APPENDIX C

Semantic Blocks

The semantic-blocks, nodes of a semantic block tree, are achieved by merging semantically correlated render-blocks with the Gestalt laws of grouping [1]. A render-block tree model represents retrieved visual information of a web page by taking a web page's DOM tree as the input instead of parsing the source code. This is because the DOM tree contains all the visual information of a web page [1].

Two major ways, explored by researchers, to represent a web page for visual similarity evaluation are: screen shots (images) and DOM trees. The advantages of the two methods are combined [2], and it proposes a new web page representation method, a block tree. It only extracts visible DOM elements and merges these elements into separate groups according to their semantic meanings.

To construct blocks, separate render objects are merged into semantically related groups based on the Gestalt laws of grouping [2]. Given a web page, *WP*, the render object maps to a visible DOM element e of the DOM tree *DT* of *WP*. The object contains all visible CSS properties of e as the visual features and serves as the merging candidates to build the blocks of the block tree *BT* [2].

The Gestalt laws of grouping explain a human's mechanism for perception. To construct, each block for the block tree, these laws need to be translated into computer compatible rules [1, 3]. The Gestalt laws are described in Appendix B.

Among all the six Gestalt laws (in Appendix B), the first two show us how to extract render objects from the DOM tree, and the remaining four regulate the way of merging the extracted render objects into groups (that is, the blocks in the block tree) by the visual features [2].

The block tree takes the previously merged blocks as tree nodes, and follows the DOM tree's hierarchy to organize these nodes. At the beginning, the first visible DOM element is the "BODY", so the root node of the block tree will be a block that holds it. Next, it follows the bottom up rule. From the root block onwards, all the direct child render objects of a block are evaluated by the Gestalt laws and split into one or more groups. Each of the laws are then applied to create a block. These blocks will maintain their hierarchy in the DOM tree [2].

Block Tree edit distance (TED)

The tree edit distance (TED) is defined as the minimum cost of editing operations ("insert", "delete", and "relabel") when shifting from a block tree to another different block tree [4]. This reflects the structural similarity between two different block trees by mapping node pairs. The pseudo code [4] represented in Algorithm 1 details this step's calculations.

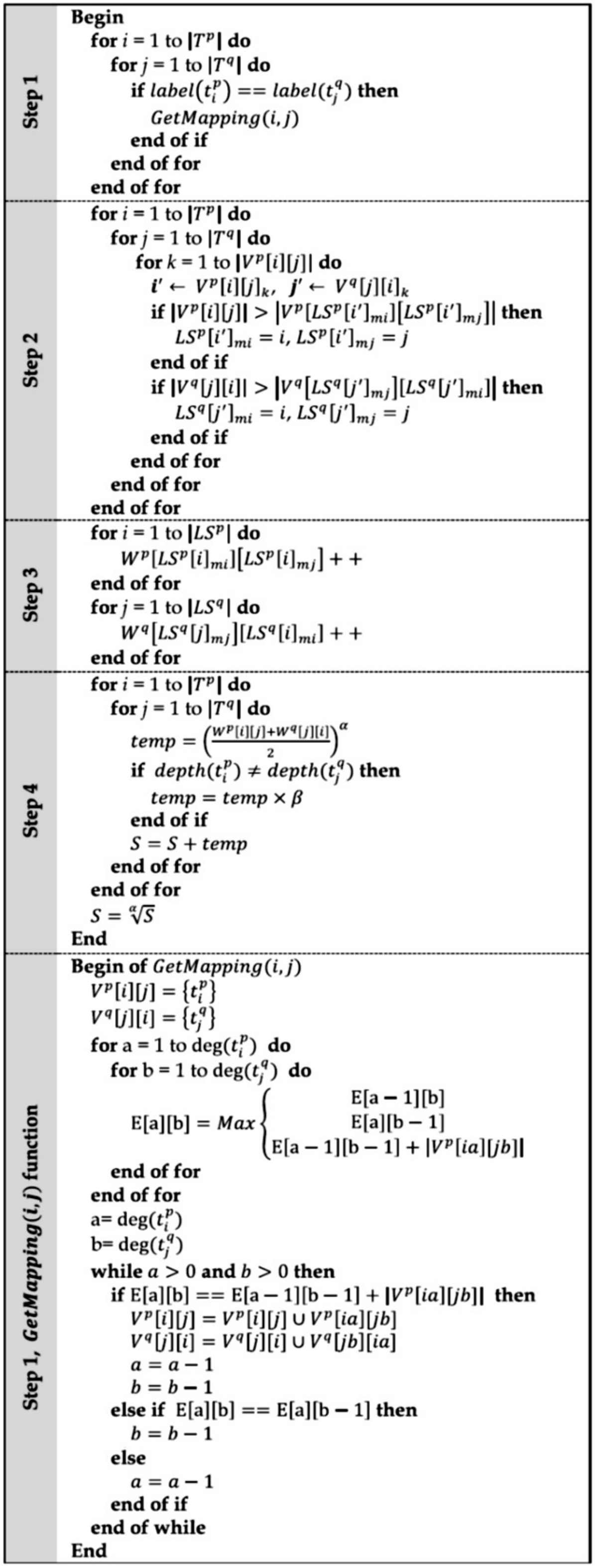
Assume $T_{i,j}^{p}$ represents a subtree (block-tree) of T^p rooted at T_i^p which is mapped to an identical subtree (block-tree) of T^q rooted at $T_{j'}^{q}$, namely $T_{j,i}^{q}$. Accordingly, computing the extended subtree similarity $S(T^p, T^q)$ has four following steps according to [4].

Step 1: *Identify all the mappings*: It finds all the possible mappings and stores two lists of nodes for each mapping, one for each subtree. T^p and T^q are the inputs to this step and V^p and V^q are the outputs. The GetMapping(i,j) function produces two lists of nodes ($V^p[i][j]$ and $V^q[j][i]$) for a mapping. Its objective is to detect the largest possible mapping. T_{ia}^{p} denotes the ath child of

ALGORITHM 1: Subtree Similarity Function Algorithm

Input: Method calls in tree format (T^p, T^q)

Output: Similarity Score



```
Step 1
Begin
  for $i$ = 1 to $|T^p|$ do
    for $j$ = 1 to $|T^q|$ do
      if $label(t_i^p) == label(t_j^q)$ then
        $GetMapping(i,j)$
      end of if
    end of for
  end of for
Step 2
  for $i$ = 1 to $|T^p|$ do
    for $j$ = 1 to $|T^q|$ do
      for $k$ = 1 to $|V^p[i][j]|$ do
        $i' \leftarrow V^p[i][j]_k,\ j' \leftarrow V^q[j][i]_k$
        if $|V^p[i][j]| > |V^p[LS^p[i']_{mi}][LS^p[i']_{mj}]|$ then
          $LS^p[i']_{mi} = i, LS^p[i']_{mj} = j$
        end of if
        if $|V^q[j][i]| > |V^q[LS^q[j']_{mj}][LS^q[j']_{mi}]|$ then
          $LS^q[j']_{mi} = i, LS^q[j']_{mj} = j$
        end of if
      end of for
    end of for
  end of for
Step 3
  for $i$ = 1 to $|LS^p|$ do
    $W^p[LS^p[i]_{mi}][LS^p[i]_{mj}] + +$
  end of for
  for $j$ = 1 to $|LS^q|$ do
    $W^q[LS^q[j]_{mj}][LS^q[i]_{mi}] + +$
  end of for
Step 4
  for $i$ = 1 to $|T^p|$ do
    for $j$ = 1 to $|T^q|$ do
      $temp = \left(\frac{W^p[i][j]+W^q[j][i]}{2}\right)^{\alpha}$
      if $depth(t_i^p) \neq depth(t_j^q)$ then
        $temp = temp \times \beta$
      end of if
      $S = S + temp$
    end of for
  end of for
  $S = \sqrt[\alpha]{S}$
End
Step 1, $GetMapping(i,j)$ function
Begin of $GetMapping(i,j)$
  $V^p[i][j] = \{t_i^p\}$
  $V^q[j][i] = \{t_j^q\}$
  for a = 1 to $\deg(t_i^p)$ do
    for b = 1 to $\deg(t_j^q)$ do
      $E[a][b] = Max \begin{cases} E[a-1][b] \\ E[a][b-1] \\ E[a-1][b-1] + |V^p[ia][jb]| \end{cases}$
    end of for
  end of for
  a= $\deg(t_i^p)$
  b= $\deg(t_j^q)$
  while $a > 0$ and $b > 0$ then
    if $E[a][b] == E[a-1][b-1] + |V^p[ia][jb]|$ then
      $V^p[i][j] = V^p[i][j] \cup V^p[ia][jb]$
      $V^q[j][i] = V^q[j][i] \cup V^q[jb][ia]$
      $a = a - 1$
      $b = b - 1$
    else if $E[a][b] == E[a][b-1]$ then
      $b = b - 1$
    else
      $a = a - 1$
    end of if
  end of while
End
```

the T_i^p node, where $1 \leq a \leq deg(T_i^p)$, and ia represents the index of the ath child of the T_i^p node. E is a matrix which indicates how the children of T_i^p and T_j^q are matched.

Step 2: *Identify each node's largest mapping*: To compute this step, first, assume two arrays, namely LS^p and LS^q, of size $|T^p|$ and $|T^q|$, respectively. $LS^p[i]$ indicates the largest subtree that T_i^p belongs to by the indexes of root nodes of the mapping, denoted by $LS^p[i]_{mi}$ and $LS^p[i]_{mj}$. As indicated in Algorithm 1, filling LS^p and LS^q with appropriate values is the objective of this step.

Step 3: *Compute the weight of each subtree*: This step calculates $W(T_{i,j}^p)$ and $W(T_{j,i}^q)$ for all the subtrees in the mappings. In Algorithm 1, they are denoted by $W^p[i][j]$ and $W^q[j][i]$. It goes through LS^p and increases the weight of a subtree when it is reported as a largest subtree of a node in LS^p. This procedure is repeated for LS^q as well.

Step 4: *Calculate* $S(T^p,T^q)$: It can calculate $S(T^p,T^q)$ according to $S(T^p,T^q) = \alpha\sqrt{\sum_{m_k \in M} \beta_k \times W(m_k)^\alpha}$, where $\alpha, \alpha \geq 1$, is a coefficient to adjust the relation among different sizes of mappings. β_k is a geometrical parameter. Further, m_k is the weight of subtree k in the mapping [4].

REFRENCES

[1] Z. Xu and J. Miller, "A New Webpage Classification Model Based on Visual Information Using Gestalt Laws of Grouping," Cham, 2015, pp. 225-232: Springer International Publishing.

[2] Z. Xu and J. Miller, "Estimating similarity of rich internet pages using visual information," International Journal of Web Engineering and Technology, vol. 12, no. 2, 2017.

[3] H. Stevenson, "Emergence: The Gestalt Approach to Change," Unleashing Executive and Organizational Potential, 2012.

[4] A. Shahbazi and J. Miller, "Extended Subtree: A New Similarity Function for Tree Structured Data," IEEE Transactions on Knowledge & Data Engineering, vol. 26, pp. 864-877, April 2014.