Master Thesis

Inferring webdocument clusters based on structural similarity

Bob Klaase

March 25, 2018

Primary Supervisor:
prof.dr.dr.h.c.ir. M.J. Plasmeijer

Secondary Supervisor:
prof.em.dr.ir. Th.P. van der Weide

Abstract

The quintessential website consists of a domain, a landing page and additional pages providing further content. In this case links to and from the website can be defined as links to and from any of the pages that exist on the domain, which makes it easy to map and visualize in relation to outside sources. Furthermore search results that point to member-pages of the website can be clustered and presented as a single source of information.
Within the world wide web though, domains exist where multiple websites are housed. Examples of these are the personal pages of researchers on the domain of a research group or university; Blog articles written by individual bloggers on a common, framework providing, domain; or simply a multipurpose domain with a blog, a front page and a forum, without the separation of third level domains.
Being able to cluster such domain sheltered websites offers the same benefits as having a single website domain: incoming and outgoing links can be differentiated from references to another part of the same site.
This paper shows that - at least with the methods conceived of in this research - structural DOM layout is a strong, but not perfect indicator of pages belonging to one another which can subsequently be used to model their interactions with the outside world. Multiple strategies for traversing DOM trees are combined with metrics for determining DOM node similarity to reach this conclusion.

Preface

This document is written to be a master thesis; the last part of my computer science degree. The process from formulating the intent to write such a document, to having it at last in it's completed form has been a long one. Initially I approached my then supervisor, professor Theo van der Weide, with the idea to use all sorts of advanced tooling to infer the social and professional relations of people based on their presence on the internet. This went as far as scanning *.pdf* documents for second author, found on the university research pages. Needless to say, this is not the research described in these pages. In fact, much fear and doubt has been suffered in the process of adjusting the scope and topic to something that can reasonably be investigated within the time frame of a master thesis project.

The final topic sprang on the one hand from the need to narrow the subject down to something more specific, and an actual practical need within the internet solutions company that I was part-owner of at the time of this research. Mindconnect, the company, needed a way to tell if customers were using a service in a way that was explicitly prohibited by the end user agreement. Namely to have more then one different website on what was meant to be an introductory package. In this the setup created for this thesis was actually very successful.

So now the work was done; literature was consulted, strategies were selected and results were compiled. All that was left was to write it all down in a coherent format, give credit were credit was due and talk about possible other applications already hinted upon in the referenced literature. So trivial seemed this task, that I had started a full-time job, in order to write down the details in the evening hours. A big mistake.

Since then I have struggled with all matter of insecurities regarding the relevance and viability of the whole endeavor; kept putting even thinking about the project off for other matters that "had to be" done as well in order to keep the guilt over procrastinating at bay; and even contemplated giving up altogether. Luckily professor Rinus Plasmeijer kindly offered to become my supervisor. Now I want to make it very clear that through no fault whatsoever of professor Theo van der Weide has this process been stalled so long. It was simply my own lack of discipline and the injudicious decision to start a full-time job. Finally I would like to use the remainder of this section to give my thanks to the people that slowly but surely, helped my break the stalemate.

First of all I want to thank Theo van der Weide, for getting me started, and helping me to refine the scope and focus of this project. Second I want to thank Rinus for helping me get back on track and without any hand holding making it clear, that whenever there was some progress he would give me feedback and recommendations on the steps required to have this be a valid master thesis. I want to thank my girlfriend Hilde, who has been a tremendous support and never derided me for taking so ridiculously long to complete. And finally my "friends" whose constant mockary and redicule were in a way also part of the motivation for pulling through.

Contents

1 Introduction

The term website is a loosely defined one. Intuitively a website is a collection of pages with related information, maintained by the same (group of) person(s). Dictionaries offer definitions like: *A set of related web pages located under a single domain name.* (Oxford-Dictionary). Or: *a group of World Wide Web pages usually containing hyperlinks to each other and made available online by an individual, company, educational institution, government, or organization*(Merriam-Webster).

The web and the technology it is build on is constantly evolving. As of this writing the transition from HTML4 and its' derivatives to HTML5 is almost complete. The line between websites and web applications is blurring. However, even with application state and complexity moving to the front-end of web applications being stateless, or able to capture state in a URL still yields very helpful properties. There are best described in Fielding [2000]. That means that methods able to harvest information through URL based requests remain relevant. As long as the requesting application is able to handle JavaScript and/or front-end routing.

In this paper methods of clustering web pages based on their structure are being explored. The hypothesis is, that when a person, or group of individuals create content for the world wide web, the documents between which they divide this content will share enough idiosyncratic (outer) structure, so as to be detectable. When viewing websites, the individual pages share visual similarity (header, footer, menu's) to help guide the attention of the viewer to the content. This is accomplished with recurring pieces of markup language that are shared between those documents. It is this overlap the setups discussed in this paper, attempt to leverage to group pages together.

To do this 7 experiments are conducted on a sample set. Each experiment is a combination of a method for traversing HTML nodes, and a metric for determining whether two DOM nodes ought to be considered similar. HTML structure is shaped like a tree. The traversal methods will start at the outer edges of the HTML tree structure with the assumption that layout related content is situated there. Each time two documents are compared, further and further in their tree structure according to the traversal strategy, a similarity metric is used to determine whether the next parts match. This results in a score for each combination of documents which can then be used to have statistical performance analyses of each of the 7 setups.

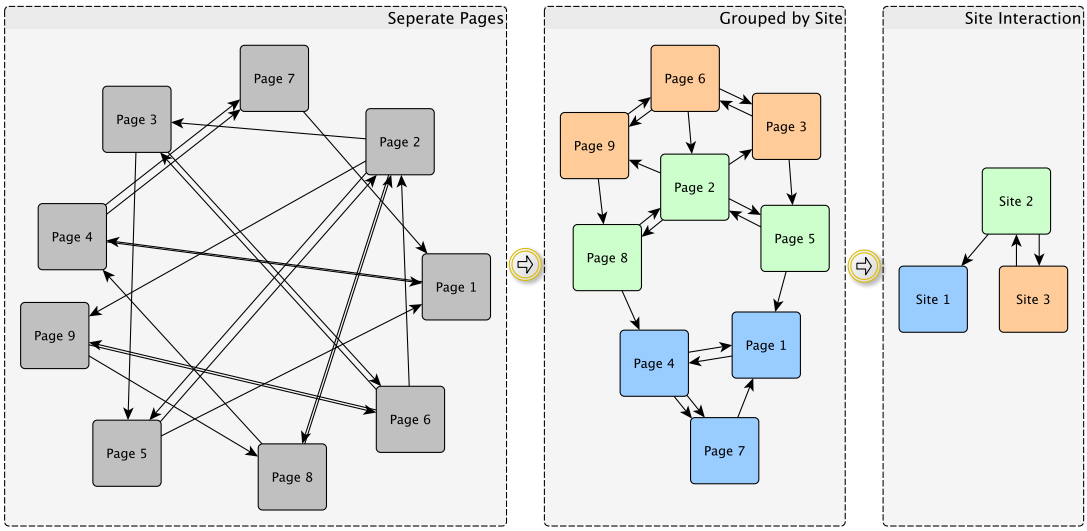

Figure 1: Going from a collection of pages, to modeled website interaction.

Sometimes, multiple conceptual websites are housed on a single domain. Think for example about the profile pages of researchers on a university website. Being able to cluster such domain sheltered websites offers the same benefits as having a single website per domain. In fig. 1 the process of going from a collection of pages (potentially within a single domain), to pages grouped, to finally the interactions between the groups is visualized. Those interactions can only be extracted if documents are not all considered to be in the same collection. Another benefit from looking at the overlap in web pages, even on a singular domain, is to be able to separate content from structure. This can be used in combination with other methods to better position the website in relation to others, or generating logical views for easier navigation; examples of which are referenced in the next section.

2 Related Work

The clustering of web documents is an extensive field. Usually documents are clustered cross-site as a means to group and search for information. However research into clustering documents inside a single website is done as well. Motivation varies from the alternative presentation of content to optimizing data-mining tasks. The body of work that I could locate that focuses on document structure as a metric for similarity was not extensive. Nor was information on as to why that might be.

2.1 Clustering within a single website

The clustering of documents based on the structure of their outlinks is researched in Crescenzi et al. [2005]. The research proposes a setup where from a single entry point, the algorithm starts visiting a representative number of web pages. A prototype has been developed and was used to perform experiments on life web sites. In the introduction Crescenzi et al. describe to have chosen this method over what they view is the issue with similar research: representative pages have to be manually selected. This research operates under the premise that if a page has similar groups of links (for example a list of HTML `<li>` tags in an un-ordered list in a `<div>` element) that the page is bound to be of similar structure. Levering structural similarities like this, the group wishes to automatically group classes of pages together belonging to a website, in order to supply the groups to the next step: building wrappers for content extraction. The algorithm performs this task by building a model of groups of links in the entry point. Selecting some, but not all documents the link group points to and repeating this process. This results in a collection of documents with in-links belonging to link-groups for which the model is used to judge structural similarity of the groups.

In Liu et al. [2004] the structure of individual web documents is used to infer the structure of the entire website. The goal of he research is to present the resulting structure as a logical view for easy navigation. In the abstract, the reason behind this is again given as automated content extraction. With a visual mapping tool, referred to as the mapping wizard, initial manual mappings are made to specific types of model component recognized by the team; ea "record", "item", "collection". Also common menu layout is exploited to find the highest level of content separation: pages. With a tool dubbed the "section extraction tool" groups of links that are present on multiple pages are presumed to be menu items and used to generate the highest level of the resulting logical tree structure. The next level is finding subsections on the pages themselves. The team have create another tool "the structure copier" to help assist the process of first selecting one, or more parts of web documents that can be deemed subsections, and allow the tool to discover parts with similar structure. The results section of the paper shows statistics on the performance of the setup on four major websites at the time. Satisfactory logical tree representations were achieved with as little as between 2.5 and 10.4 percent of extra manual modification.

In Blanco et al. [2011] URL structure as well as content features are used to build what is presented a highly scalable algorithm. The title and abstract suggest this setup is particularly suited for very large websites. For this paper 43 pages spanning 4 domains were used. The focus on URLS is done by destructering the address in it's domain and subsequent slash (`/`) separated sub folders, which are referred to as "tokens". All URLS with the same arity (that is: the same number of tokens) are grouped and encoded. This perspective is then combined with the actual text objects on the pages themselves. Not just the occurrence of such objects are considered ("address", "menu"), but also the position in the web document. For these documents are considered as trees, and with an *XPATH* query the element path from the root is taken as the value for comparing

an occurrence with others on different documents. The paper references quite a few other works in advocating the need for efficiency when actual content is included in the structuring, saying that "current state-of-the-art... " (as of 2011) " ...structural clustering techniques do not scale to large websites" . (sections 1 and 7).

2.2 Clustering over multiple websites

In He et al. [2002] three clustering methods are examined for web documents: K-means, multi-level METIS and normalized-cut. The research uses link references between pages, co-citation structures and content itself. In order to consider link structure all documents are considered as a graph, with links represented as the edges. Direction or frequency of the links are ignored. Textual similarity is here taken from the entire HTML document (not just anchor tags, or similar content containing elements). Tf.idf is used to determine similarity. Consequently this is then used to gauge the strength of the connection between two documents; with links between more similar documents judged to be stronger connections, resulting in a weighted graph. From here on out, graph edge-cut techniques are used to partition the graph in order to ultimately have a clustering of the documents represented in its edges.

In Flesca et al. [2007] document structure is used as well, but their approach goes about encoding documents based on the occurrences of different tags as opposed to assuming that "outside" structure is most indicative. Occurrences of tags are modeled as pulses on a timeline and similarity inferred by analyzing the frequencies of the corresponding Fourier transform. The motivation behind this approach is its attractive scaling function of $O(Nlog(N))$ w.r.t. the amount of unique encountered HTML tags. This is presented as performing rather better than techniques that were available at the time. An interesting observation made, is that while it is easy to determine whether two documents have identical structure, it is more difficult to find a measure in which to express the amount of similarity. In stead of using graph-matching, looking at documents as time series is proposed as the more efficient method. Before applying any of this setup, a method of transforming the original HTML to valid XML is added, in order to deal with unclosed or unmatched tags. This required validity being a consequence of having the nestedness, as well as amount of child tags be input for the amplitude in the time series.

3 Approach

In the referenced literature it becomes clear that it is considered reasonable to expect similar structure within pages belonging to a single website. Indeed, in Liu et al. [2004] this consistency is the basis of believing a logical view can be generated, once it is represented as a model. The ideas presented in Flesca et al. [2007] have arisen from the fact that using graph-matching is quite an expensive method. Triggered by the idea of "flattening" tree structure, and combined with the fact that the interest is not finding

similar content, but in fact similar (visual) layout, a different approach is presented here. As pointed out in Flesca et al. [2007] equivalency is much cheaper to detect, than coming up with a measure of similarity. This is why, with various variations, documents will be compared as trees, for as long as they are similar, breaking of the comparison as soon as they diverge. In order to verify the validity of document structure as indicator of pages belonging to the same website, seven experiments are conducted on a sample set. This section will give a high overview on the general approach and reasoning behind it that was used to acquire statistical data on the combination components as well as comment on the dataset itself.

3.1 Page structure

A distinguishing feature between web pages, is their layout. The reason to attempt to infer similarity of layout from HTML structure, is the hypothesis that this layout is reflected in the peripheral elements of the DOM hierarchy. An example of this is a web site, in which all pages consist of a header and footer, as well as a menu on the top left. All of which are wrapped in `<div>` tags with a specific name. In this case only looking at the top level elements of the `<body>` tag, all pages with this structure have a DOM tree which at its root and a few nodes in, is the same. Treating documents as trees they will be compared node by node. Intuitively comparison will stop, when things get "too different" (this will be expanded upon further in section 3.3 and more formally in section 4 and section 5).

3.2 Data set

To truly demonstrate the ability of separating pages on the same Domain, it would have been an obvious choice to select a large domain that houses a great number of different websites and either label them by hand, or find some url structure that enables automation. An approach less dependent on human labeling, however is to work with websites on separate domains and keep the algorithm oblivious as to where pages belong to. In this fashion, all pages are already tagged for inspection. For this research the websites that have been selected are housed at Mindconnect Webhosting, a shared hosting company of which I was co-owner. Since web pages are always changing I have included a link to an archive file which contains all referenced pages. This way results can be reproduced and verified at a later time. For steps to reproduce as well as links to required data see: section 8.2 in the appendix.

The set of pages P consists of 101 pages, spread across 20 domains in the domain set D. This gives each experiment a set of comparisons C in the set of possible comparisons in the set of pairs in $P \times P$. In this paper all similarity metrics are designed to be symmetric, thus only unique combination of pages are considered. Of which there are $\binom{101}{2} = 5050$.

3.3 Two parts per experiment

This paper describes seven experiments all consisting of two parts: A traversal strategy and a similarity metric. A traversal strategy takes two trees as its input and from the root node on, determines which two nodes to compare after that, and how to react given the similarity of two nodes. A similarity metric takes two nodes and returns a judgment on the similarity of those nodes. The reasoning behind and, the formal definition of, both parts warrant their own sections which directly follow this one. The reason for having an architecture that allows for swapping both parts is twofold: allowing for the separation of measuring how indicative certain parts of the web documents are, and through which lens. Secondly this makes it very easy for other researchers to implement and test their own ideas concerning both which part of the document would be most relevant and how to examine it.

4 Traversal strategies

In this section the various traversal tactics are discussed. These are ways in which the DOM tree is walked through as well as the behavior when a mismatch is encountered. The images used to visualize the elements that are going to be included for a certain strategy might need a little elaboration. When two trees are compared, unless they are the same document (which the algorithm prevents from happening in an earlier step), the trees are not going to be isomorphic. However, this does not prevent the visualization of the nodes that are deemed similar within a single tree. Consider that we have two trees, T_1, T_2. Let T_{ij} denote tree i, element j. Let C_{ij} donate the children of that element. Now each comparison is done with either the two root elements (T_{11}, T_{21}), or two elements that have those roots as an ancestor through a line of descendants that have also been deemed similar. If for any such a succession there is an equal number of children in $C_{ij}(i \in \{1, 2\})$ and they are all similar, there is no problem with representing the path as a single tree. Else when comparing nodes in one direction (left to right, or the reverse, more on this later) either we run into a mismatch, or one of the sides runs out of nodes. Since in both cases no more similarity comparisons are done over elements in C_{ij} traversal is both commutative over T_1 and T_2 and the elements included in any similarity score can be identified on either. This means that for traversal visualization, one tree will suffice.

For each of the visualizations, the squares represent DOM nodes, or elements. The green squares are elements that have been deemed similar by the selected similarity metric. The red squares represent a mismatch, which can happen on three occasions:

- The nodes are dissimilar;
- T_1 has one or more child-nodes to evaluate while T_2 does not for this level;
- The converse of the previous statement.

Finally the green outline denotes which nodes are considered and thus count towards the similarity score.

4.1 Simple Breadth-first

The first and simplest strategy is a basic breadth-first approach. Elements from the root node are compared left to right, and when no mismatch is encountered, all elements are then treated as new root nodes and the process continues recursively. In this strategy a mismatch is either: not similar, or one of the two collections C_{ij}, $i \in \{1, 2\}$ runs out before the other.

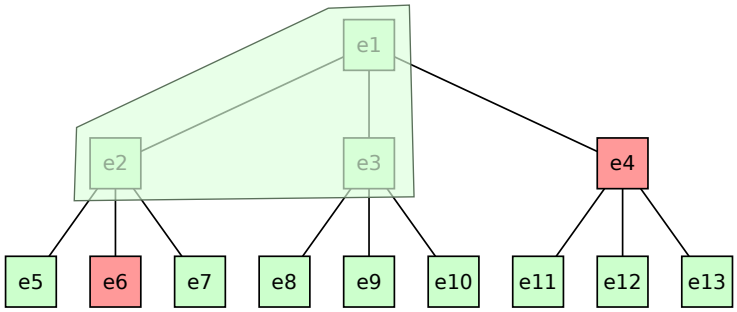

Figure 2: Inclusion of elements in basic breadth-first traversal

Element $e8$ is not considered, because the run stops after a mismatch (see: algorithm 1).

Algorithm 1 Basic breadth-first

Require: isSimilar $\leftarrow [function]$
  function breadthFirstCompare(e_1, e_2)
    $s \leftarrow 0$
    while $c_1 \leftarrow e_1.childElements.next()$ **do**
      if $c_2 \leftarrow e_2.childElements.next()$ **then**
        if isSimilar(c_1, c_2) **then**
          $s++$
          $similarElements.add([c_1, c_2])$
        else
          return s
    if $e_2.ChildElements.next()$ **then**
      return s ▷ don't enter recursion
    for $i = 0; i < e_1.childCount; i++$ **do**
      $s \leftarrow s$+breadthFirstCompare($e_1.childElements[i], e_2.childElements[i]$)
    return s
  $result \leftarrow$ breadthFirstCompare(T_{11}, T_{21})

As can be seen, isSimilar is assumed to be defined elsewhere; Possible implementations of such functions are discussed in the next section. The function is first started on the two root nodes (the `<body>` element, of T_1 and T_2). If the iteration of the direct descendants of those elements can terminate without the method returning, the process will recursively continue on the collection of child nodes C_{11} and C_{21}.

4.2 Exhaustive traversal

The basic breadth-first approach does not cover those page structures well that do not have a top heavy level of hierarchical ordering. This is were the exhaustive traversal comes in. Nodes are still considered breadth-first from left to right, but a mismatch only ends comparisons on the children of an assumed shared parent between the two trees.

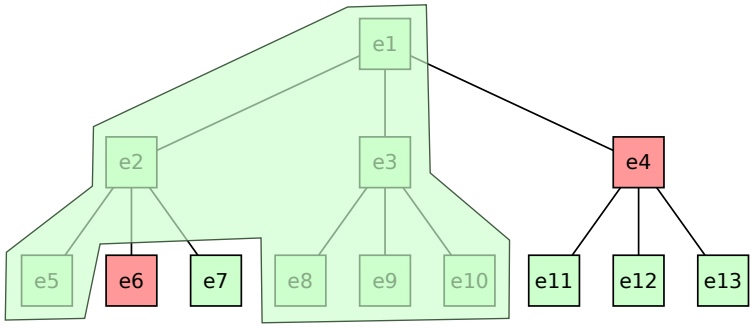

Figure 3: Inclusion of elements in exhaustive traversal

Algorithm 2 Exhaustive traversal

Require: isSimilar $\leftarrow [function]$
  function exhaustiveCompare(e_1, e_2)
    $s \leftarrow 0$
    $similarElements \leftarrow \{\}$ ▷ keep track of verified elements
    while $c_1 \leftarrow e_1.childElements.next()$ **do**
      if $c_2 \leftarrow e_2.childElements.next()$ **then**
        if isSimilar(c_1, c_2) **then**
          $s++$
          $similarElements.add([c_1, c_2])$
        else
          break
    for all $[e_1, e_2] \in similarElements$ **do**
      $s \leftarrow s$+exhaustiveCompare(e_1, e_2)
    return s
  $result \leftarrow$ exhaustiveCompare(T_{11}, T_{21})

As can be seen in algorithm 2, previously the traversal returned on either a mismatching number of direct descendants, or a mismatch. This time the collection of nodes that *did* match, is still considered and "exhausted". Intuitively the algorithm collects nodes that are considered matching, and stops on a mismatch only for that "level". Then it takes the collection and goes to look for more similarity.

4.3 Exhaustive right to left traversal

Right to left traversal is used to, given a particular similarity metric, test the hypothesis that documents might be more similar on the bottom right, than the top left. One might imagine a very simple header, and/or menu on the top, but an elaborate footer with disclaimers and various links to for example contact information. It might be interesting to see whether invariant site structure for a sample set resides not in the obvious place.

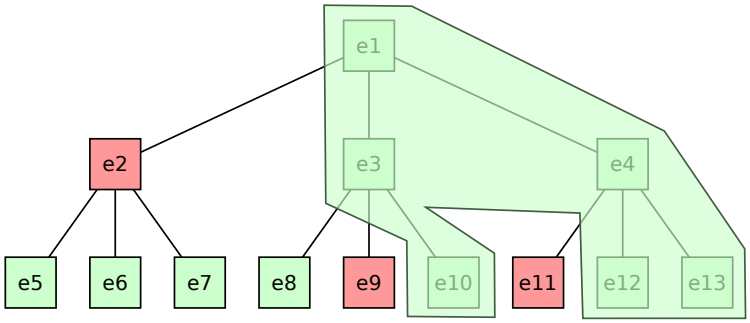

Figure 4: Inclusion of elements in exhaustive right to left traversal

Algorithm 3 Exhaustive right to left traversal

```
Require: isSimilar ← [function]
  function exhaustiveRtLCompare(e1, e2)
    s ← 0
    similarElements ← {}
    e1.childElements.iteratorIndex(e1.childCount)          ▷ reverse iteration
    e2.childElements.iteratorIndex(e2.childCount)
    while c1 ← e1.childElements.previous() do
      if c2 ← e2.childElements.previous() then
        if isSimilar(c1, c2) then
          s++
          similarElements.add([c1, c2])
        else
          break
    for all [e1, e2] ∈ similarElements do
      s ← s+exhaustiveRtLCompare(e1, e2)
    return s
  result ← exhaustiveRtLCompare(T11, T21)
```

Intuitively this method is very similar to the previous one. The only difference is the order in which nodes are considered. Because of the different order nodes that where previously excluded are now considered and vice versa!

4.4 Bidirectional traversal

Bidirectional traversal combines algorithm 2 and algorithm 3. As can be seen in fig. 5 each collection of children is approached exhaustively from the left *and* following that from the right. Of course on any giving *level* nodes are still considered only once. So when all nodes in a child collection are deemed similar, the right-to-left step, will simply return immediately.

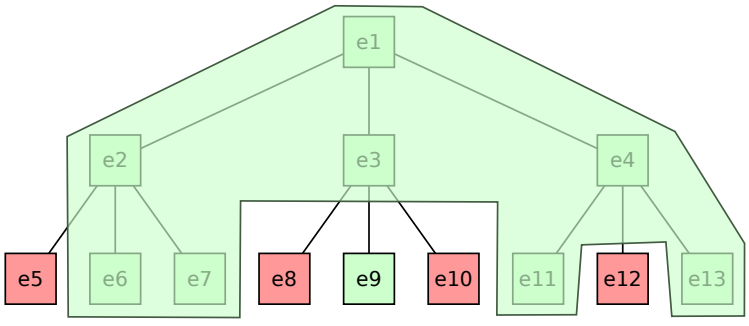

Figure 5: Inclusion of elements in bidirectional exhaustive traversal

This combined approach can reach elements which neither single directive approach could have reached. Doing so has of course the risk of upsetting the balance between excluding part of the structure and including part of the content, so it is probably best to use with a very strict similarity measure. This effect will be explored in the experiments section. Still this setup will test whether looking for overlap in structure in both the top-left and bottom-right of a website yields better results then only considering either.

Algorithm 4 Bidirectional exhaustive traversal

```
Require: isSimilar ← [function]
  function bidirectionalExhaustiveCompare(e1, e2)
    s ← 0
    similarElements ← {}
    while c1 ← e1.childElements.next() do
      if c2 ← e2.childElements.next() then
        if isSimilar(c1, c2) then
          s++
          similarElements.add([c1, c2])
        else
          break
    e1.childElements.iteratorIndex(e1.childCount)          ▷ reverse iteration
    e2.childElements.iteratorIndex(e2.childCount)
    while c1 ← e1.childElements.previous() do
      if c2 ← e2.childElements.previous() then
        if [c1, c2] ∉ similarElements ∧ isSimilar(c1, c2) then
          s++
          similarElements.add([c1, c2])
        else
          break
    for all [e1, e2] ∈ similarElements do
      s ← s+bidirectionalExhaustiveCompare(e1, e2)
    return s
  result ← bidirectionalExhaustiveCompare(T11, T21)
```

In this traversal strategy the two collections of child nodes, for every two parent nodes, is iterated over from left to right, and from right to left. A mismatch, will stop one of the iterations, but not the other. Of course, every combination of nodes is only considered once.

5 Similarity Metrics

In this section the used metrics for DOM node similarity are discussed. Since the comparison of two nodes is much simpler than deciding how to traverse two non-isomorphic trees simultaneously this section is going to be somewhat brief by comparison. All metrics here are binary; two nodes are either similar, or they are not. This property has certain consequences for those aspects of similarity metrics that are usually discussed. Since there is only similar or not similar, there is no concept of "distance". Sometimes when discussing similarity of documents, properties are mapped to a vector space, for example viewing documents in terms of normalized word occurrences. Another consequence that might follow somewhat trivially from viewing two nodes as being either being similar, or not, is symmetry and transitivity. For each of the metrics discussed below, certain properties of a node are considered, with the judgment of being similar being a function of the matching of those properties.

5.1 Tag type matching

The first and most obvious matching strategy is comparing just the 'type' of two DOM nodes. This refers to the XML tagname by which the node is declared in the HTML syntax. Examples are `<div>`, or `<table>`. In practice two `<table>` elements could contain completely different contents. The assumption here is however, that if they are contained in the same place in the hierarchy of the trees, this might be indicative of the sharing a common structure.

5.2 Tag type and Element Id matching

Some websites might use `<table>`elements to define their layout, where in another by the time such elements are reached is already presenting content. Such pages might then be deemed "similar" if these element types are encountered roughly in the same place of the tree. To counteract this somewhat, additional properties of the node can be considered; in this case: the 'id' attribute. This attribute is to be a unique yet optional identifier for the element, should later reference or manipulation outside its original definition take place W3Schools [2014]. The reason that it is interesting to even consider Tag type *without* looking at the id attribute as well, is that this attribute is sometimes used to differentiate between pages within the same website. A common example of this, is adding `id="currentPage"` to one of the `<li>` elements making up a navigational menu.

5.3 Number of child elements matching

The final metric that is used in the experiments, is comparing the number of direct descendants. In other words nodes T_{1n} and T_{2n} are similar if the number of elements in C_{1n} is the same as those in C_{2n}. The assumption being as before, that pages of the

same website, share some structure just before (and possibly after) their page specific content. This metric can help determine whether looking at tree structure regardless of the elements that make up that structure is indicative of pages belonging together.

6 Experiments

In this section various runs on the data and their results are discussed. An experiment will consist of a traversal strategy (such as "Exhaustive") and a matching strategy (such as "Tag type matching"). The reasoning behind the combinations and interpretations of the results will receive attention as well.

6.1 Numbers in the results

For each of the different combinations of options metrics are discussed indicating how much promise a particular setup shows. These metrics are split between the positives and negatives. When discussing positives and/or negatives, unless otherwise noted, this will always concern true e.a. pre-labeled positives/negatives. Together these numbers give a clear answer whether it is possible or not, to draw a line between the positives and negatives given the scores the experiment has come up with, if not for all possible data sets.

6.1.1 Average match score

The average number of nodes to be considered similar between two documents. For the positive matches (those labeled beforehand), this ought to be considerably higher then the negative ones. These numbers by themselves might be indicative of a combination able to draw a line in the sample set, but not conclusive. The symbols used for these metrics will be μ_+ and μ_- for positives and negatives respectively.

6.1.2 Standard deviation

The standard deviation among both the positive pairs, and the negative ones. This metric can be used to evaluate distance between numbers pertaining to both sides. If for example the scores of the positives and negatives are close, with a high standard deviation, this does not bode well. The symbols used for these metrics will be σ_+ and σ_- for positives and negatives respectively.

6.1.3 Minimum and maximum match score

The minimum and maximum score found for any two documents, both belonging to the same domain, and from separate domains. For an experiment to be able to draw a line between positives and negatives, these numbers should not overlap. Preferably there should be multiple standard deviations between them. These metrics will be represented by Min_+ and Max_+ for positives and Min_- and Max_- negatives respectively.

6.1.4 Mean standard deviation within domain

The mean of standard deviations computed for combinations of documents within the same domain. The lower this number, the closer the similarity scores between two arbitrary documents being a labeled positive match. This metric gives direct insight into how much true positive scores within the same domain differ from each other for a given experiment. Since the figure focuses on documents withing a domain, there is no equivalent for true negatives. This metric will be represented by $\mu[\sigma^{dom}]$.

6.1.5 Mean delta of minimum and maximum score within domain

As the section title implies, this figure is calculated by taking the maximum similarity score for each domain, and subtracting the minimum score that is still within that same domain. Like the previous metric, this one gives a strong indication of the amount of similarity between labeled positive matches and how much this is stable across domains. This metric will be represented by $\mu[\delta^{dom}_{min/max}]$.

6.2 Graphs in the Results

The experiment results are visualized in multiple graph types to illustrate the performance of a given setup. The first experiment: Basic breadth-first traversal with tag type matching, has additional graphs, to clearly show additional perspectives of the results view. When looking at fig. 6, fig. 8 and fig. 9 it is clear to see that combinations are mapped to positions on a two-dimensional plane, with the similarity score indicated by the position on the Z-axis. Once this concept has been made clear, the subsequent result section will not be providing these perspectives. In all graph types positives are indicated by green plus signs (+), while negatives are indicated by red crosses (×).

6.3 Basic breadth-first traversal with tag type matching

The first experiment is going to be using the basic breadth-first traversal strategy specified in algorithm 1. The matching strategy is tag type matching, meaning that only the DOM element type is considered for similarity.

6.3.1 Results

Statistic	Positives	Negatives
Average match score; μ_+/μ_-	78.7381	0.7165
Standard deviation; σ_+/σ_-	63.6031	0.9876
Minimum match score; Min_+/Min_-	1	0
Maximum match score; Max_+/Max_-	280	7
Mean stdev / domain; $\mu[\sigma^{dom}]$	21.9179	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	53.4000	na

Table 1: Results; Basic breadth-first traversal with tag type matching

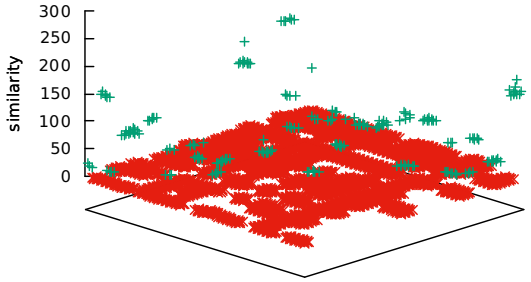

Figure 6: Breadth-first/Tag Isometric

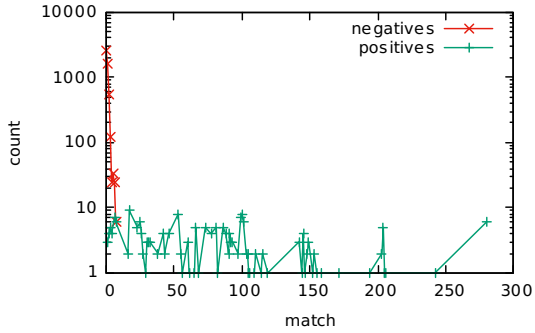

Figure 7: Breadth-first/Tag Matches

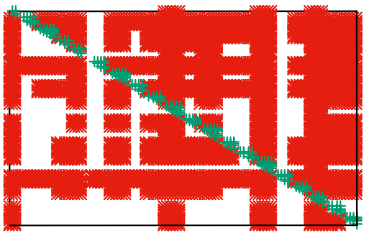

Figure 8: Breadth-first/Tag Top

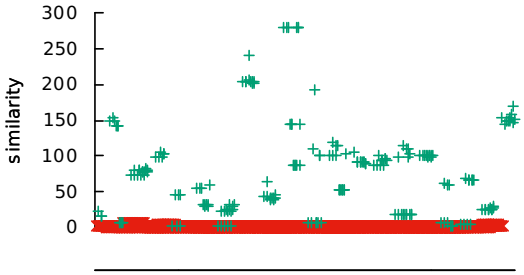

Figure 9: Breadth-first/Tag Front

Because of the number of experiments, the graph format in which results will be presented for subsequent sections will just be the "frontal" view with positives and negatives in a separate graph. This has the benefit of being able to better zoom in the spread of the negative similarity scores, as well as being much easier to interpret in black and white.

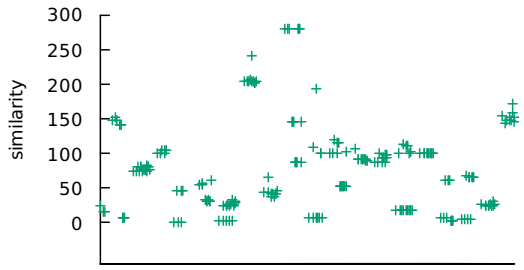

Figure 10: Breadth-first/Tag Positives

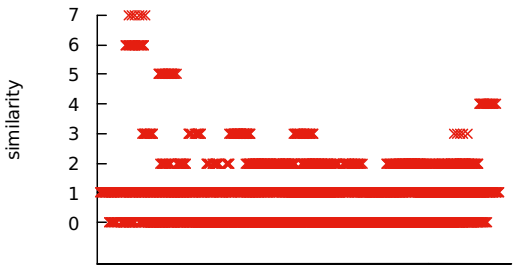

Figure 11: Breadth-first/Tag Negatives

6.3.2 Interpretation

On the one hand, the average positive and negative scores μ_+ and μ_- are far apart. When focusing on how negatives are judged, with both μ_- and σ_- less then 1, it seems the conservative nature of basic breadth-first keeps most scores very low.

However still looking at Min_+ there is more then 7 standard deviations worth of overlap considering σ_-. Coming from the positive side μ_+ is only 1.13 standard deviations removed from negative match scores.

6.4 Exhaustive traversal with tag type matching

This experiment uses the exhaustive traversal strategy, meaning that the DOM is explored both in depth and breadth. Whenever the binary matching function evaluates to false, exploration stops for that branch. The tag comparison strategy simply evaluates to true if two nodes of the DOM are of the same html tag type.

6.4.1 Results

Statistic	**Positives**	**Negatives**
Average match score; μ_+/μ_-	97.0762	1.8700
Standard deviation; σ_+/σ_-	57.6702	3.0552
Minimum match score; Min_+/Min_-	19	0
Maximum match score; Max_+/Max_-	280	16
Mean stdev / domain; $\mu[\sigma^{dom}]$	11.4338	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	32.6500	na

Table 2: Results; Exhaustive traversal with tag type matching

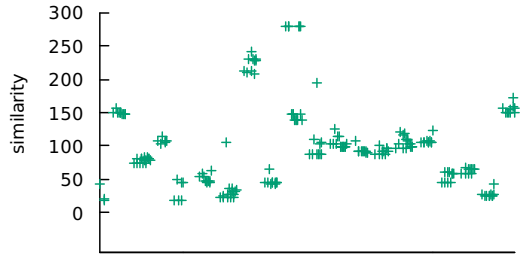

Figure 12: Exhaustive/Tag Positives

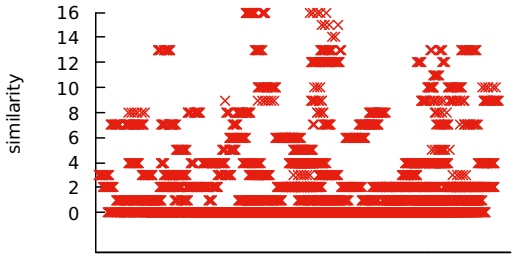

Figure 13: Exhaustive/Tag Negatives

6.4.2 Interpretation

This experiment resulted in scores that have a separation point between the negatives and the positives with Min_+ and Max_- not overlapping. μ_- is 5.61 standard deviations away from a positive score. For μ_+ this is 1.41. Being less conservative then basic breadth-first this setup performed better at differentiating between related and unrelated documents on average, but worse in preventing unrelated documents from getting a higher score. Within a domain, the scores were more stable, with $\mu[\sigma^{dom}]$ score being almost half as low as that of the previous experiment.

6.5 Exhaustive right to left traversal with tag type matching

This experiment uses the exhaustive right to left traversal strategy, meaning that the DOM is explored on each level from right to left. The tag comparison strategy remains the same, in that it evaluates to true, if two DOM-nodes have the same type.

6.5.1 Results

Statistic	**Positives**	**Negatives**
Average match score; μ_+/μ_-	84.7905	1.1591
Standard deviation; σ_+/σ_-	61.4278	2.1332
Minimum match score; Min_+/Min_-	0	0
Maximum match score; Max_+/Max_-	280	15
Mean stdev / domain; $\mu[\sigma^{dom}]$	23.2159	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	59.2500	na

Table 3: Results; Exhaustive right to left traversal with tag type matching

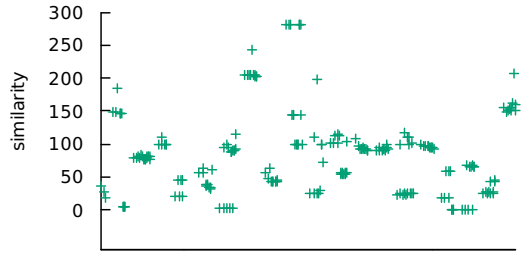

Figure 14: ExhaustiveRtL/Tag Positives

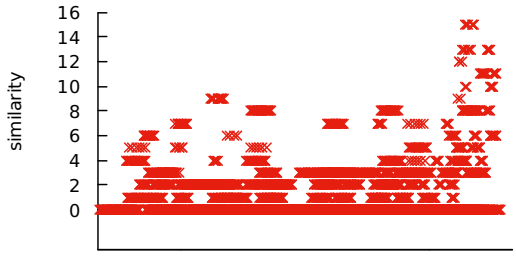

Figure 15: ExhaustiveRtL/Tag Negatives

6.5.2 Interpretation

This experiment shows that for the test set more overlap is in the top left, than in the bottom right. This conclusion can be drawn from the lower μ_+ and the much lower Min_+ scores. A direct consequence of the later, is complete overlap with the negative matches. The Max_+ being 280 is caused by the fact that for at least some combinations of documents, they are structurally identical to each other when looking at just element types as is done in the tag type metric. Finally, when comparing with the previous experiment the $\mu[\sigma^{dom}]$ is more then twice as high, which means much more variation within a domain.

6.6 Exhaustive bidirectional traversal with tag type matching

This experiment uses the exhaustive bidirectional traversal strategy, that combines algorithm 2 and algorithm 3. This will be the final experiment in which the tag matching strategy is used. Coming in from both directions at each level takes longer of course. This experiment will test, whether it leads to more accurate results.

6.6.1 Results

Statistic	**Positives**	**Negatives**
Average match score; μ_+/μ_-	102.3857	3.0395
Standard deviation; σ_+/σ_-	55.1989	4.0250
Minimum match score; Min_+/Min_-	24	0
Maximum match score; Max_+/Max_-	280	24
Mean stdev / domain; $\mu[\sigma^{dom}]$	11.7409	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	33.4500	na

Table 4: Results; Exhaustive right to left traversal with tag type matching

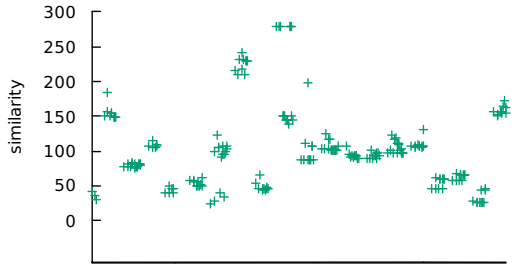

Figure 16: Bidirectional/Tag Positives

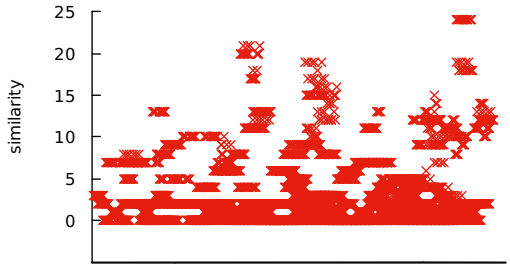

Figure 17: Bidirectional/Tag Negatives

6.6.2 Interpretation

Looking at Min_+ and Max_- there is slight overlap which speaks against this method being able to differentiate between the two classes. It is not surprising that overall scores are higher, since more nodes get a chance at comparison. On the whole the results in listing seem like a strict regression on all fronts compared to those presented in section 6.4.

6.7 Exhaustive traversal with tag type and Id matching

This experiment uses the normal exhaustive traversal strategy from algorithm 2 and uses both tag and id to determine similarity. Looking at both tag and id is of course more strict and might pull positive matches where similar element types receive (perhaps generated) id's based on the content they contain. It will however also rule out unrelated documents that happen to use the same style of formatting relying heavily on `<div>` and `<span>` wrapper elements.

6.7.1 Results

Statistic	**Positives**	**Negatives**
Average match score; μ_+/μ_-	89.3810	0.0616
Standard deviation; σ_+/σ_-	56.6056	0.5640
Minimum match score; Min_+/Min_-	10	0
Maximum match score; Max_+/Max_-	280	7
Mean stdev / domain; $\mu[\sigma^{dom}]$	14.2961	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	39.2000	na

Table 5: Results; Exhaustive traversal with tag type and Id matching

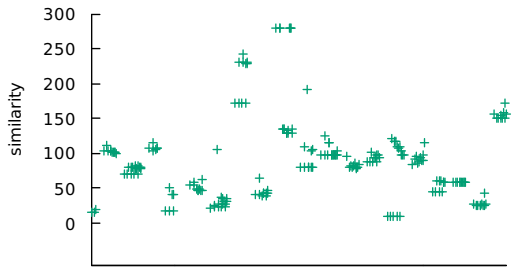

Figure 18: Exhaustive/Tag+Id Positives

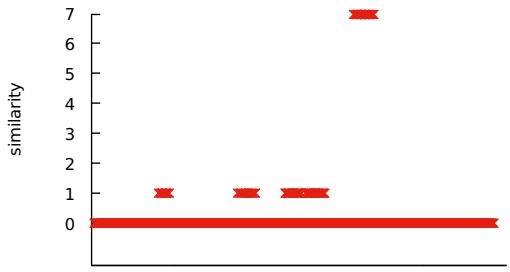

Figure 19: Exhaustive/Tag+Id Negatives

6.7.2 Interpretation

When looking at fig. 19 it is immediately obvious that for negatives this setup performs significantly different than the experiments so far. Except for a few rare outliers all match scores are either 1, or in most cases 0. Looking at the results in table 5 μ_- and σ_- back this up nicely.

Conversely comparing fig. 18 with figures from earlier experiments yields only slightly more conservative results. It's interesting to note that when comparing with exhaustive with tag matching (table 2), that although the σ_+ is slightly lower, within a domain, $\mu[\sigma^{dom}]$, it's actually higher. Presumably the first is mostly influenced by the overall lower scores, while the stricter similarity metric is causing a bit more variation within domains.

This experiment is up until now the most promising combination with a clear divide between most documents in the two classes.

6.8 Exhaustive bidirectional traversal with tag type and Id matching

This experiment uses the bidirectional traversal strategy from algorithm 4 and uses both tag and id to determine similarity. For just tag matching, bidirectional performed slightly and strictly worse than left to right traversal, this experiment test whether this is the case for a more stricter similarity metric.

6.8.1 Results

Statistic	Positives	Negatives
Average match score; μ_+/μ_-	97.6000	0.5407
Standard deviation; σ_+/σ_-	55.4299	1.2572
Minimum match score; Min_+/Min_-	10	0
Maximum match score; Max_+/Max_-	280	8
Mean stdev / domain; $\mu[\sigma^{dom}]$	14.2629	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	38.4500	na

Table 6: Results; Exhaustive bidirectional traversal with tag type and Id matching

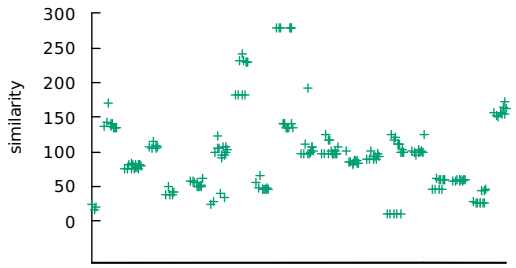

Figure 20: Bidirectional/Tag+Id Positives

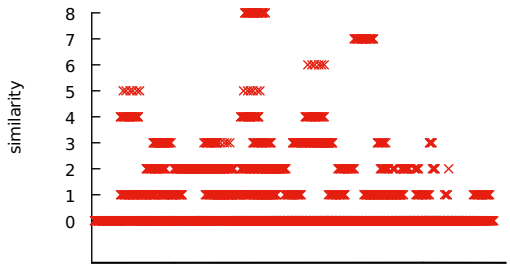

Figure 21: Bidirectional/Tag+Id Negatives

6.8.2 Interpretation

Again starting with the visual representation of the negatives in fig. 21 it seems that while tag+id is an improvement over just looking at tags, bidirectional is a regression compared to left to right. Positives get overall higher scores, but the significant change is where the two meet; looking at the results in table 6 it is immediately obvious that a higher μ_- and σ_- make it more difficult to distinguish between similar and dissimilar documents.

6.9 Exhaustive traversal with child count matching

This experiment uses the left to right exhaustive traversal strategy from algorithm 2 and uses node child count to determine similarity. This metric looks purely at graph structure by counting child nodes.

6.9.1 Results

Statistic	Positives	Negatives
Average match score; μ_+/μ_-	74.3810	0.2132
Standard deviation; σ_+/σ_-	61.7126	0.7708
Minimum match score; Min_+/Min_-	0	0
Maximum match score; Max_+/Max_-	280	7
Mean stdev / domain; $\mu[\sigma^{dom}]$	21.1105	na
Mean delta Min, Max; $\mu[\delta^{dom}_{min/max}]$	52.1000	na

Table 7: Results; Exhaustive traversal with child count matching

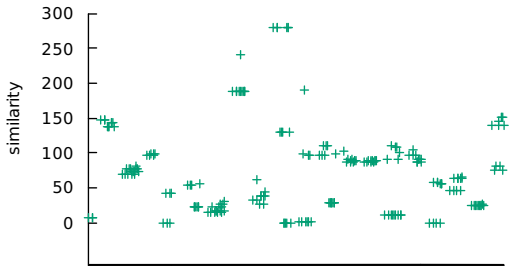

Figure 22: Exhaustive/Child Positives

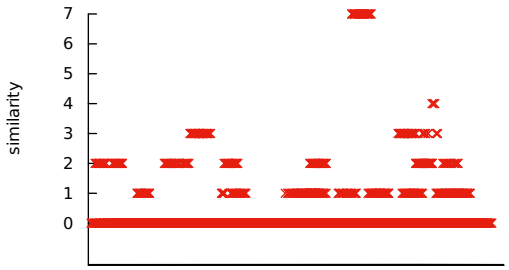

Figure 23: Exhaustive/Child Negatives

6.9.2 Interpretation

The look-ahead of counting childnodes proves to be stricter then examining tag types for both negatives and positives. In both the table and in fig. 23 scores are lower than those discussed in section 6.4. Comparing with tag+id similarity it fall short. It also has low scoring positive outliers, which causes overlap between the two classes. Given that child counting performs worse in the otherwise best scoring traversal method there is no reason to explore it further.

6.10 Reflection

The hypothesis before doing any of the experiements, was that the structure of `html` documents would be a strong indicator of documents belonging to the same website. For even the arguably worst runs - basic breadth-first/tag and exhaustive/child count - the scores were distributed in a way that calling them "indicative" doesn't seem like overstating. Looking at the best performing setup - exhaustive traversal combined with tag and id matching - the divide becomes sufficiently clear, that it becomes feasible to implement automated clustering. Of course it's all well and good to be able to draw a line between pre-labeled clusters, given the vast and ever changing nature of the internet,

it is perfectly conceivable that on average the measure in which documents are more, or less similar to one another will change over time. The results are not perfect. Even being able to prevent overlap in a small testset does nothing to guarantee this will be the case for the internet at large.

Another challenge comes from the vast difference in which different websites seem to overlap in structure within their own ranks. Especially structurally complex websites, yield high scores, since it takes so long to get from layout to content. This effect can well be seen in the extended result section in the Appendix. It is for this reason that within this paper the properties of positives and negatives are kept so separate. For if the combination of positive and negative matches proves very defuse, it might still be the case that one might keep itself within boundaries with little variance. The best case of this is visible in table 5. With μ_- of 0.06 and σ_- of only 0.56.

7 Conclusion

After definition, implementation, and interpretation of each set of results, this section will reflect on whether the hypothesis is confirmed and where to go from here.

The hypothesis that was the basis for this research was that document structure is indicative of pages belonging to the same website. I feel this has been shown to be the case for the test set. Multiple experiments were able to draw a perfect line between positives and negatives. The question one might ask now is: "Is document structure *sufficiently* indicative to have practical applications?". Well, for me personally it was, and had, as I will discuss in the next subsection.

7.1 Impact and application

While not the initial reason for this research topic, one very obvious application was applying the setup to websites hosted on commercial servers to see if customers are playing "fair". By collecting documents through parsing web server logs, and eventually running with the most successful combination - exhaustive traversal with tag and id match - we were able to detect a few accounts that technically violated the terms of service. This results in a domain with some documents being "quite" similar to each other and others completely (scores lower than 2) dissimilar. In all cases upon human verification more then one discrete website was being hosted (usually for testing purposes) on a single service account.

Also, the diverse properties of matching websites can have other interesting applications. If it is already known that a set of documents belong to one another, the part that they have in common, could indicate where structure ends, and content begins, for example.

As for impact, I personally have no real interest in pursuing this topic further. It might be that the results, and/or source code can someday help someone. Perhaps as complementary to other document clustering work, or grouping pages on a single domain together as separate bodies of information.

7.2 Future work and extensions

Possibilities to extend this research should, I feel, start with larger data sets. Comparing the results with those presented in this paper. Also it could be interesting to combine the methods used, with other similarity measures that look at content specifically. Thus combining distance in feature space, with distance in structure.

8 Appendix

8.1 Extended result listings

In this section some extended results are printed that give more in depth view on the performance of a particular combination of traversal and matching type.

Table 8: Extended results; Basic breadth-first traversal with tag type matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	152.40	7.3919	144	171
barbaragildevught.nl	90.10	68.7509	6	153
alerthygiene.nl	100.20	0.7483	99	101
deboschkens.com	77.33	3.4960	73	82
mindcontrolled.nl	26.00	1.5492	25	30
artesse.nl	41.20	12.5762	30	60
lotusvitaal.nl	78.40	27.6810	53	119
kookze.nl	18.47	11.2538	3	32
digipluggen.nl	92.60	4.5211	90	106
ttuin.nl	52.53	43.6744	17	114
financewerving.nl	101.67	2.6874	99	105
doelgerichtfysio.nl	20.70	26.0002	2	61
tintelingen.nl	179.33	85.2734	86	280
deputyit.nl	92.50	4.2249	88	100
voedselbank.my-mc.nl	41.40	30.5424	4	68
greenart.nl	54.70	63.4761	7	194
krebworkwear.eu	23.50	22.5000	1	46
sportprijzentiel.nl	18.33	3.2998	16	23
immotix.nl	44.10	7.3137	38	65
bi-kring.nl	207.90	11.3969	203	242

Table 9: Extended results; Exhaustive traversal with tag type matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	155.10	6.7000	150	173
barbaragildevught.nl	149.30	2.8653	147	157
alerthygiene.nl	107.60	5.5172	105	124
deboschkens.com	78.60	2.8000	75	82
mindcontrolled.nl	27.60	5.5714	25	44
artesse.nl	51.20	5.4918	46	62
lotusvitaal.nl	103.67	7.8117	98	125
kookze.nl	33.67	19.6254	22	105
digipluggen.nl	93.30	4.6701	90	107
ttuin.nl	105.13	8.2532	97	122
financewerving.nl	107.67	3.5434	104	115
doelgerichtfysio.nl	53.80	7.2083	45	61
tintelingen.nl	198.07	66.9731	139	280
deputyit.nl	92.60	4.4091	88	101
voedselbank.my-mc.nl	63.00	4.1231	58	68
greenart.nl	103.40	31.3949	87	194
krebworkwear.eu	33.17	14.2293	19	50
sportprijzentiel.nl	27.33	10.4030	19	42
immotix.nl	46.70	6.1652	43	65
bi-kring.nl	223.60	10.9197	209	242

Table 10: Extended results; Exhaustive right to left traversal with tag type matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	159.40	16.0947	148	206
barbaragildevught.nl	93.40	73.7837	4	184
alerthygiene.nl	95.80	1.4000	93	98
deboschkens.com	78.80	2.5087	76	83
mindcontrolled.nl	30.80	8.2195	25	44
artesse.nl	44.80	11.9482	31	63
lotusvitaal.nl	79.40	25.8710	54	114
kookze.nl	63.53	43.9680	2	115
digipluggen.nl	94.00	4.7329	90	107
ttuin.nl	56.87	40.1063	23	117
financewerving.nl	101.50	3.3541	100	109
doelgerichtfysio.nl	23.20	24.3549	1	59
tintelingen.nl	183.40	80.9047	98	280
deputyit.nl	92.60	2.5377	90	98
voedselbank.my-mc.nl	39.90	32.5836	0	68
greenart.nl	70.40	53.8576	25	197
krebworkwear.eu	32.67	12.6711	20	46
sportprijzentiel.nl	27.00	7.3485	18	36
immotix.nl	47.00	6.6182	43	63
bi-kring.nl	207.70	11.4547	203	242

Table 11: Extended results; Exhaustive bidirectional traversal with tag type matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	157.90	6.5338	150	173
barbaragildevught.nl	154.30	10.7056	148	185
alerthygiene.nl	108.70	7.1840	105	130
deboschkens.com	79.40	2.3036	76	83
mindcontrolled.nl	31.30	8.5563	25	45
artesse.nl	53.90	4.3232	49	62
lotusvitaal.nl	105.40	7.3194	101	125
kookze.nl	83.67	32.3948	24	122
digipluggen.nl	93.80	4.6217	90	107
ttuin.nl	105.20	8.3921	97	123
financewerving.nl	107.83	3.5785	104	115
doelgerichtfysio.nl	54.30	6.7978	46	61
tintelingen.nl	200.20	65.2301	139	280
deputyit.nl	93.60	3.8262	90	101
voedselbank.my-mc.nl	63.00	4.1231	58	68
greenart.nl	104.50	32.4815	87	198
krebworkwear.eu	43.17	4.3748	39	50
sportprijzentiel.nl	35.67	5.3125	29	42
immotix.nl	48.30	6.5582	43	66
bi-kring.nl	224.40	10.2000	209	242

Table 12: Extended results; Exhaustive traversal with tag type and Id matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	155.10	6.7000	150	173
barbaragildevught.nl	102.90	2.9479	100	111
alerthygiene.nl	92.90	8.3958	84	115
deboschkens.com	77.27	4.5821	71	82
mindcontrolled.nl	27.60	5.5714	25	44
artesse.nl	51.20	5.4918	46	62
lotusvitaal.nl	101.87	8.6708	97	125
kookze.nl	33.67	19.6254	22	105
digipluggen.nl	83.10	4.5266	78	95
ttuin.nl	75.07	46.5324	10	121
financewerving.nl	107.67	3.5434	104	115
doelgerichtfysio.nl	53.80	7.2083	45	61
tintelingen.nl	191.53	72.2559	130	280
deputyit.nl	92.60	4.4091	88	101
voedselbank.my-mc.nl	58.60	0.4899	58	59
greenart.nl	98.90	32.8495	80	191
krebworkwear.eu	31.33	13.5974	18	50
sportprijzentiel.nl	16.33	1.8856	15	19
immotix.nl	44.90	7.0349	39	65
bi-kring.nl	208.00	29.6041	172	242

Table 13: Extended results; Exhaustive bidirectional traversal with tag type and Id matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	157.90	6.5338	150	173
barbaragildevught.nl	140.30	10.7056	134	171
alerthygiene.nl	102.40	7.3919	96	124
deboschkens.com	78.80	3.0155	75	83
mindcontrolled.nl	31.30	8.5563	25	45
artesse.nl	53.90	4.3232	49	62
lotusvitaal.nl	102.60	8.8979	97	125
kookze.nl	83.67	32.3948	24	122
digipluggen.nl	86.60	5.1614	81	101
ttuin.nl	76.33	47.4140	10	125
financewerving.nl	107.83	3.5785	104	115
doelgerichtfysio.nl	54.30	6.7978	46	61
tintelingen.nl	194.07	70.2002	134	280
deputyit.nl	93.60	3.8262	90	101
voedselbank.my-mc.nl	58.60	0.4899	58	59
greenart.nl	111.00	27.6478	98	193
krebworkwear.eu	41.33	4.2687	38	50
sportprijzentiel.nl	19.67	2.8674	16	23
immotix.nl	49.10	6.4413	45	66
bi-kring.nl	212.00	24.7467	182	242

Table 14: Extended results; Exhaustive traversal with child count matching

Domain	**Mean** μ	**Stdev** σ	Min	Max
mindconnect.nl	118.10	32.2752	76	152
barbaragildevught.nl	142.50	3.8536	138	147
alerthygiene.nl	92.30	5.2735	87	104
deboschkens.com	74.47	3.7035	70	81
mindcontrolled.nl	24.30	0.6403	24	26
artesse.nl	35.60	15.4415	23	56
lotusvitaal.nl	63.07	37.7756	28	111
kookze.nl	19.80	5.2051	15	31
digipluggen.nl	89.60	4.2708	87	102
ttuin.nl	47.33	44.8162	11	111
financewerving.nl	97.50	0.9574	96	99
doelgerichtfysio.nl	33.90	27.6819	0	57
tintelingen.nl	146.67	119.8147	0	280
deputyit.nl	88.20	0.9798	87	89
voedselbank.my-mc.nl	56.90	8.9045	46	65
greenart.nl	48.90	63.8850	1	190
krebworkwear.eu	21.50	21.5000	0	43
sportprijzentiel.nl	8.00	0.0000	8	8
immotix.nl	37.50	9.3622	27	61
bi-kring.nl	193.40	15.8695	188	241

8.2 Reproducing the results

In this section links to source code, database, aggregated data with plot files is linked. It is possible to reproduce the results in this paper from any of the stages in the following subsections, starting with those closest to the final result, and moving down to actually running the setup on a similar, or even completely different set of pages.

8.2.1 raw data and plot files

The data of the various runs is stored in tab separated format for processing with gnuplot (Gnuplot) (or indeed other plotting programs).

http://ethon.nl/universiteit/afstudeerscriptie-bijlagen/plots-and-data.zip

To reproduce the plots seen in this paper with gnuplot move into the sub folder and run the gnu plot program. For example: `Exhaustive-tagAndId`. Now simply load one of the prepared plots, by loading the config from the mailfolder like so:

```
load '../3dscatter-iso.gp'
```

This will open a gnuplot window. For the 3dplots, rotation is possible by pressing and holding the left mouse button.

8.2.2 database dump of captured websites

The websites were crawled from a generated pagelist. This pagelist is included with the java source (see: next section). The result of visiting those pages is stored in a Mariadb

database. MySQL should work equally well, the two are sufficiently compatible for this purpose. This was done so that for new runs the websites need not be visited again. It also protects against changes, or indeed websites going offline. An export of the database containing the pages can be found here:

http://ethon.nl/universiteit/afstudeerscriptie-bijlagen/database-export.sql

When running the java program, results are saved in a separate table, and can by identified with their `run_id`. The SQL file in the link above generates everything required to perform imports, runs and even exporting stats in LaTeX format.

8.2.3 java source

The experiments were done in java. The source code required to (re)perform the experiments is located here:

http://ethon.nl/universiteit/afstudeerscriptie-bijlagen/java-source.zip

In the root of this archive is a pagelist.txt which can be used, or altered to perform experiments. All lines should contain a valid link to an HTML document. Lines beginning with an `#` are ignored and can be used as comments.

To get the program working, simply load the structure as an eclipse project and modify the `Database.java` util class with the correct database location and credentials.

No GUI was implemented, but running specific parts can easily be done from any java environment. To begin, locate the `Entry.java` file and perform one of the operations referenced there:

```
// uncomment one (or more) of the lines below to perform a
// certain stage of the experiment.


// importPages(); // saves the pages from a textfile in a table
// getOracleData(); // gets all previously imported pages
// performRuns(); // performs desired runs. See: for details
// getRunStats(11); //when given a runId prints stats in LaTeX
```

List of Figures

List of Algorithms

List of Tables

References

Lorenzo Blanco, Nilesh Dalvi, and Ashwin Machanavajjhala. Highly efficient algorithms for structural clustering of large websites. *WWW-2011*, 2011.

Valter Crescenzi, Paolo Merialdo, and Paolo Missier. Clustering web pages based on their structure. *Data & Knowledge Engineering*, 54:279–299, 2005.

Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000.

Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, and AndreaPugliesea. Exploiting structural similarity for effective web information extraction. *Data & Knowledge Engineering*, 60:222–234, 2007.

Gnuplot. Gnuplot homepage. URL `http://gnuplot.info`.

Xiaofeng He, Hongyuan Zha, Chris H.Q. Ding, and Horst D. Simon. Web document clustering using hyperlink structures. *Computational Statistics & Data Analysis*, 41: 19–45, 2002.

Zehua Liu, Wee Keong Ng, Ee-Peng Lim, and Feifei Li. Towards building logical views of websites. *Data, Knowledge Engineering*, 49:197–222, 2004.

Merriam-Webster. Merriam-webster dictionary on websites. URL `https://www.merriam-webster.com/dictionary/website`.

Mindconnect. Mindconnect webservices. URL `https://mindconnect.nl`.

Oxford-Dictionary. Oxford dictionary on websites. URL `https://en.oxforddictionaries.com/definition/website`.

W3Schools. Html 5.0 specification, 2014. URL `https://www.w3.org/TR/html5/dom.html`.