Test Collection Management and Labeling System

Eunyee Koh
Advanced Technology Labs, Adobe Systems Inc.,
San Jose, CA, 95110-2704, USA
eunyee@adobe.com

Andruid Kerne, Sarah Berry
Interface Ecology Lab,
Dept. of Computer Science & Engineering
Texas A&M University, College Station, TX 77843, USA
andruid@cse.tamu.edu, berry.sarah@gmail.com

ABSTRACT
In order to evaluate the performance of information retrieval and extraction algorithms, we need test collections. A test collection consists of a set of documents, a clearly formed problem that an algorithm is supposed to provide solutions to, and the answers that the algorithm should produce when executed on the documents. Defining the association between elements in the test collection and answers is known as labeling. For mainstream information retrieval problems, there are publicly available test collections which have been maintained for years. However, the scope of these problems, and thus the associated test collections, is limited. In other cases, researchers need to build, label, and manage their own test collections, which can be a tedious and error-prone task. We built test collections of HTML documents, for problems in which the answer that the algorithm supplies is a sub-tree of the DOM (Document Object Model). To lighten the burden of this task, we developed a test collection management and labeling system (TCMLS), to facilitate usability in the process of building test collections, applying them to validate algorithms, and potentially sharing them across the research community.

Categories and Subject Descriptors
H.3.7. [**Digital Libraries**]: Collection.

General Terms
Algorithms, Experimentation.

Keywords
Test collection, XML schema, Document Object Model.

1. INTRODUCTION
Many information retrieval or information extraction researchers use test collections to validate their algorithm's precision and recall in reference to a test collection [1]. The *test collection* consists of a set of documents, for which an algorithm is supposed to provide solutions. A test collection is *labeled*, that is, annotated with the answers that the algorithm should produce when executed on the documents. Test collections are an important factor in research validation, so they need to be built objectively and maintained consistently. There are publicly available test collections, developed by institutions such as TREC [11]. However, the documents in those collections are not labeled in a manner appropriate for all information retrieval and extraction research problems. Thus, researchers have needed to build and label their own test collections, for example, the Open Video test collection [9].

A systematic mechanism for building test collections eliminates errors and enforces consistency in labeling practices. In addition, a system that manages test collections facilitates usability in the process of building test collections, applying them to validate algorithms, and potentially sharing them across the research community. To lighten researchers' burden of building their own test collections, we developed the Test Collection Management and Labeling System (TCMLS).

The system is designed with the client-server model. The client is implemented as a Firefox browser extension, which enables researchers to collect and label any HTML documents using their browser. The extension sends a service request message, such as what document needs to be labeled. The server, which is built using lightweight semantic distributed computing services [12], performs the requested service, such as uploading a copy of the document, and sends a response, including result status. We specified the request and response message format between client and server using XML. If the message does not follow the specified syntax and semantics, the server will ignore it.

This paper starts by examining related work addressing and explaining the need for managing test collections. Then, we present the design of the TCMLS, in the context of our research problem, which involves extracting informative parts from web documents. We describe how the system works, and how we are using it. We close by discussing how this work can meet the research community's needs.

2. RELATED WORK
Various test collections have been used throughout the years for the evaluation of information retrieval systems. TREC provides a set of large reference test collections that are extensively used by researchers [11]. TREC collections have included Web Test Collections, the Blog Track, the Query Track, the Question Answering Track, and the SPAM Track. It also supported a video track devoted to research in automatic segmentation, indexing, and content-based retrieval of digital video, which then emerged as independent [13]. Beyond TREC, other test collections include CACM, ISI collections, and Cystic Fibrosis [5].

This would seem to be a large set of test collections. However, in fact, the utility of these collections for research is to evaluate an important but small set of possible information retrieval and extraction problems. To address this limitation, many researchers

DocEng'09, September 16–18, 2009, Munich, Germany.

build their own test collections to enable conducting performance evaluation. For example, Dakka *et al.* could not rely on the above popular research collections because the collections do not include the variety of alternative news sources in news portals, which is critical in their research [3]. Instead, they collected news articles crawled and processed by Newsblaster [8] and conducted user studies to collect users' relevance judgment for their experiments. Liu *et al.* collected PDF documents from various sources, which they used to evaluate the quality of their table detection algorithm [7]. Song *et al.*, investigating an information extraction problem similar to ours, collected 600 web documents from 405 sites in 3 categories in Yahoo: news, science and shopping [10]. Both Song *et al.* and our research problems involve using the Document Object Model (DOM) tree representation of an HTML document [14] to identify document components. They created their own tool for labeling importance of blocks in web documents. Five human assessors manually labeled blocks in the documents with importance values. They then used this labeled test collection to assess the precision of their extensions to the VIPS algorithm for automatically assessing block importance [2]. A disadvantage of this tool is that instead of being able to label any DOM nodes, the blocks that can be labeled are only those identified by VIPS. Thus, the tool has limited extensibility for building test collections for validating solutions to different research problems. By contrast, our system can label any informative blocks in documents, as well as specific metadata, such as images and image captions.

3. SYSTEM DESIGN

Building a test collection is conducted through two stages. The first stage is defining how documents will be labeled and categorized. The next is interactively collecting and labeling documents so that we can utilize the test collection in evaluating algorithm performance. This paper focuses these two stages, which the TCMLS is being developed to support. We designed the system in a client-server model in order to manage a central test collection repository.

This section presents the semantics with which our test collection documents can be labeled. Once such semantics are defined, the test collection can be formed. TCMLS usage begins with the user

Figure 1. Example of nodes highlighted with Modified DOM Inspector and labels assigned to the test collection document.

Table 1. Document labeling semantics for the test collection to validate informative images and text extraction algorithm.

`category`	The category the test document belongs to.
`partition`	For partitioning a document to identify semantic sub-trees of informative context. Partitions are not nested or overlapping in the DOM tree.
`inform_img`	Label informative image with an appropriate caption and the best field true for the one best representing the content.
`inform_text`	Label informative text.
`noninform_text`	Label non-informative text only within a block of informative text to restrict scope of the labeling.
`caption`	Mark as a caption text that describes an informative image.

identifying each document to collect, in response to which the TCMLS stores a copy of the document and its media assets in its repository. Next, the user applies the semantics to each test collection document using the TCMLS, and the system stores the labels with the associated document in the repository.

3.1 Document Labeling Semantics

We defined labeling semantics for the research problem of extracting informative images and text from a document. Table 1 describes these labeling semantics. The labels are annotated to the appropriate DOM nodes (elements) in an HTML document. We also enable labeling the document as a whole as belonging to one or another `category`. So far, in our case, the category labels for documents are either "news article" or "news index". The labeling semantics are coded in the TCMLS.

3.2 Interactive Collecting and Labeling Client

The TCMLS client enables researchers to collect any HTML document using their browser. It is implemented using the DOM Inspector (DI) [4], an open source Firefox extension. DI enables the user to examine the hierarchical DOM tree of the HTML source code of a web document [14]. The DI already contains a built-in feature that allows the user to add, edit, or remove attributes of document elements. When you click an HTML element in the DI, the corresponding document block is highlighted with a red rectangle box in the browser (see Figure 1). Details about the HTML element, in the form of attribute-value pairs, are displayed. We extended this software by creating the aptly named *Modified DOM Inspector* (MDI), which enables the user to easily label appropriate nodes in test collection documents.

3.3 Identify Each Document to Collect

Figure 2 shows the buttons that we added to the MDI to enable the user to apply label semantics to a DOM element. The researcher uses Firefox to browse. She selects a document to collect, and assigns a `category` from the set offered by the Modified DOM Inspector. She clicks the URL to Server button (see Figure 2). Then, the MDI extension forms the `collect_document` message, in XML, which requests the system server to store a document in the test collection repository, as follows:

```
<collect_document category="" url="" datetime=""/>
```

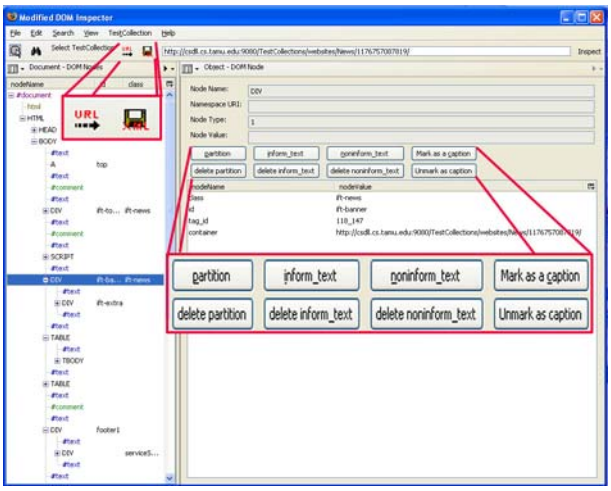

Figure 2. Modified DOM Inspector: URL to Server button stores document in repository. Semantic buttons (right) label selected HTML element. Save XML button stores labeling in repository.

In the `collect_document` message, the `category` field specifies the category selected with the interface. The `url` field is for the document URL that the researcher is collecting, and the `datetime` is the time that the message is sent to the server. The `datetime` is utilized in the server to track the document at the time as Web continues to get updated overtime. This step, and the subsequent performance of the service to store the document in the test collection repository, must be performed prior to interactive labeling.

3.4 Store the Document in Repository

When the server receives the `collect_document` message, it connects to the specified URL, receives the document, and stores it and referenced resources such as images and JavaScript in the repository. As the documents themselves and the associated resources can be changed or removed, we stored copies. This requires resolving all URLs for referenced resources into relative paths stored in the repository. Hyperlinked documents were not stored and links to them were not transformed. The TCMLS stores and fixes resource references in order to preserve complete visual copies of each document.

Some HTML documents do not follow the specification completely. For example, sometimes documents have some missing ending tags. Thus, a typical XML parser is unable to form the DOM tree from them. To address this issue, we use JTidy [6], a syntax checker and pretty printer. JTidy cleans up malformed and faulty HTML, so that

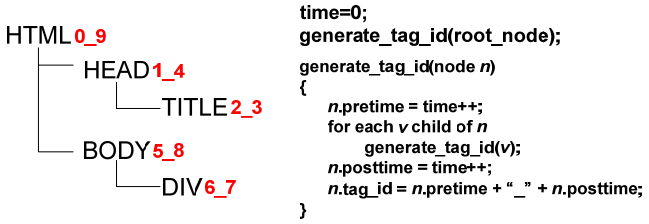

Figure 3. Generate tag_id of each HTML tag by traversing the DOM tree with DFS algorithm. The left diagram shows the DOM tree, with each tag_id generated by algorithm (right).

the TCMLS can build DOM trees from any document in the test collection repository.

Upon forming the DOM for an HTML document, the system also generates a unique identification number, *tag_id*, for each HTML element in test documents. This enables the cross-reference between the label and the test document in separate files. The generated tag_id was added as an attribute in each element in test documents in the repository. The following is the example from a test document.

```
<html tag_id="0_1144" lang="en">
<head tag_id="1_39">
```

We used the depth-first-search (DFS) algorithm to generate tag_id. The DFS algorithm records the discovery time and the finishing time as each element in the DOM tree is traversed. We defined the tag_id by combining the discovery time and finishing time. The tag_id is unique in the DOM tree, and also provides parent-child relationship among the elements in the DOM by seeing the number range in the tag_id.

For example, in Figure 3 left, document elements have been labeled with tag_id. The tag_id of the HTML is 0_9, the HEAD is 1_4, and the BODY is 5_8. The starting and ending range of the tag_id shows that the HTML element is the parent of both the TITLE and the HEAD, and that the TITLE is in the different tree from the BODY. Knowing the parent-child relationship helps researchers to label the test documents without having redundant labels inside the same sub-tree. It also helps to locate the labeled tags in the test documents. This is a more efficient way to label DOM node relationships than XPath [15], which has functionalities to find relative nodes, because XPath incurs tree traversal iterations to operate, while our tag_id directly represents parent/child relationships.

After the system server processes the request to store the document in the repository, it sends the client a response. The response is either `ok_response` (the request has been successfully finished) or `error_response` (the request failed to be performed by the server). When the client receives the `ok_response`, the browser redirects to the test document URL stored in the repository, so that researchers can label it.

3.5 Label Each Document

The MDI is used to label each document, with the semantics described in Table 1, by clicking labeling buttons (see Figure 2). When an `img` element is selected, another view appears. This view enables labeling each image as informative or not, and, in the former case, for one or more captions to be associated. A dropdown checklist will contain all captions that have been labeled through the interface in Figure 2. The user can check and uncheck captions to associate them with the correct image.

3.6 Store Labels in Repository

When the user has finished labeling a document, she clicks the Save XML button (see Figure 2). Then, a function recursively walks through the HTML DOM tree with the DFS algorithm, and checks for the annotation of the labels. Each label is represented with its tag_id to form compact XML that represents the labeling. Post-processing is performed to clean up the XML, such that `partition` labels are ordered from least to greatest, and each

inform_image is associated with the correct caption. The completed XML labeling string is sent to the server to store in the repository. Here is an example:

```
<document
url="http://csdll.cs.tamu.edu:9080/TestCollections
/websites/News/1176757087819/" title="BBC NEWS |
UK | England | Berkshire | Friendly fire pilot
back in Iraq">
   <partition_set>
      <partition id="0" tag_id="362_700">
        <noninform_text_set>
          <noninform_text tag_id="428_433"/>
          <noninform_text tag_id="434_449"/>
        </noninform_text_set>
        <inform_text_set>
          <inform_text tag_id="366_367"/>
          <inform_text tag_id="372_451"/>
        </inform_text_set>
        <inform_img_set>
           <inform_img tag_id="379"
           url="newsimg.bbc.co.uk/media/images/42
           687000/jpg/_42687225_matty_pa203b.jpg"
           best="true">
             <caption_set>
               <caption tag_id="380_381"
               value="L/Cpl Matty Hull died four
               years ago in the attack in Basra"/>
             </caption_set>
           </inform_img>
        </inform_img_set>
      </partition>
  </partition_set>
</document>
```

The label_document message encapsulates the XML labeling string, to send it to the server for storage in the repository:

```
<label_document>
     XML labeling string
</label_document>
```

3.7 Browsing Test Collection

All the built test collections can be browsed from http://ecologylab.net/testcollections/. The directory structure is based on the selected category of test documents. If a user clicks a category, all the collected documents under the category are listed. Users can easily browse and download the test documents with the label XML files.

4. Conclusion

We have developed a system to reduce researchers' tedious task of test collection management and labeling. The system provides usability for iteratively building test collections. It facilitates algorithm validation. As they are developed, test collections are published on the web, enabling sharing by the research community. By installing the browser extension on Firefox, other researchers can also contribute to the test collection. They can browse and download the built collection, and use it for the algorithm validation. Our goal is to maintain our system to enable sharing and extending collections among the research community, to support algorithm development efforts.

While institutionalized test collections have been developed to promote solutions to important research problems, there is a world of important research problems they have not addressed. However, test collections are necessary for much research on information retrieval and extraction. The burden of creating test collections may function as a barrier to entry for important new research areas. The present research develops tools to support test collection management and labeling. It thus has the potential to facilitate the diversification of research efforts in the fields of information retrieval and extraction, by reducing the efforts necessary to address research problems whose significance has not yet been institutionally acknowledged, but which may turn out to be of great importance.

5. ACKNOWLEDGMENTS

Support is provided by NSF grants IIS-0633906 and IIS-0747428. We would like to thank Laurie Byrum, Adobe Systems for useful comments to improve this paper and the anonymous reviewers for helpful comments.

6. REFERENCES

[1] Baeza-Yates, R., Ribeiro-Neto, B., Modern Information Retrieval, Addison-Wesley Longman Publishing, 1999.

[2] Cai, D., Yu, S., Wen, J. R. & Ma, W. Y., VIPS: a Vision-based Page Segmentation Algorithm, Microsoft Technical Report, MSR-TR-2003-79, 2003.

[3] Dakka, W., Gravano, L., Efficient summarization-aware search for online news articles, JCDL 2007, 63-72.

[4] DOM Inspector, Mozilla, http://www.mozilla.org/projects/inspector/, last visited 12/13/2007.

[5] Fox, E. A., Characterization of two new experimental collections in computer and information science containing textual and bibliographical concepts, Technical Report 83-561, Cornell University, Department of Computer Science, Ithaca, NY, 1983.

[6] JTidy, http://jtidy.sourceforge.net/, last visited 01/08/2008.

[7] Liu, Y., Bai, K., Mitra, P., Giles, L. C., TableSeer: automatic table metadata extraction and searching in digital libraries, JCDL 2007, 91-100.

[8] Newsblaster, http://www.newsblaster.com/, last visited 04/22/2009.

[9] Slaughter, L., Marchionini, G., Geisler, G., Open video: A framework for a test collection, Journal of Network and Computer Applications, 23(3), 2000, pp. 219-245.

[10] Song, R., Liu, H., Wen, J. R., & Ma, W. Y., Learning Important Models for Web Page Blocks based on Layout and Content Analysis, Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations, 6(2), pp. 14-23, 2004.

[11] Text REtrieval Conference (TREC), http://trec.nist.gov/, last visited 01/18/2008.

[12] Kerne, A., Toups, Z.O., Dworaczyk, B., Khandelwal, M., A concise XML binding framework facilitates practical object-oriented document engineering, ACM DocEng 2008, 62-65.

[13] TREC Video Retrieval Evaluation (TRECVID), http://www-nlpir.nist.gov/projects/trecvid/, last visited 04/22/2009.

[14] W3C, Document Object Model (DOM) Level 2 Core Specification, http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/, 2000.

[15] XML Path Language (XPath), http://www.w3.org/TR/xpath, last visited 04/22/2009.