2016 IEEE 17th International Conference on Information Reuse and Integration

Clustering Web Pages Based on Structure and Style Similarity

Thamme Gowda[1] and Chris Mattmann[1,2]

[1]University of Southern California, Los Angeles, CA
Email: {thammegowda.n, mattmann}@usc.edu

[2]NASA Jet Propulsion Laboratory, Pasadena, CA
Email: chris.a.mattmann@jpl.nasa.gov

Abstract

We consider cluster analysis task on web pages based on various techniques to group the pages. While grouping the web pages based on the semantic meaning expressed in the content is required for some applications, we focus on clustering based on the web page structure and style for applications like categorization, cleaning, schema detection and automatic extractions.

This paper describes some of the applications of similarity measures and a clustering technique to group the web pages into clusters. The structural similarity of HTML pages is measured by using Tree Edit Distance measure on DOM trees. The stylistic similarity is measured by using Jaccard similarity on CSS class names. An aggregated similarity measure is computed by combining structural and stylistic measures. A clustering method is then applied to this aggregated similarity measure to group the documents.

1 Introduction

The World Wide Web (WWW) has become the go to place for all kinds of information. The web has evolved from merely connected html documents to rich applications with asynchronous interaction techniques (e.g., AJAX) and pleasantly-themed styles. The broad variety of information on the WWW has facilitated the creation of applications that rely on this data. These applications require the content to be fetched, parsed, cleaned, extracted, before any decisions can be made. One of the challenges faced by current data scientists is the extraction of required content on the WWW through the identification of relevant portions of web pages and content, and through the elimination of unnecessary, less-relevant WWW web pages and content. This process, known as deduplication, relies heavily on being able to model both the structure of web pages and documents; as well as using those models to compare and decide unique documents to include in the corpus, and those too similar to others that can be removed. Several researchers have studied this topic [13, 12, 16].

The WWW is described as inter linkage of heterogeneous web pages and nowadays it includes all sorts of content (images, videos, PDF documents, etc.). It is evident that there are many different ways to produce the content (e.g., web editors; document authoring applications; cameras; and video systems) and many different ways to consume the content (browsers, mobile devices, game consoles, etc.)

Thanks to the efforts of World Wide Web Consortium (W3C), all the producers and consumers of *web pages* comply to Document Object Model (DOM) specification [14]. These web pages are represented in one or more versions of the Hyper Text Markup Language (HTML). HTML is meant to offer clues to the rendering engines to structure the content. In this paper we describe how the same clues can be used to measure the similarity between web documents. We envision these techniques being extensible to other document types through the use of Apache Tika [9]. Tika can automatically identify a document's type; parse the document, and generate an intermediate XHTML representation/DOM of metadata of *any* document - even videos, and images. A detailed explanation of that process is beyond this paper and we refer the reader to Chapters 4 and Chapters 5 of [9].

Graph and Tree data structures are some of the popular data structures known to Computer Scientists. The Document Object Model (DOM) specifies web document as a

978-1-5090-3207-5/16 $31.00 © 2016 IEEE
DOI 10.1109/IRI.2016.30

labeled ordered *tree* of DOM elements and as such we can apply existing tree based algorithms to them. One of the interesting algorithms for this context is Tree Edit Distance. We have used Zhang-Shasha's Tree Edit distance algorithm to compute the structural similarity between DOM trees.

It is not surprising to claim that modern web pages have much more than the content wrapped in HTML tags. Cascading Style Sheets (CSS) serve as a fundamental building block of web pages. Web developers nowadays put the information not only in the HTML tag hierarchy but also in the style sheets. So we do need to consider the styles to determine the similarity between the web pages. In this paper we describe the Jaccard Similarity, a simple measurement of resemblance and containment between sets, however other measures can also be used for the same purpose. In this case we treat the CSS style sheets as a set of features within a set corresponding to each web page and use that as the set comparison.

Client side scripting like *JavaScript* is another component of the web pages determining the overall similarity, however determining the similarity of scripts is beyond the scope of this paper.

This article is organized as follows: Section 2 briefs the scope of clustering the web; Section 3 describes the application of Tree Edit Distance (TED) measure and then derives the structural similarity from the distance measure. Section 4 presents a method to compute stylistic similarity. Section 5 presents a way to combine structural similarity with stylistic similarity and Section 6 presents a method to cluster the web pages based on our similarity measure. Section 8 rounds out the paper.

2 Application of the web document clusters

As of 2016, most modern websites are dynamic in nature. A common practice to achieve the dynamism is by binding the data models with view templates. The data models are usually retrieved from structured data stores and the view is generated by combining it with the templates. Let d_1, d_2,... d_n be the data models secured in data stores on the web servers. These could be details of products on e-commerce sites or user profiles on social networks. Web servers and application frameworks binds these data models, d_1, d_2,... d_n, to a programmed view template, T_i, and generating the HTML views D_1, D_2,... D_n. In a real crawl of the WWW, the views from various templates will be fetched and mixed with each other. One of the applications of this similarity based web page clustering is to group all the documents which are generated from the same template. This operation is a pre-processing step for an automatic extraction of data models that works by reverse engineering the template bind operation.

In addition to the above, we have found the following applications:

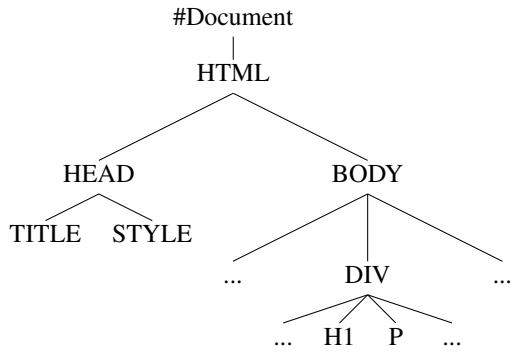

Figure 1. Web page viewed as a DOM tree

1. Extraction of structured content based on structural alignment - precise clusters results better results with automated extractions. Also the query based extraction mechanisms like XPATH, guarantees the yield on all documents in clusters if it works for single document in the cluster.

2. Cleaning - Since the documents which require attention appear in different clusters than those documents that do not require attention, the removal of noise can be done at cluster level instead of document level.

3. Analysis based on the category - Clusters represents categories of web pages. Each category can be treated separately for analysis.

4. Crawl coverage test - Clustering the output of crawler and counting the documents in clusters facilitates crawl coverage testing. This also helps to more precisely identify the category of web pages that are available at the source but missing in the crawler output.

3 Structural Similarity using Tree Edit Distance Measure

As mentioned in Section 1 and shown in Figure 1, web pages are structured as labeled ordered trees in which the textual and multimedia content is wrapped inside the *HTML* tags. These tags specifies the clues to the rendering engines to display the content. Thus by its very purpose, web pages with almost same structures most likely share the same DOM tree structure.

We have reviewed Tree Edit Distance (TED) algorithms for determining the similarities [3, 17]. We applied Zhang - Shasha's algorithm for labeled rooted trees because of its simplicity and completeness [17].

Highlights of the Zhang-Shasha's algorithm include those enumerated below.

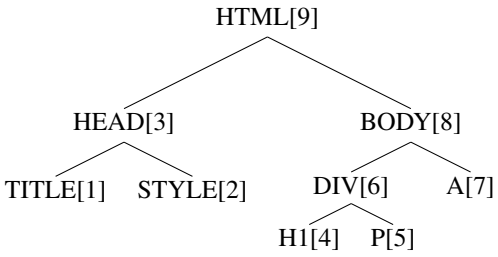

Figure 2. A sample DOM tree with post order numbering for DOM elements

- The elements in tree are indexed in post order as shown in Figure 2, the nodes in the DOM tree are similarly indexed in post order.
- The tree is incrementally built from smaller forests and the edit cost between two forests is computed by gradually aligning nodes with Insert, Remove and Replace operations as described in [17].
- Dynamic programming is used to efficiently compute the edit distance between the root nodes of two DOM trees.

The function `treedistance` (T_1, T_2) is defined as per [17] with custom costs for $\gamma_{insert}, \gamma_{remove}$, and γ_{update} for `insert`, `remove`, and `replace` operations respectively. The function returns a positive `cost` associated for the sequence of edit operations required to transform $T_1 \rightarrow T_2$. Since the Edit Distance is unbounded, the following normalization is performed. Let γ_{max} be the maximum of $\gamma_{insert}, \gamma_{remove}$, and γ_{update} costs. The structural similarity of two DOM trees, T_1 and T_2, is determined by :

$$similarity = 1 - \frac{treedistance(T_1, T_2)}{\gamma_{max}(\mid T_1 \mid + \mid T_2 \mid)}$$

In our experimental evaluation, we noted that the Zhang-Shasha's tree edit distance (TED) algorithm is slower for modern web pages due to its higher time complexity. Zhang Shasha's TED algorithm has the worst case time complexity of $O(|T_1| \times |T_2| \times min(depth(T_1), leaves(T_1)) \times min(depth(T_2), leaves(T2)))$ and the space complexity of $O(|T_1| \times |T_2|)$ i.e., in the order of $O(n^4)$ and $O(n^2)$ respectively [17]. We also evaluated Robust algorithm for Tree Edit Distance (RTED) [10] and All Path Tree Edit Distance (APTED) [11] for an efficient replacement. The RTED has the time complexity of $O(n^3)$ and space complexity of $O(n^2)$. Results showed that RTED is slightly faster than Zhang-Shasha's algorithm for modern web pages but it uses comparatively higher memory due to its higher constant in the space complexity. The newer APTED implementation from the same authors combines best of performance and memory efficiency. In our practical system we used APTED implementation for structural similarity evaluation. This decision for reducing the runtime did not alter accuracy of results as the tree edit distance is same for chosen $\gamma_{insert}, \gamma_{remove}$, and γ_{update} costs irrespective of implementation mechanism.

```
def classes(doc):
  # get all unique values of 'class'
  classes = set(xpath(doc,
      '//*[@class]/@class'))
  result = set()
  for cls in classes:
    # class names separated by space
    # add all the class names to set
    res.add_all(cls.split('\s+'))
  return result
```

Figure 3. Getting set of unique CSS class names

The tree edit distance measure behaves precisely for a vast majority of documents. However the results have shown that the tree edit distance measure alone is not sufficient to accurately determine the similarity of web pages. For instance, web pages with uneven number of repeated groups tends to possess higher tree edit distance even though have same DOM structure at the higher level. Some of the examples of repeated groups are user comments to blog posts and listing of results in search pages. Even though two blog posts are similar in DOM structure, the unequal number of comments causes higher tree edit distance and thus lower structural similarity. However all the repeated groups possess similar styles, so the next section describes our proposed stylistic similarity measure to address these situations.

4 Stylistic Similarity using Jaccard Similarity Measure

Style of the webpages present in Cascading Style Sheets are also critical information to determine the similarity of web pages. The documents generated by using the same template possess same styles. Web developers have the option to specify the styles either inline with the DOM elements as a value to the `style` attribute or via the `class` attribute. The scope of the method described next is limited to the high level analysis of `class` values.

Let D_1 and D_2 be two web pages for the sake of computing the stylistic similarity. The set of style class names are retrieved by using W3C's DOM API and an XPATH expression as shown by the code snippet in Figure 3. Since the set theoretic based similarity measure efficiently

addresses the repetitions in CSS class names across the documents, we chose to use Jaccard's similarity for stylistic measure. The Jaccard similarity coefficient of styles is computed by determining the fraction of styles overlapping in both of them[15]:

$$A = classes(D_1)$$
$$B = classes(D_2)$$
$$style\_similarity = \frac{\mid A \cap B \mid}{\mid A \mid + \mid B \mid - \mid A \cap B \mid}$$

Implications of Jaccard's Similarity coefficient in style class names is comprised of :

- Since the unique class names are used for computing the similarity, the unequal number of repeated groups does not alter the stylistic similarity.
- The documents displaying similar content possess the same set of class names thus they result in a higher value for the Jaccard similarity coefficient.
- The stylistic similarity measure might also cause false positive for the dissimilar documents from same website. This is due to the fact that the styles are most likely kept consistent across all the web pages in the same website. Thus it only complements the proposed structural similarity measure defined in Section 3.

The next section describes a way to combine stylistic and structural similarity measures with an aim to reduce false positives and false negatives.

5 Aggregating the Structural and Stylistic Similarities

The $structural$ and $stylistic$ similarities are computed as per section 3 and 4 respectively. The significance of structural and similarities for grouping the web pages varies greatly depending on the data set as not all websites are created equal. A constant, κ of range [0.0, 1.0] is chosen as a fraction of significance of $structural$ similarity. Thus, the aggregated similarity is obtained by the following:

$$similarity = \kappa \cdot structural\_similarity + (1 - \kappa) \cdot stylistic\_similarity \quad (1)$$

To start with, we can give equal significance to the DOM structure and CSS style features by choosing $\kappa = 0.5$. However, κ is a way to experiment with various levels of significance to find the best value for any given data set.

6 Clustering based on Similarity Measures

The aggregated similarity from Section 5 has been used as the deciding factor for partitioning the web pages into clusters. Since the web pages are arbitrary in DOM structure and CSS styles, it is expected that the clusters are non-globular in shapes. In addition, the number of clusters, i.e. kinds of web pages, in the dataset cannot be easily anticipated. These insights ruled out the possibility of using clustering techniques that requires prior knowledge of number of clusters. The shared nearest neighbors method of Jarvis and Patrick [8] serves as a good fit for clustering the web pages based on pairwise similarity measures. It is an Hierarchical Agglomerative Clustering (HAC) technique capable of producing non globular clusters. This technique also reduces the effect of outliers in dataset. The authors of Shared Near Neighbor Clustering technique suggested an implementation as follows: After computing the similarity matrix of size $n \times n$, the neighborhood table of size $n \times (k + 1)$ is constructed. Further more the neighbors are ordered based on the decreasing order of similarity. Note that the node itself appears at the zeroth position in the neighborhood because of its highest similarity with itself. Two rows of the neighborhood table can be collapsed (thus growing the cluster size and reducing the number of clusters) to one if they share at least k_t number of neighbors. This collapsing operation is repeated until no further collapsing is possible or a maximum of k_{max} iterations are reached. At the end of this step, final clusters of web pages are produced based on the remaining rows in the neighborhood table.

One of the requirement of this clustering system is to be able to cluster larger output from web crawlers. This requirement forced us to scale horizontally on cluster computing systems. We noted that the table datastructure is complex to scale on distributed clusters. Our implementation of shared near neighbor clustering algorithm has following improvements:

- We used graph data structures in which vertices are clusters and edges are similarity measures between them. We built upon the Apache™ Spark's GraphX library [6] to scale the graph size.
- Our work followed the agglomerative (bottom-up) approach and based on this initially each document belonged to its own Vertex. On each successive iteration, vertices which share near neighbors are merged. The information of documents belonging to clusters are stored as properties in vertices. Vertex properties are updated on every merge with its near neighbor.
- Instead of k_t number of neighbors as a threshold to merge clusters, we generalized the threshold by making use of k_p percent of overlap between the clusters.

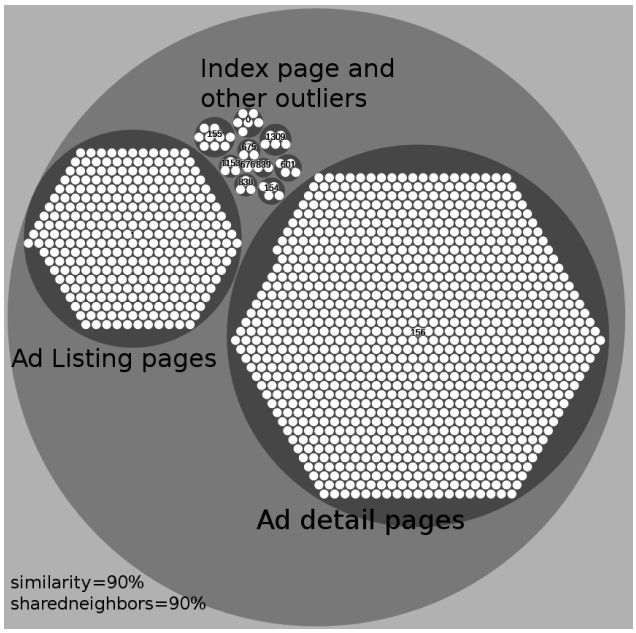

Figure 4. Total 12 clusters when threshold is 90%

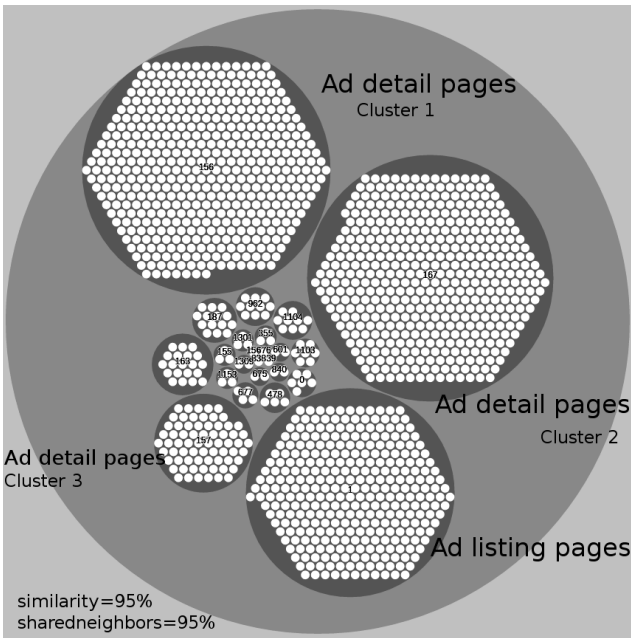

Figure 5. Total 24 clusters when threshold is 95%

7 Evaluation

As part of DARPA MEMEX project[4], we used Apache Nutch[5] web crawler to fetch web pages related to public listing classifieds. Our test dataset included 1310 pages from a popular weapons classifieds site. There were 987 web pages having details of weapon ads, 311 web pages had search and categorical listing details and the remaining 12 of them included index, contact, about, policy and other pages that are typical on an e-commerce site. We followed the methods described above to cluster these pages. The results are then visualized using cluster visualization chart of Data Driven Documents [2, 1].

Initially, we used a threshold of 90% similarity to treat documents as near neighbors and a threshold of 90% shared near neighbors to merge the clusters. With these parameters, we obtained the clusters as shown in Figure 4. After the careful investigation we found that there were 12 clusters in the dataset. A total of 981 out of 987 ad details pages were belonged to same cluster and the missing 6 documents resided in their own clusters. All the 311 documents of categorical and search listing type pages were correctly clustered into the second largest cluster as in Figure 4. Interestingly, the web pages such as contact form, policy and home index pages stayed in the outlier clusters.

In later tests we altered the threshold values. At first, we raised both the similarity and near neighbor thresholds to 95%, and found that there were a total of 24 clusters as shown in Figure 5. The ad details clusters were split into several clusters of smaller size. As expected, the documents in the clusters were more similar than earlier. With few more trials, we found the optimal threshold values for our test dataset, which was approximately 85% similarity to treat them as near neighbors and 85% overlap between shared near neighbors. The final clusters are as shown in the Figure 6. With these thresholds, we found that all the 311 pages of listing type in input dataset belonged to the same cluster. Out of 987 ad detail pages, 982 of them resided in a single cluster and the remaining 5 resided in a different cluster. We found that those 5 misplaced pages missed few html elements that are usually present in other 982 pages. The remaining 3 clusters were made of outliers such as index and contact pages.

8 Conclusion and Future Work

We have described our approach for leveraging the DOM-based structure of web pages to determine similarity between those pages using Tree Edit Distance on the DOM, and using Jaccard Similarity on the stylesheet information. We have shown a shared near neighbor method to combine these information and so far the early evaluation from these results is positive.

Our future work is expanding our evaluation in open

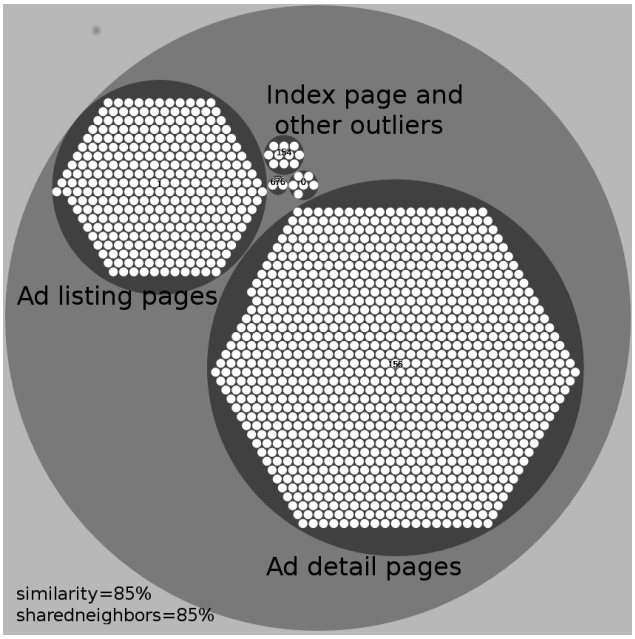

Figure 6. Total 5 clusters when threshold is 85%

datasets provided from the DARPA Memex Deep Search project and efforts for which our team is a contributor to. In addition we plan on applying our similarity metric to any type of document (not just web pages) using Apache Tika [7] as a method first to convert any document into a DOM.

Acknowledgments

This work was supported by the DARPA XDATA/Memex program. In addition, the NSF Polar Cyberinfrastructure award numbers PLR-1348450 and PLR-144562 funded a portion of the work. Effort supported in part by JPL, managed by the California Institute of Technology on behalf of NASA.

References

[1] M. Bostock. D3.js - data-driven documents, 2016. URL: https://d3js.org/; [Online; Accessed: 16-Jun-2016].

[2] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[3] W. Chen. New algorithm for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40(2):135 – 158, 2001.

[4] DARPA. Memex (domain-specific search), 2016. URL: http://www.darpa.mil/program/memex; [Online; Accessed: 16-Jun-2016].

[5] A. S. Foundation. Apache nutch, 2016. URL: http://nutch.apache.org/; [Online; Accessed: 16-Jun-2016].

[6] A. S. Foundation. Apache spark, 2016. URL: http://spark.apache.org/graphx/; [Online; Accessed: 16-Jun-2016].

[7] A. S. Foundation. Apache tika, 2016. URL: https://tika.apache.org/; [Online; Accessed: 16-Jun-2016].

[8] R. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared near neighbors. *Computers, IEEE Transactions on*, C-22(11):1025–1034, Nov 1973.

[9] C. Mattmann and J. Zitting. *Tika in action*. Manning Publications Co., 2011.

[10] M. Pawlik and N. Augsten. Rted: A robust algorithm for the tree edit distance. *Proc. VLDB Endow.*, 5(4):334–345, Dec. 2011.

[11] M. Pawlik and N. Augsten. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173, 2016.

[12] P. Ramirez and C. Mattmann. Ace: improving search engines via automatic concept extraction. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 229–234, Nov 2004.

[13] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 502–511, New York, NY, USA, 2004. ACM.

[14] W3C. Document object model (dom) level 3 core specification, 2004. URL: http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/ [Online; Accessed: 03-Apr-2016].

[15] Wikipedia. Jaccard index, 2016. URL: https://web.archive.org/web/20160403164752/https://en.wikipedia.org/wiki/Jaccard_index; [Online; Accessed: 03-Apr-2016].

[16] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 76–85, New York, NY, USA, 2005. ACM.

[17] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, Dec. 1989.