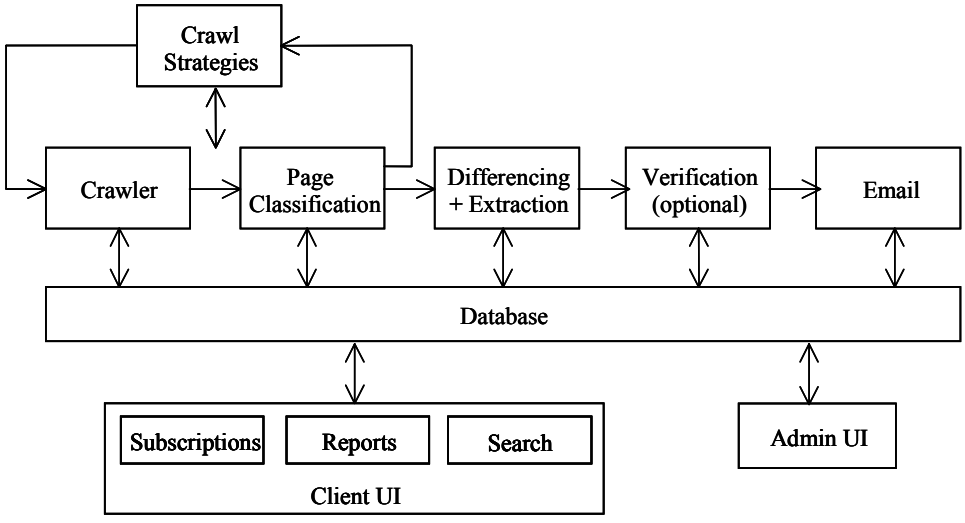

Figure 1. ChangeDetector Architecture

However, for large sites, there are two problems with the default policy. First of all, we will achieve only a partial crawl for companies with more than 1000 pages (these comprise about 3% of sites we crawled). Thus, we will not be able to effectively monitor sites with thousands of pages. A related problem is how to ensure that the same pages are crawled in successive crawls. Because the hyperlink structure of the site may change, the default approach could end up crawling very different sets of pages from one crawl to the next.

Secondly, with a breadth-first crawl, the spider may end up retrieving a high proportion of pages not likely to be of interest to subscribers. For example, retail organizations frequently have on-line catalogs. It is likely that monitoring all items for sale at Walmart, say, is not of interest. However, the default crawler may end up spidering many pages of the catalog at the cost of possibly missing pages in categories of interest to subscribers, such as investor relations or press release pages. There is also an efficiency cost. If the part of the Web site of interest to subscribers is a small subset of the entire site, then large portions of the site may be avoided during the crawl.

Thus, the problem of efficiently and effectively crawling large sites points to a need for intelligent crawling that focuses on sections of Web sites with content types that ChangeDetector subscribers wish to monitor. Our solution is two-fold. The first part involves classifying each page crawled into one of many categories. Section 3.2 below describes both the taxonomy of categories and the classification algorithms employed.

The second part of the solution is the automatic creation of a site map for each site that takes into account the classification label for each page. From the labeled pages, a site map is inferred which groups similarly labeled pages by their common URL directory structure. Using the site map, each category can then be mapped onto a set of directories. The crawler uses this information to decide which parts of the site to preferentially crawl or preferentially exclude. If the goal is to preferentially crawl pages falling within the press release category, for example, then all directories which are associated with pages identified as press releases will be preferentially crawled.

To generate crawl strategies to send to the crawler, the intelligent crawling process first queries the database to retrieve the default taxonomy of categories. It then queries for all pages retrieved in the previous crawl that were classified into one of these categories. Next, a site map is generated for each site. The site map describes the directory structure of the previously crawled and labeled pages. The directory structure is automatically inferred from the URLs of the pages crawled. For example, if the previous crawl found the following "press release" pages:

- www.domain.com/press/2001/pr1.html
- www.domain.com/press/2001/pr2.html
- www.domain.com/press/2001/pr3.html

then a parent directory www.domain.com/press/2001 is inferred. The following partial tree structure is produced:

- www.domain.com/press/2001/
    - pr1.html
    - pr2.html
    - pr2.html

The intelligent crawling process then instructs the spider to preferentially crawl paths beginning with www.domain.com/press/2001/. Potentially, the spider might now reach other press release pages missed in the previous crawl.

The intelligent crawling process also identifies the paths that lead only to URLs *not* classified into one of the categories in the taxonomy. As a result, it is able, for example, to instruct the spider to not crawl on-line catalogs.

The output of the intelligent crawling process is a set of instructions whose syntax is reminiscent of robots.txt files. The __DISALLOW__ keyword tells the spider to not crawl URLs matching the path. The __PRIORITIZE__ keyword instructs the spider to give priority to URLs matching the path. There is also an __ENQUEUE__ keyword which enforces individual subscriber requests to crawl specified URLs.

Here is a sample taken from an automatically generated set of crawler directives:

```
http://www.interelate.com/
        __DISALLOW__=/contact_us/map.asp
        __PRIORITIZE__=/company
        __PRIORITIZE__=/careers/missionv
        alues.asp
        __PRIORITIZE__=/contact_us/email
        form.asp
        __PRIORITIZE__=/news
        __PRIORITIZE__=/partners
        __PRIORITIZE__=/company
```

This set of instructions steers the spider towards executive information (/company), contact pages (/contact_us), and press releases (/news), among other categories falling within the taxonomy currently instantiated in ChangeDetector.

The intelligent crawling process is run periodically, generally before each new crawl, although this is not strictly necessary. The first time a site is crawled the default crawling policy applies. However, subsequent crawls are guided by the map generated for the site.

The automatically generated site map also indicates the typical amount of clustering within given categories of pages. We define the degree of clustering by the proportion of inferred directory paths required to cover all URLs in a category. For example, we found career information, financial information, product information and press releases to be highly clustered into specific site directories. Press releases were the most highly clustered with approximately a 1:10 ratio between directories and pages. On the other hand, we found executive information, legal information, staff pages, company relations information, and contact information to be more dispersed across the site. Legal information, for example, has a 1:1.4 ratio between directories and pages. Finally, the average site map for all categories taken together is highly clustered with a 1:5 ratio. On the whole, these results indicate that the site map tool is able to effectively summarize the portions of sites to preferentially spider.

Previous work on intelligent spidering has explored, on the one hand, using reinforcement learning to guide decision-making during crawling [Ren99] and, on the other hand, using page classification during crawling [Chak99].

3.2 Text Classification

Text classification is the task of assigning textual documents to one or more predefined classes. It is a significant problem whenever a large number of documents are available and automatic organization of the documents would be useful. For instance, it would be useful to have the documents available on corporate web sites automatically organized into predefined classes such as "Press Releases," "Executive Biographies" or "Financial statements" so that a user could quickly identify a topic of interest and focus their search to documents within that topic. In the ChangeDetector system, classification is one of a number of components employed so that users of the system can provide expressive queries to specify which changes are of interest.

The taxonomy used in ChangeDetector was chosen to provide an organization of documents on corporate web sites that captured the typical semantic breakdown of pages on such sites, while not being specific to any particular industry sector.

The taxonomy used in ChangeDetector has 12 separate semantic classes:

- *Press releases.* Pages that contain a public announcement by the company, intended for the press.
- *Executive biographies.* Pages that contain biographical information about the company's executives.
- *Company profile information.* Pages that contain information about the company's vision, history, business model or strategic direction.
- *Company relations.* Pages that contain information about the company's business relationships, relations to customers or strategic partnerships.
- *Financial information.* Pages that contain information about the company's financial performance.
- *Contact information.* Pages that contain the contact details of the company. For example, the company's primary phone and fax numbers, and their email or postal mailing address.
- *Product or service pages.* Pages that contain information about products or services that the company offers to its customers.
- *Careers information.* Pages that contain information about job vacancies or employment opportunities at the company.
- *Company outreach.* Pages that contain information related to the company's participation in or support of the wider community. Examples include information about academic scholarships, fund raising events and pro bono legal work.
- *Legal notices.* Page that contain legal statements. Examples include privacy statements, terms of usage, and copyright statements.
- *Events and conferences.* Pages that contain information about events or conferences that the company plans on organizing. Examples include trade fairs, open days and technical seminars.
- *Foreign language pages.* Pages written in a language other than English.

Note that the classes need not be mutually exclusive. For example, a page may be both a press release and contain information about the company's financial performance.

3.2.1 The method used for classification

In the ChangeDetector system we employed a machine learning approach to build classifiers for each semantic class in the taxonomy. We provided training examples for each class consisting of both positively and negatively labeled documents. A learning algorithm was then applied to the list of examples to produce a model for each class. Each model could then be applied to a new document to produce a probabilistic confidence specifying the likelihood that the document belonged to the associated class.

3.2.2 Feature engineering

Before a learning algorithm can be applied to the set of training documents, the documents must be converted to a representation that can be manipulated by a learning algorithm. The representation chosen in ChangeDetector is the feature vector. A feature vector is simply a vector of the features occurring in a document and counts representing how many times each feature occurred. Typically in text classification, the features extracted are simply the word tokens from a document. However, the process of extracting features may include more complex analysis, such as information about geometry and layout.

The set of features used in the ChangeDetector system is richer than the standard bag-of-words features that are typically used in text classification. Some of the extra features that we extract from documents include character n-grams from the URL, emphasized words, such as bold-faced or italicized words, anchor text (as studied by Ghani et al. [Ghan2001]), semantic entities, such as dates or locations, and the positional or structural association between the extracted entities. An example of a positional feature is: "Does a date occur next to a location?" The occurrence of this feature is particularly indicative of a Press Release. We found that by making use of the extra, richer space of features, the models produced were able to classify documents much more accurately.

3.2.3 The learning algorithms

The machine learning algorithms we found to be most accurate were Boosted Decision Trees, and Exponential Models trained using the maximum entropy method. Boosting is a technique that can be used to improve the performance of other learning algorithms [Scha99]. In our case we used it to improve the performance of Decision Tree based models. We also found that Exponential models provided the best classification performance for some of our classes such as Product pages. Exponential models were studied by McCallum and Knigam for text classification [Nigam99]. However, unlike the Iterative Scaling training algorithm they reported using, we used the Conjugate Gradient algorithm to find the optimal Exponential Model [Shew94]. Minka provides empirical evidence suggesting that for the same computational cost, Conjugate Gradient is much more effective at producing the optimal Exponential model than the Iterative Scaling technique [Mink01]. Figure 2 illustrates the precision-recall performance we achieved on the Press Release class. The curve is typical of the performance we achieve on many of the classes in our taxonomy.

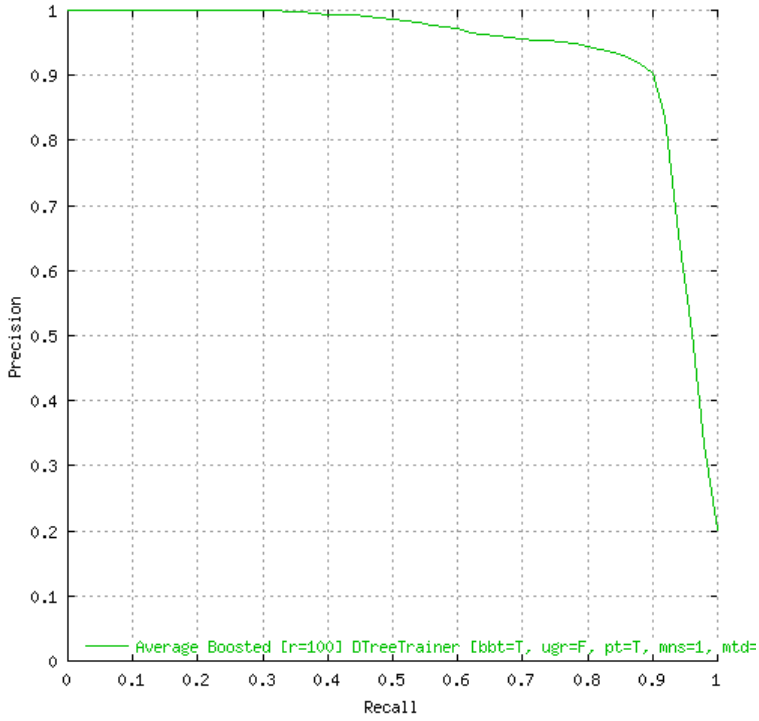

Figure 2. Precision recall performance for press release class

3.3 Entity-Based Change Detection

After classification, a site is passed to the entity-based change detection subsystem. First, a simple first-pass comparison is done on a per-page basis to see if the new and old versions of each page are character-identical. If a page has changed, it is then run through an XML-aware differencing algorithm and through a series of entity extractors. Each change is then associated with the nearest instance of each type of extracted entity.

The differencing algorithm alternately operates on pages as W3C DOM trees or plain text. Initially, for each node in the new and old DOM trees, a hash is computed using a normalized representation of the text that falls below that node. The normalized text will gloss over differences in whitespace or structure; such changes are common on database-driven sites that have changed their layout but not their content. If the hashes of the root nodes match, the documents are deemed equivalent, otherwise we attempt to align the child nodes.

Alignment of the child nodes is achieved by comparing a hash of the text that falls beneath each node. Nodes that remain unaligned will be marked as *insertions* or *deletions*. Nodes that generate unequal hashes but are aligned with a node representing the same type of markup are recursively compared. If two nodes are aligned but represent different types of HTML markup, we fall back on a token-based alignment strategy. If no significant matches are found on the token level, we simply mark all the text below the nodes as an *edit*.

This strategy handles most common changes very efficiently. Usually, when a web page changes, either the structure of the document or its content remains largely unchanged. If the structure remains intact, the change is quickly localized and the relatively expensive token alignment can be applied only to the affected subtree. If the content remains unchanged, the hashes will still match at the root. A further benefit of this structure-oriented approach is that it tends to generate changes that are well suited for displaying with in-band HTML markup.

After determining all the changes made to a page, we filter out uninteresting changes. Spelling corrections, pairs of equivalent insertions and deletions, and repetitions of earlier changes are all considered uninteresting. These strategies are each simple, but together they are very effective at reducing the volume of uninteresting and repetitious changes.

If a page has any changes that are not filtered out, it is run through a set of entity extractors. Our extractor library is built using a transduction library with a rich scripting interface, which allows users to describe machines in simple text files. The transducers that implement the extractors can be either engineered or trained [Hobbs96, McCall00]. The extractors that were used in the ChangeDetector prototype recognize company names, dates, email addresses, monetary amounts, phone and fax numbers, postal addresses, and person names.

After the extractors are run, each change is associated with a set of entities, representing the nearest entity of each type to that change. Distance is measured in words, with respect to a linear document view (i.e., flat HTML with markup elided). More advanced notions of distance, for example considering adjacency within a column of table cells, are possible, but were not implemented for the prototype. These associations, along with the page classifications, allow the system to provide powerful information about each change, such as "this change contains a person's name, is within five words of a company name, and appears on an executive biography page." All changes and extractions are entered into the database, as well as the distances between each extraction and the entities it was associated with, to support filtering for customer reports.

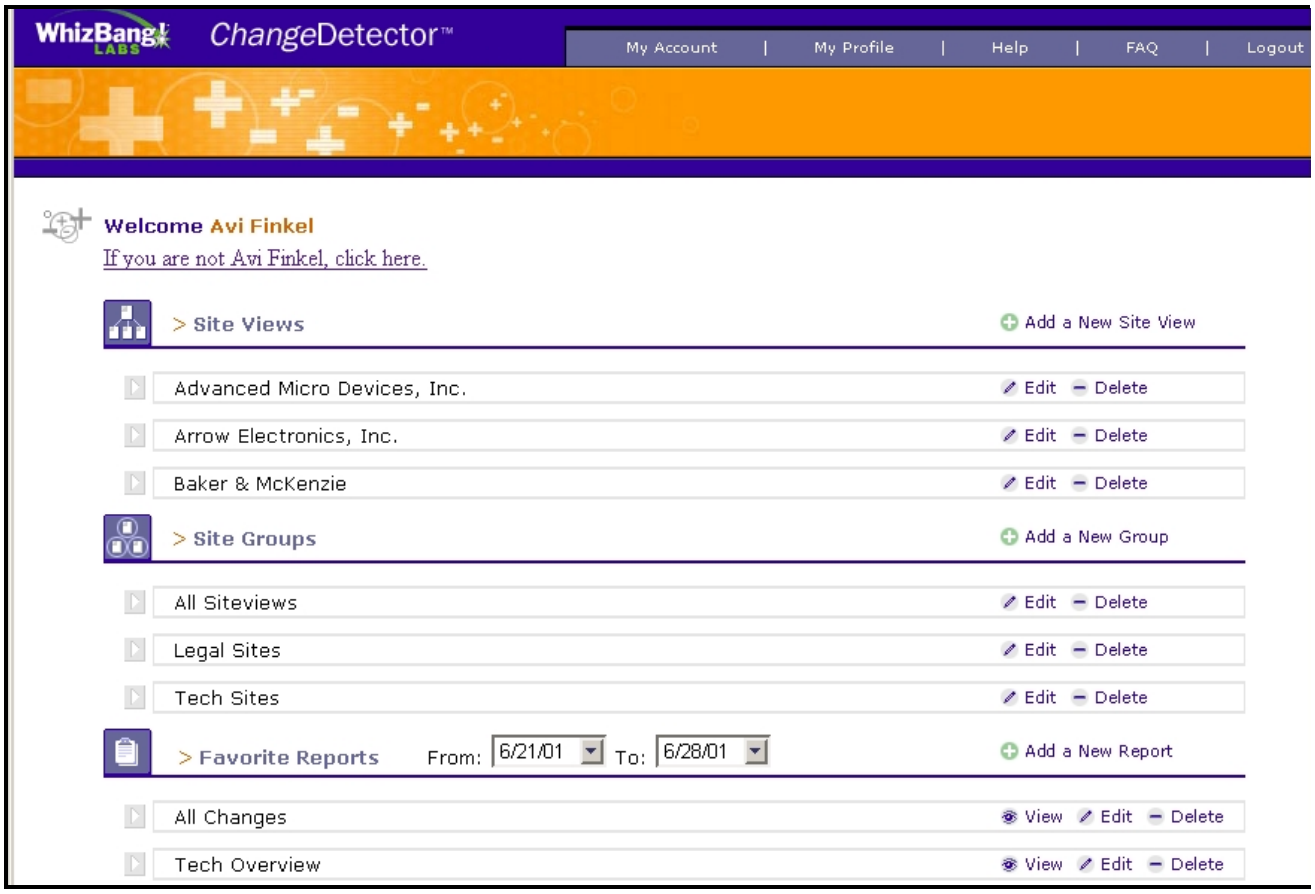

Figure 3. Account home page with customized site groups and site views

4. USER INTERFACE & CUSTOMIZABILITY

The driving idea behind the Change Detector user interface design was that it should be both highly customizable for the advanced user, but with default settings that provide immediate value. We have tried to keep it as flexible as possible without overwhelming users with too many options. It is also important to note that this interface has not yet seen any sustained use by outside customers – thus, it is a work in progress.

4.1 Subscriber Home Page

After logging in, the subscriber is taken to his/her account home page. The account home page, shown in Figure 3, allows the subscriber to customize his/her set of *site views*, *site groups* and *favorite reports*. These are explained in detail below. Initially, the account home page is empty apart for a default site group containing all site views and a default report covering all changes to all site groups.

In order to request ChangeDetector to monitor a particular site, the subscriber first creates a site view. A site view consists of the user-supplied name of the site view, a root URL, and some options that determine which pages within that web site will be monitored. Note that a user can construct several site views with the same base URL, but with different customizations. This is why a distinction is made between *sites* and *site views*. Users can monitor all of the pages on a web site or only a selected list of pages. Existing site views can be edited or deleted as needed.

A site group is a collection of site views. Defining a fixed set of these site groups makes it much easier to configure the reporting options over those sites. Creating a new site group is very simple – the user simply chooses a name for that group, and selects the site views that will go into it.

Figure 3 shows the account home page for a user who has already created two site groups: one for the technical sites and another for legal sites. The third site group is the default "All Siteviews" group, which exists for all users, and contains all of their site views. This is available as a convenience for users. Thus, at a minimum, a new user must create one or more site views, but need never create a site group. Below, we will explain how site groups are useful for customizing reports.

4.2 Customizing Reports

A report defines what data a user wishes to see about changes to a particular site or group of sites. Figure 4 shows the options a user has in customizing a report. Selecting the "All pages" option turns off change filtering by topic: the report will include changes from any page in the site. Alternatively, the user may choose specific types of pages for which to view changes. The left side of Figure 4 shows a user who has requested to view only changes on pages that have been classified as product, event, or career pages.

In addition, the user can choose which types of extracted entities to include in reported changes. The user may select all changes, or only changes that contain text that has been marked by an entity extractor. The right side of Figure 4 shows the actions taken by a user who wishes to see only changes containing either a person's

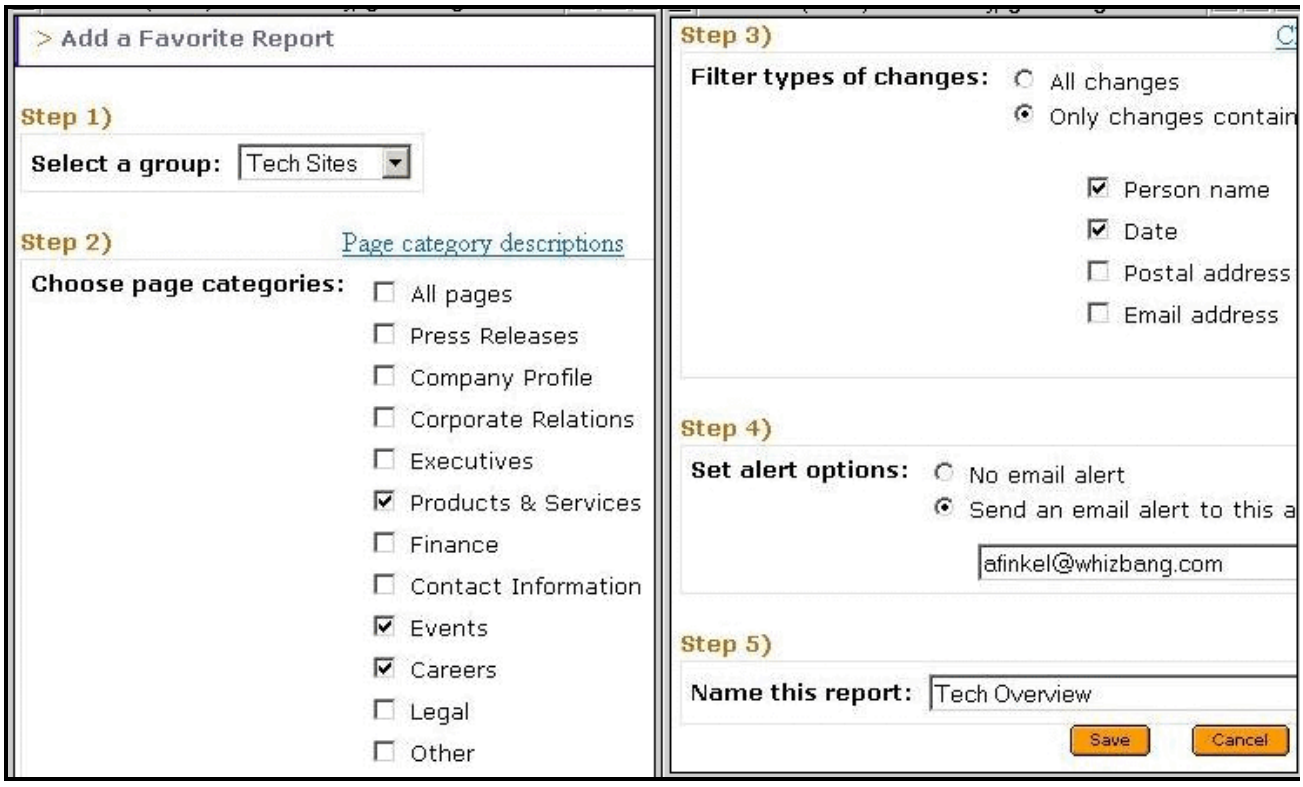

Figure 4. Customizing a report

name or a date. The report can then either be set up as a one-time execution, or as a recurring execution whose results will be emailed on a regular basis.

4.3 Viewing Reports

The user can now view the reports that he has customized. First, he selects the date range that he wishes to view from the drop-down boxes on his account home page, as shown in Figure 3. The dates listed reflect the dates of data runs that are available. (Note that at least two data runs are required to produce comparisons for any particular web site.) Once the desired dates are selected, the user clicks the *View* Button associated with the report he wishes to view.

An example of a ChangeDetector report is shown in Figure 5. At the top of the page is a table summarizing the statistics of the given report. In the example in Figure 5, ChangeDetector analyzed a number of web sites, including the web site for Commerce One, on April 11, and again on April 18. For Commerce One, ChangeDetector identified a total of 76 differences on 15 changed web pages. The number of changes reported corresponds to those that meet the topic- and entity-based filtering criteria.

Following the statistical summary is a detailed list of the changes found for each site in the report. Changes are sorted first by company, then by page category, and finally by page title.

Each page entry shows the page title, followed by a link to the *Current* page, and then by a dated link to the page for each data run available. Clicking on *Current* will point the browser to the live URL of the page in question. Clicking on one of the dated links will display the page as it existed on the given date, including markup to show changes and extracted fields, as shown in Figure 6.

The page title is followed by a list of all changes from that match the report's criteria, in order of occurrence. Newly added text is underlined, while deleted text is struck-out. A few words of the change's context are displayed in normal text. Each entry also includes a dated link to a cached version of the page where the change occurred.

Figure 6 shows the details of the report on all changes involving People in Press Releases and Executive Pages. In this case, ChangeDetector underscores an important addition: a new vice chairman of the board and chief operating officer for Commerce One. In addition, we see that ChangeDetector is still fooled by spurious changes: the addition of "Dr." in front of "Mary Loomis" is not successfully filtered out.

Each page entry includes a drop-down box listing all page categories. Each change entry is also followed by a *Delete* checkbox. These features allow users to provide feedback by re-classifying pages manually that they feel were not correctly classified and by deleting changes not of interest to them. This information can then be used by the system to improve the analysis on future change detection runs.

Figure 5. Change Detector Report

As mentioned before, clicking on a date link takes the browser to the cached page for that date. On cached pages, the newly added text is shown with a green background, while deleted text is stricken out. Extracted data fields, such as product names, dates, addresses and telephone numbers are highlighted. Figure 6 shows the cached page from Commerce One associated with the change in the executive team. The person names are marked up and the change is highlighted.

5. RELATED WORK

The first Web monitoring tools appeared in the early 1990s as commercial systems. Users were able to register URLs to monitor and were notified when the contents pointed to by those URLs had changed. URL-minder, now called Mind-It [NetM], was one of the first of such tools. NetMind offers its monitoring service for free as an Internet service. Users register URLs of interest on the Mind-It web site. In turn, the server fetches pages on a regular basis to detect changes and notifies the user by e-mail or via the WWW. They also have a number of customers who have integrated NetMind's monitoring facilities into their web sites. These customers include eBay, RedHerring and Siemens. For example, the auction alerts provided by eBay are powered by NetMind's monitoring technology. NetMind's main strength is the ability to key off database fields to identify changes.

Another server-side monitoring system is www.spyonit.com. It has pre-configured "spies" that alert subscribers to changes in structured information: new houses coming for sale in a certain area; news headlines from Pittsburgh containing the keyword "election," etc. The spies work off of database-driven sites. It is also possible using this service to monitor a page for any change. The service states that spurious changes, such as changes in timestamps and banner ads, are filtered out.

Other companies provide software that allows users to monitor web pages from their desktop machines. For example, WebSpector from Illumix [WebS] is a client-side tool that allows users to monitor a number of URLs.

Recent research work on monitoring tools has introduced novel architectures and features. WebTracker [Fish97] allows users to

Figure 6. Marked-up cached page

either share or maintain privately tracked URLs. It uses the Unix *diff* tools to show differences to users. Pluxy [Dedi98] introduces a novel architecture: Pluxy is a Web proxy that hosts a dynamically extensible set of services; the authors propose, among other services, a Web monitoring service.

The Do-I-Care agent [Acke97] employs relevance feedback. User feedback on their interest in the change detected by the agent is used to train the agent and improve its future performance. A Bayesian classification process is used to build a profile of the user interest and to score detected changes, which in turn are compared against a configurable threshold.

The AT&T Difference Engine [Doug98] incorporates sophisticated methods for performing difference analysis, using an adaptation of Hirschberg's solution for finding the longest common subsequence [Hirs77] to HTML pages. This work was extended in TopBlend [Chen00], an implementation of their HTML differencing tool in Java. It uses the fast Jacobson-Vo algorithm, which solves the Heaviest Common Subsequence problem, for page comparison. Their performance results indicate that TopBlend outperforms their previous HTML differencing tool in most time-consuming jobs, often by 1 to 2 orders of magnitude.

Website News [Chen99] performs a difference analysis on the hyperlink structure of web sites. The Website News interface displays newly added links to users along with their anchor text. However, the analysis does not include the textual content of the web pages.

Finally, Knowledge Pump [Glan01] integrates a document monitoring service into a Web-based recommender system. Based on early user evaluations of changes, the collaborative system can filter out uninteresting changes for other users. For each document, the system also maintains a version history of only those changes deemed interesting.

6. DISCUSSION

In this paper, we have presented ChangeDetector, a system for site-level Web monitoring. ChangeDetector incorporates machine learning techniques in order to search out significant nuggets of new information from ever-evolving web sites. The intelligent crawling component allows the crawler to focus on the most user-relevant portions of large sites. The classification component categorizes downloaded pages into content types. The entity-based change detection component compares versions of content referred to by the same URL at different dates in order to find changes associated with extracted entities, such as person names, job titles and company names.

The end product of the machine learning pipeline is a table of change information that is updated after each new crawl. Associated with each change is the content category of the parent page and the proximity to extracted entities, along with a reference to the parent site.

The user interface described in the paper allows subscribers to flexibly generate reports of changes that are likely to be relevant to them. For example, a subscriber can create a favorite report equivalent to: "Show me all changes from my health industry sites that appear on executive biography pages and include person names or job titles."

Refinements to the machine learning pipeline are under consideration. For example, text that appears on an executive biography page does not necessarily have to do with executive biographies. It would be useful to have sub-page level classification of text. Secondly, we are working on normalization of extracted entities. For example, if we can normalize on the person name level, then changes can be associated with individuals (e.g., John Smith) and not just person names (e.g., John E. Smith, John Smith, Mr. John Smith).

The user interface will evolve as we get feedback from pilot customers. The toolkit of extractors and classifiers is continuously being improved and extended and will help ChangeDetector further pinpoint significant changes in Web site content.

ACKNOWLEDGMENTS

Many thanks to Andrew McCallum, Tom Mitchell, Barney Pell and Dallan Quass for numerous discussions that helped shape the evolution of ChangeDetector. We also thank Timo Brimhall for

his contributions to the user interface and Brent VanZweden for his technical support.

REFERENCES

[Acke97] Ackerman, M., Starr B., Pazzani, M. "The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web." In: *Proc. of RIAO'97*, pp. 17–31.

[Chak99] Chakrabarti, S., Van den Berg, M., Dom, B. "Focussed Crawling: A New Approach to Topic Specific Resource Discovery." In: *Proc. of World Wide Web 8*, Toronto, Ontario, Canada, May 1999.

[Chen00] Chen, Y.-F., Douglis, F., Huan, H., Vo, K.-P. "TopBlend: An Efficient Implementation of HtmlDiff in Java." In: *Proc. of the WebNet2000 Conference*, San Antonio, TX, Nov. 2000, http://www.research.att.com/sw/tools/topblend/paper/index.html.

[Chen99] Chen, Y.-F., Koutsofios, E. "Website news: A website tracking and visualization service." In: *Poster Proc. of World Wide Web 8*, Toronto, Ontario, Canada, May 1999.

[Dedi98] Dedieu, O. "Pluxy: un proxy Web dynamiquement extensible." In: *Proc. of the 1998 NoTeRe Colloquium*, Oct. 1998, http://www-sor.inria.fr/publi/PPWDE_notore98.html.

[Doug98] Douglis, F., Ball, T., Chen, Y.F., Koutsofios, E. "The AT&T Internet Difference Engine: Tracking and Viewing Changes on the Web." In: *World Wide Web*, 1(1), 27–44, Jan. 1998.

[Fish97] Fishkin, K., Bier, E. "WebTracker - a Web Service for tracking documents." In: *Proc. of World Wide Web 6*, Santa Clara, CA, 1997, http://www.parc.xerox.com/istl/members/fishkin/doc/webtracker.html.

[Ghan2001] Ghani,, R., Slattery, S., Yang, Y. "Hypertext categorization using hyperlink patterns and meta data." In: *Proc. of the Eighteenth International Conference on Machine Learning (ICML'01)*, pp 178–185, 2001.

[Glan01] Glance, N., Meunier, J.-L., Bernard, P., Arregui, D. "Collaborative Document Monitoring." In: *Proc. of the 2001 Int'l ACM SIGGROUP Conference on Supporting Group Work*, Boulder, CO, 2001.

[Hirs77] Hirschberg, D.S., "Algorithms for the longest subsequence problem." *Journal of the ACM*, 24(4), 664–675, Oct. 1977.

[Hobbs96] Hobbs, Jerry R., Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. "FASTUS: A cascaded finite-state transducer for extracting information from natural-language text." In: *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA.

[McCall00] McCallum, A., Freitag, D., Pereira, F. "Maximum entropy Markov models for information extraction and segmentation." In: *Proc. of ICML-2000*, 2000.

[Musl99] Muslea, I. "Extraction Patterns for Information Extraction Tasks: A Survey." In: *Proc. of AAAI'99 Workshop on Machine Learning for Information Extraction*, Orlando, FL, 1999.

[Nigam99] Nigam, K., Lafferty, J., McCallum, A. "Using Maximum Entropy for Text Classification." In: *IJCAI'99 Workshop on Information Filtering*, 1999.

[NetM] NetMind, http://www.netmind.com/

[Mink01] Minka, T. "Algorithms for maximum-likelihood logistic regression." Technical Report, 2001. http://www.stat.cmu.edu/~minka/papers/logreg.html

[Ren99] Rennie, J., McCallum, A. "Using Reinforcement Learning to Spider the Web Efficiently." In: *Proc. of ICML'99*, 1999.

[Scha99] Schapire, R. E. "A brief introduction to boosting." In: *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[Shew94] Shewchuk, R. "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain." Technical Report, 1994. http://www-2.cs.cmu.edu/~jrs/jrspapers.html

[Spy] SpyOnIt, http://www.spyonit.com/

[WebS] WebSpector, http://www.illumix.com/.