Abstract

Gift registries take the guesswork out of gift-giving by enabling registrants to specify the items they wish to receive. However, purchasing a gift for someone who is registered at one or more retailers requires knowledge of which of the person’s gift registries actually contain items within some desired price range. Existing “universal gift registries” address this situation by allowing registrants to create a single, replacement registry at the universal gift registry’s website. The Gift Registry Aggregator has the added benefit of allowing users to keep registries at multiple third-party retailers. By utilizing a user-provided link to each gift registry, the aggregator automatically updates as the items are bought off the third-party site’s registry. The summarizing page allows the items to be grouped by store, by price, by user-supplied category, and alphabetically by name. Each item has a link to enable its purchase from the registry on the third-party site. The registrant can also add and delete items that are not associated with a third-party registry.

Introduction

Gift registries are a key marketing tool in retail industries, providing a convenient way to channel gift purchases toward a specific retailer, and drawing traffic to retailers’ stores and websites. They also make gift-giving easier, by communicating to the gift-giver exactly what items the user wishes to receive. This eliminates the hassle of returning unwanted gifts to the store.

There are a number of websites that allow the user to construct a so-called “universal” gift registry by either manually pasting links to an item or by pressing a button to mine a single-item from the page. The inability to allow the user to maintain registries at multiple retailers may be disadvantageous for those gift-receivers who enjoy creating registries in-store, or those gift-givers who prefer buying for the recipient in-store or directly through the retailer-hosted registries.

There are number of simplistic sites that claim the “universal gift registry” title. www.mygiftlist.com allows gift-recipients to create a list of gifts they would like to receive by manually linking to the web site where the gift can be purchased. There are no product pictures on the site. Similarly, www.thingsilove.org.uk allows users to manually add descriptions of items that they wish to be purchased on their behalf. This website asks users to email the site if they wish an existing, third-party gift list to be “imported” into their system.

Several other websites provide more enhanced functionality. www.myregistry.com allows the gift recipient to automatically add an item from any website, providing a toolbar-installable button that adds the main item on the page to the gift registry. While some partner stores have a "quick checkout" option, at most sites, the gift-giver clicks on "buy" and, after the site confirms the item’s availability,
the buyer is asked to confirm whether or not the item was actually purchased. Another site, weddingchannel.com, allows users to provide links to “partner” websites where users have gift registries; however, the website does not actually bring those lists together into one combined list. Wedding Channel also allows users to search partner websites for items, providing a link to sign up for a registry with that website.

A useful, related service, provided by www.giftregistrylocator.com, allows potential gift givers to search multiple websites that have gift registries to see where the party has a gift registry.

Many of the above websites require the manual creation of a gift list from scratch, even if it allows the purchase of items from other websites. Rather than simply providing a link to the registry, the Registry Aggregator dynamically imports the items off of the third-party registry. This allows the user to retain the advantages of maintaining third-party lists: would-be gift givers who prefer to shop in-store, store promotions that incentivize gift registry use through their store, or streamlined store-specific registry creation procedures (such as the quintessential “scanning-gun” device used in-store to add items to a registry by scanning UPCs). After importing the items, the user can add comments and specify categories for the items.

Related Works

An important component of the Gift Registry Aggregator is the ability to retrieve data records from HTML documents or web pages. Though no truly general parsing scheme was incorporated into the final product, several candidate methods were considered for adaptation for the purposes of the project, and one, PAT-trees, was attempted. Many of these methods varied in their mode of parsing. Some general categories were both supervised and unsupervised methods, string matching, and tree alignment methods.

One supervised approach to data extraction from web-pages is called “wrapper induction”. In such methods, a user generally provides numerous examples of labeled pages, from which a parsing scheme is automatically induced. Lim and Lam (2000) describe a method whereby, from labeled examples of data records in a page, a set of syntactic rules (right pattern, left pattern, and features of the data itself) are automatically inferred to extract individual records and their component details. Their method determines all possible fields for each record, and then produces rules for extracting the data from each record. This method could be harnessed by providing a marked-up page of any new gift registry to be incorporate into the system. Rather than relying on an automated method to determine all possible fields, the system could allow the user to indicate on a list which fields may be included on the page, and later explicitly assign names to the fields.

The first of two supervised tree-alignment methods of interest is described by Liu and Zhai (2005). The first step in the method was the alignment of data records by using visual rendering information, which was mostly useful for malformed HTML. The second step was the alignment of corresponding data fields from multiple records to be stored database table using a novel partial tree
alignment technique to align corresponding fields. Such tree alignment can provide better results than simple string alignment of HTML tags.

A supervised, tree-alignment method called Thresher (Hogue and Karger, 2005) determines the DOM tree structure of fields of interest for given datasets. By the use of relatively few examples, wrappers are induced by using the edit distance between example DOM subtrees. The patterns in these wrappers are used extract related entries from a web site. One particular heuristic used was that of "lists"-- the idea that, when aligning two HTML subtrees, one may contain more interior entries that match a given field. Neighbor nodes of the example node chosen to train the wrapper are used to suggest other sample trees for the alignment algorithm. The paper went further to describe a simple to use, end-user system for defining wrappers within a web browser, providing a good example of what a user might expect in being able to aid the scraping of a new webpage.

Another supervised method uses pattern discovery to perform supervised extraction from a different direction. In the IEPAD (Information Extraction for Pattern Discovery) system, the products of candidate patterns for extracting data are presented to the user (Chang et al, 2003). The user then selects which of the patterns provide the best results. The patterns are generated by the use of PAT-trees (Gonnet et.al., 1992) to discover maximal repeats. This is accomplished by encoding block-level (non-text) HTML tags into binary codes, constructing the suffix tree with special indicators, and then, for a minimum repeat count k and pattern length, traversal of the tree in post-order to detect patterns (Chang et al, 2003). The advantage in this approach is that it allows the user to easily select between suggested record lists (involving just a few clicks), rather than requiring them to laboriously label numerous examples.

An unsupervised method, described by Alvarez et al. (2007), first locates the data region of interest by choosing the DOM subtree of the node with the maximum number of text node pairs that belong to the "same" path from the root. One of the assumptions made is that a record consists of one or more consecutive child nodes of some parent. To divide the subtree of interest into records, all subtrees are clustered by their edit-distance similarity strings obtained by traversing each subsequent subtree in depth-first order. The best record division scheme is chosen according to the similarity of the records to one another. Attributes are extracted by using text alignment, aided by the inclusion of identifiers in each record. This method might be useful for many registries, regardless of their use or omission of tables, and header identifiers could aid in attribute recognition. The expected properties of certain fields, such as price (which would always contain a currency sign, numbers, and zero or 1 decimal point), could also be of use. User knowledge about the number and nature of records and headings to such a scheme would aid greatly in verification and elimination of invalid parsing schemes.
Technical Approach

The first phase of the project was able to parse two registries that had simple, non-nested table-based layouts, but it was clear that automated methods could be more powerful, possibly circumventing the need to laboriously inspect each registry’s HTML code, and perhaps even allowing the user to aid in its training. However, the method explored and described in the next section, PAT-trees, was not included in the final product. Instead, a more site-specific parsing was utilized, which was adequate for the table-based registries parsed by the Gift Registry Aggregator.

To parse gift registry tables, the HtmlUnit package ([http://htmlunit.sourceforge.net](http://htmlunit.sourceforge.net)) was used to generate DOM (Document Object Model) trees, allowing the traversal of an HTML “tree” of nested tags, which was easier than operating directly on the (possibly malformed) HTML. HTMLUnit has a complex API that runs a virtual “web browser”, allowing a program to interface with many web sites as though a user was actually clicking on links. Initially, there was some difficulty using the packages due to scripts that HTMLUnit was unable to run. However, the content of interest on the page was still readable when scripts were disabled.

For each gift registry’s parsing scheme, with the aid of the DOM Inspector add-on for Mozilla Firefox, the HTML source of an example page was analyzed. A good example page would contain multiple types of item entries, each one with or without images, links, or even complex price information (such as original and sale prices). By inspecting multiple gift registries on the same site, one would hope to encounter many general types of product entries.

The registries incorporated into the Aggregator display all their data in tables. Because these gift registry websites had a uniform, program-generated table layout, each gift item generally had the same number of cells per row. Due to the use of headings, each field was consistently in the same column. HTMLUnit provided a way of extracting every row in every table of the page, and each row was considered a candidate for data inclusion. By specifying which column numbers contain the data, the parser can filter rows that have missing data. For example, for a row of interest, if in the “Quantity” column no number can be found, the row can be omitted from the dataset. If other fields are optional, such as images, this fact was also included in the parsing scheme for such a registry.

Other fields, such as links and images, contain information included in tag attributes and not between tags. HTMLUnit has an API for extracting such information.

The final, general algorithm that was developed to parse registries the six retailers takes in only four structure-related parameters:

- The number of columns in a data-containing row
- The mapping of columns to data fields
- Any extra columns whose data should be included in an “extra info” field in the database for the purpose of identification (useful for colors, UPCs, etc.)
- Whether or not an actual item record can contain a blank price
The mapping of columns to data fields includes simple mappings such as a “price column”, but some retailers have more complex mappings, such as a “linked image column” (denoting the fact that the image is embedded in a link, requiring an extra traversal to reach it) and “linked image and name table column” (denoting that this column contains a table with two cells, cell 1 containing a linked image, and cell 2 containing the name of the product). In addition, while most of the retailers had a “still needed” column to express how many of an item the registrant still needs, other retailers express quantities such as “requested” and “purchased” (and thus the quantity displayed in the Aggregator is requested minus purchased).

Price, if the retailer consistently included it in all records, was a useful discriminator of data rows from non-data rows, as the search for the dollar sign is relatively simple. This was especially useful where retailers interspersed headers with the data; in such a case, the word “Price” obviously does not contain a dollar sign. Quantity fields were also required to contain number. For some retailers, the price cell had a more complex structure and required some quick parsing to find the lowest price among the included text.

See Figure 1 for a data-flow diagram of the Gift Registry Aggregator. By using the table-based parsing scheme with retailer-dependent parameters, items are imported into the database on each refresh of the aggregated gift registry. Item name, url, price, quantity, image source, and extra columns included by the parsing scheme as “extra info” are imported directly from the page. After all items have been imported, the gift recipient can then update comments and category information about each one. Imported items are compared to existing records in the item table; if an old item matches a newly imported item in name and “extra info”, its quantity, price, and timestamp are simply updated, so that any registrant-added data (category and comments) can persist between refreshes in the table.

![Figure 1: Overall Design](image-url)
Figure 2 provides an overview of the structure of the Gift Registry Aggregator website. Potential gift-givers visit the site to view items from all the gift registries a registrant has included. To find the registry of interest, the visitor first searches for the registrant (See Figure 3). All items from a third-party registry have links that enable gift-givers to buy the item through the official retailer’s website. Gift givers can sort by category, item name, store, and price, without triggering a refresh from source registries. This enables the viewer to quickly know where they can find the right gift for appropriate price, without needing to cross-reference—and search for—the registrant’s gift registries at multiple third-party websites. Figures 3 and 4 on the next page display non-member views of the site.

The Aggregator website allows registered users to create accounts, add registries, and edit category and comment information about imported items. Registrants can also add items that are not associated with any third-party registry. This even allows users to link to a third-party registry that is not supported by the Aggregator, by supplying an item name, link, and comment that the item is available at some other registry. The registrant can delete custom items as gifts are received. See Figure 5 and 6 on the next page for member views of the Gift Registry Aggregator.

For website development, JSPs and Servlets were used for ease in integrating the dynamic, webpage parsing features (written in Java and using the HTMLUnit package) with the web site and ensuring that pages are not loaded until the parsing is finished executing. The web service runs on Apache Tomcat, a web container that can also be used, as it was for this project, as a web server.

Figure 2: Website Design
Figure 3: A potential gift-giver uses a search-box to find the registry of the registrant. The search is lenient, using substring matching.

Welcome to your Aggregated Registry!

View your aggregated registry

Edit your aggregated registry

Add a custom item to your registry

Name: 
URL: 
Image Source: from a website: 
Price: 
Quantity: 
Color: 
UPC: 
Category: 
Comments: 
Submit

Add a registry

- Amazon.com
- Bed, Bath & Beyond
- Bloomingdale’s
- JoOne
- Macy’s
- Tiffany & Co.

URL: 
Comments: 
Submit

Registry Summary

<table>
<thead>
<tr>
<th>Registry Store</th>
<th>Date Added</th>
<th>Items</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Registry</td>
<td>2009-04-21</td>
<td>2</td>
<td>These are items your guests can buy anywhere.</td>
</tr>
</tbody>
</table>

Amazon.com 2008-04-21 13

Legal notice: privacy@123.com

Log Out

Figure 4: Registry view for potential gift-givers. Note that items from various stores are interspersed in the “Sort by Price” view.

Figure 5: Member Home. In a single page, registrants can view a summary of their registries, include new registries and add custom items.

Figure 6: Registry view for registrants. From this view, gift-recipients can change categories of items, add comments, and delete custom items.
Alternative (Unsuccessful) Approach to Data Extraction

Since many websites do not have their data displayed in neat tables, preferring numerous nested tables or divs (perhaps to prevent automated data mining), alternative methods of parsing the webpages were explored. IEPAD and Pat-Trees for maximal repeated pattern detection and selection (Chang et al, 2003), which derive maximal repeats from a webpage, seemed to be a good option, as the system could incorporate user input through a simple selection means, where the user simply chooses the pattern or patterns that provides the best data. This seemed less tedious for the user than the manual data labeling explored in other research papers.

Before Pat-trees could be implemented, an HTML-parser was first constructed. Each “token” consisted either of an HTML tag and its attributes or the text that lay between two HTML tags. Each of these tags was then converted into a page-specific binary encoding, the length (in bits) of each tag’s encoding depending on the number of different tags included. One additional binary word is then used to represent text content.

Once the page has been converted to a binary encoding, the encoded page is stored in a PAT-tree, a type of binary suffix tree. As with any suffix tree, each suffix of the encoded string is inserted into the tree and stored in the leaves; however, only strings that are divisible in length by the tag-code length are actually inserted. This ensures that all maximal repeats represent actual tag patterns, and not bit-shifted patterns that would be translated a “misssense” tag pattern of no significance. As a binary suffix tree, each internal node in the tree represents the first bit at which the patterns in the subtree differ from one another, with the left branch representing a “0” and the left branch representing a “1”. PAT-trees reduce storage size by allowing “skips”, so that branches with no suffixes are eliminated. For example, if only suffixes “001” and “0001” were stored in a PAT-tree, since they only differ at the third position, the root of the tree would be of index 3, the leaf “0001” would be its left child, and the leaf “001” would be its right child.

Once the PAT-tree is constructed, repeats (occurring at least twice) can be extracted. Each of these repeats is represented by an internal node. The “score” of these repeats is the count of the leaves in that subtree. The pattern of the repeat is built determined by concatenating the 0-1 “labels” of the branches it took to reach that internal node from the root. These binary patterns are then converted back into their tag form. In order to enable selection of useful patterns, the program also prints out the actual, data-containing “hits” of the tag patterns in the page.

After implementing PAT-trees and viewing the results, a few issues precluded the method’s inclusion into the final product. First, the algorithm produced thousands of candidate tag/text patterns of length. The original papers suggest using statistics to sift through these, such an “even-spacing” quantifier. Such metrics work on too large of a scale, and are not really appropriate for gift registries with multiple headings, varying records, and interpolating text. Second, records within registries varied to such a degree that numerous tag patterns—and not even high-scoring, multiple-hit patterns—
actually described real records. The inclusion or exclusion of an image, for example, was fatal to finding a general pattern. Generalizing the parser (i.e. omitting font tags) did not improve the performance. Third, due to the high level of variance between records’ tag patterns, the algorithm had difficulty finding the beginning and ends of records. Once it was determined that any useable pattern selection would function best if it exploited table structure and the validation of fields such as price and item links, the development strategy returned to table-parsing.

Conclusion

Through a web-interface running on Apache Tomcat on a private machine, the Gift Registry Aggregator allows registrants to sign up for an account and add their registries from six retailers: Bed, Bath & Beyond; Macy’s; Bloomingdale’s; Amazon.com; Tiffany & Co.; and JCPenny. In addition, a framework is in place to allow the addition of parsing schemes for other table-based gift registry websites. Gift recipients can also manually add descriptions of items that are not included on any supported online registry. After searching for registered users, identified by first and last name and event date, gift givers can view the aggregated gift registry and sort by price, item name, category, and store name. Purchases off the third-party websites are reflected the next time a user views the website.

Restricting the domain of web parsing, as was explored in this project by using gift registries, can make the task of data extraction easier; however, even in a limited domain, it is very difficult to build a truly general parser to handle unseen page types. While it was certainly interesting to explore PAT-trees and their use in extracting records from a web page, it was difficult to sort through the results of the algorithm without using assumptions about data content and table structure. Thus, it would have been even more difficult to allow the user to participate in the parser training process. The process of incrementally analyzing and adding source registries was determined to be faster and easier to develop, yet it is a less extensible method of developing the Aggregator, as it restricts the product as-is to parsing only the supported third-party sites. However, since the table-based parsing process was parameterized, other table-based registries, after careful analysis of their structure, could be incorporated into the system while still leveraging the existing parsing strategies.
Annotated Bibliography

Gift Registries


This tool allows potential gift givers to search registry databases at multiple retailers to discover where the party has a gift registry. While the tool has a limited number of supported retailers, this tool could still be useful in my product to offer to users who aren’t sure where they are registered. At the least, my site could include this as a link.


This website allows a gift recipient to create a list of gifts you would like to receive by manually linking to the web site where the gift is found. The interface requires manual work to compile the gifts and provides no input for pictures.


Provides a “Universal Gift Registry Service” by providing users with an installable toolbar that find and imports into the registry the dominant product on a page. The Gift Registry Aggregator reverses the directionality of the approach of this website so that registries can still be independently maintained at the third party website.


The Wedding Channel allows celebrities to notate which of their partner retailers have gift registries; however, the website does not actually bring those lists together into one combined list.


Things I Love is a universal gift registry that allows users to write descriptions of gifts that they wish to receive. The site’s maintainers express a wish to be able to import third-party gift registries, but there is no mention of how it would be maintained.
**Data Extraction Resources**


This relatively recent paper was written by assistant professors from the Department of Information and Communications Technologies, at the University of A Coruña, Spain. The method described for extracting data records from HTML pages operates on a single page containing a list of data records. After locating the data region containing the dominant collection of data items, it clusters the possible division scheme and chooses the one that is most consistent with itself according to edit-distance. Multiple string alignment is then used to separate the records’ fields. The authors validate their method using a large number of sites and claim that they obtained good results.


In a revised version of their earlier IEPAD system, Chang et al. describe a supervised method data extraction method which uses pattern discovery to perform supervised extraction by allowing the user to choose patterns rather than examples. The primary author is an Assistant Professor at the Department of Computer Science and Information Engineering, National Central University in Taiwan. Later papers claim that this method does not produce good results. In the Registry Aggregator project, added information still had difficulty sifting through the number and quality of candidate patterns returned.


The book in which this article is included contains various perspectives on the field of Information Retrieval, bringing together the basic data structures and algorithms. The editors describe their audience as including software engineers as well as computer science students who are interested in text retrieval. This chapter provides an overview of the PAT trees, in existence for over 30 years, which are critical for the IEPAD parsing scheme implemented over the course of the Gift Registry Aggregator project.


These MIT affiliates (Karger is a professor of Computer Science) describe a system that allowed non-technical users to create and use wrappers on websites for supervised, automated data extraction. Thresher provides a parsing scheme based on tree-edit distance, in addition to providing a good yet complex example of user interaction with wrapper generators.

This site provides the download page, API, and tutorials for a so-called “browser for Java programs”. The Gift Registry Aggregator uses the packages to parse the DOM-trees for websites, but HTMLUnit also has functionality to support entering data into forms, clicking on links, and executing JavaScript.


Though not a very recent paper, the paper provides detailed algorithms for a method of “wrapper induction” for learning rules about extracting data. This is a supervised method that requires numerous training examples. The second author, who has published numerous papers on the area of information retrieval, is a professor in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong.


The work, accomplished by a Computer Science PhD candidate and professor at the University of Illinois at Chicago, describes a supervised method of data extraction using alignment of trees, applicable to HTML tag trees. The method accounts for the inclusion of exclusion of optimal fields, allowing what fields that remain to be extracted with very high accuracy. The work was supported by the NSF.