Identifying Online News Articles

Adam Standke

Problem Description

- Because many news based websites include content other than news articles, the automatic identification of news articles is a non-trivial task

- Information retrieval algorithms, heuristics, and machine learning are combined to solve the problem of identifying online new articles

Link Target Identification

- In link target identification, hyperlinks within the web page are broken down and used to determine whether a link points to relevant content (eg., other web pages that contain news articles) or irrelevant content (eg., advertisement based web pages)

Link Target Identification - five specific features

Does a hyperlink have a number/identification within its link structure ?

a. For example: The hyperlink https://abcnews.go.com/story?id=62025268 would be classified as a link that points to a news article

Does a hyperlink have a date within its link structure ?

b. For example: The hyperlink https://www.nytimes.com/2019/04/15/us/politics/trump-ilhan-omar.html would be classified as a link that points to a news article

Does a hyperlink end with a forward slash mark ?

c. For example: The hyperlink https://news.yahoo.com/ would be classified as a link that points to a navigation menu which likely does not contain a news article

Link Target Identification - five specific hyperlink features cont.

Does a hyperlink contain a reserved word such as gallery or video in its link structure?

a. For example: The hyperlink https://video.foxnews.com/v/6026408055001/ would be classified as a link that points to a video rather than a news article

How long is the hyperlink?

b. For example: The length of the hyperlink that points to a news article: https://news.yahoo.com/latest-twister-hits-texas-community-dozen-hurt-201112355.html is lower than the length of the hyperlink that points to an advertisment https://business.comcast.com/4995internet?CMP=BAC-22238190-4304415-238973491-90614167&omndfa=1

Content and/or Visual Feature Analysis

- This form of analysis consists of examining the content and/or visual features within a web page.

- Two different algorithms have been developed to identify news articles
    1. Article Extraction using Maximum Scoring Subsequence
    2. CoreEx

Article Extraction using Maximum Scoring Subsequence

- Three parts to the algorithm:
    1. Bottom-up grouping of text nodes into text segments
    2. Scoring of text segments
    3. Applying Maximum Scoring Subsequence

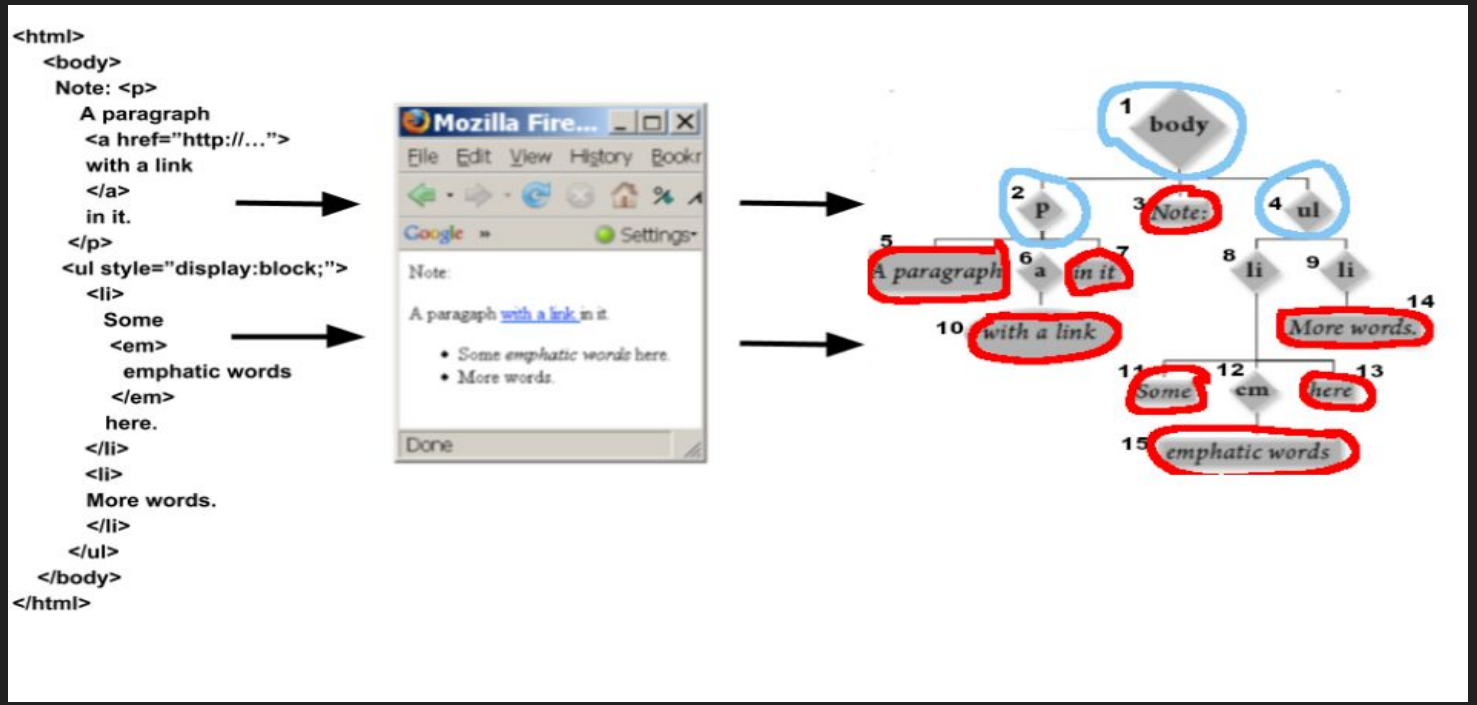
Bottom-up grouping of text leaf nodes

- Text-Leaf Nodes are leaf nodes of the DOM tree that contain text

- The Nearest Line-Break Node for a leaf node in a DOM tree, is the *first line-break node* along the shortest path from it to the root of the tree

- Parts of Algorithm:
  - 1. Generate the sequence of all the leaf nodes in the order they appear in the HTML file
  - 2. Group successive nodes in this sequence with the same nearest line-break node into a text group
  - 3. For each group generated by Step 2, remove any single-tag nodes.

Example of Bottom-up Grouping of text leaf nodes



[6]

Example of Bottom-up Grouping of text leaf nodes

- Result after first step of algorithm:

  Sequence of nodes: (3,5,10,7,11,15,13,14)

- Result after second step of algorithm:

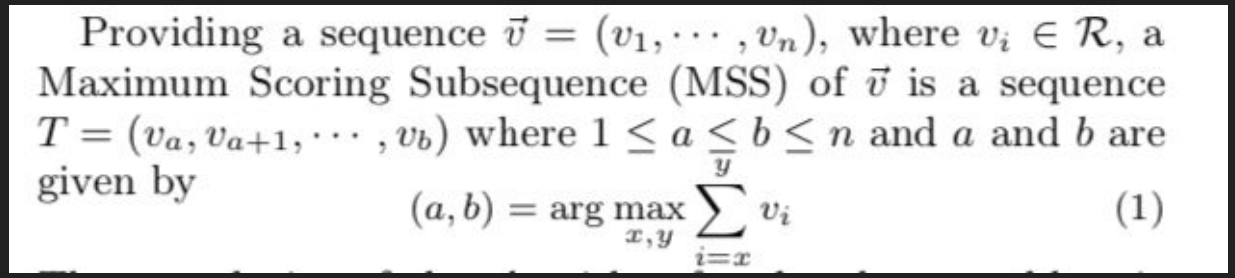
  Grouping of successive nodes: g_1={3}, g_2={5,10,7}, and g_3 = {11,15,13,14}

- Result after third step of algorithm:

  A sequence of text segments where $\vec{s}$ = ({3}, {5,10,7}, {11,15,13,14})

Scoring of text segments

- Given a sequence of text segments the second part of the algorithm seeks to identify the subsequence of text segments which bounds a text body



Providing a sequence $\vec{v} = (v_1, \cdots, v_n)$, where $v_i \in \mathcal{R}$, a Maximum Scoring Subsequence (MSS) of $\vec{v}$ is a sequence $T = (v_a, v_{a+1}, \cdots, v_b)$ where $1 \leq a \underset{y}{\leq} b \leq n$ and a and b are given by

$$(a, b) = \arg\max_{x,y} \sum_{i=x} v_i \quad (1)$$

[3]

Scoring of text segments Cont.

- Achieved by mapping text segments to numeric values based on the following relationship:

Given a sequence of text segments $\vec{s} = (s_1, \cdots, s_n)$ (s_i is a text segment), we solve the MSS problem in Equation (1) for the value sequence $\vec{v} = (v_1, \cdots, v_n)$, where

$$v_i = F(s_i) \cdot StringLength(s_i), \quad (2)$$

[3]

- Where the function *F* is defined to be:

$$F(s) = \begin{cases} 1 & p_{size} \geq c_1, p_{color} \geq c_2, p_{link} \leq c_3 \\ -1 & otherwise, \end{cases}$$

[3]

Scoring of text segments cont.

p_{size} = percentage of text in text segment that is formatted with the most commonly font size

p_{color} = percentage of text in text segment that is formatted with the most commonly-used font color

p_{link} = percentage of text in text segment that is contained within a hyperlink

c_1, c_2 , and c_3 are three constant values which have been predetermined to be 0.7, 0.2, and 0.5

Example: Scoring of text segments - segment one

{ Note: }

$$v_1 = F(s_1) \cdot StringLength(s_1)$$

$$StringLength(s_1) = 5$$

$$p_{size} = \frac{5}{5} = 1 \geq 0.7$$

$$p_{color} = \frac{5}{5} = 1 \geq 0.2$$

$$p_{link} = \frac{0}{5} = 0 \leq 0.5$$

$$v_1 = 5$$

Example: Scoring of text segments - segment two

{ A paragraph, with a link, in it }

$$v_2 = f(s_2) \cdot StringLength(s_2)$$

$$StringLength(s_2) = 27$$

$$p_{size} = \frac{27}{27} = 1 \geq 0.7$$

$$p_{color} = \frac{27}{27} = 1 \geq 0.2$$

$$p_{link} = \frac{11}{27} = 0.4 \leq 0.5$$

$$v_2 = 27$$

Example: Scoring of text segments - segment three

{ some, emphatic words, here, More words }

$v_3 = f(s_3) \cdot StringLength(s_3)$

$StringLength(s_3) = 33$

$$p_{size} = \frac{33}{33} = 1 \geq 0.7$$

$$p_{color} = \frac{33}{33} = 1 \geq 0.2$$

$$p_{link} = \frac{0}{33} = 0 \leq 0.5$$

$v_3 = 33$

Maximum Scoring Subsequence Algorithm

- MSS maintains a certain ordered list of disjoint subsequences $I_1, I_2, \cdots, I_{k-1}$

- For each subsequence, MSS records the cumulative total L_j, which equals the score up to *but not including the leftmost score* I_j

- Also, MSS records the cumulative total R_j, which equals the score up to and *including the rightmost score* I_j

Maximum Scoring Subsequence Algorithm Cont.

1. The list is searched from right to left for the maximum value of j satisfying $L_j < L_k$.
2. If there is no such j, then add I_k to the end of the list.
3. If there is such a j, and $R_j \geq R_k$, then add I_k to the end of the list.
4. Otherwise (i.e., there is such a j, but $R_j < R_k$), extend the subsequence I_k to the left to encompass everything up to and including the leftmost score in I_j. Delete subsequences $I_j, I_{j+1}, \ldots, I_{k-1}$ from the list (none of them is maximal) and reconsider the newly extended subsequence I_k (now renumbered I_j) as in step 1.

After the end of the input is reached, all subsequences remaining on the list are maximal; output them.

[8]

Example: Maximum Scoring Subsequence

$$\vec{V} = (5, 27, 33)$$

Finding Maximum Scoring Subsequence: T

I_1 = (5) and (L_1, R_1)=(0,5)

I_2 = (27) and (L_2, R_2)=(5,32)

Since $L_1 < L_2$ and $R_1 < R_2$ Delete I_1, I_2. Now I_1= (5,27) and (L_1, R_1) =(0,32)

I_2 = (33) and (L_2, R_2) =(32,65)

Since $L_1 < L_2$ and $R_1 < R_2$ Delete I_1, I_2. Now I_1 =(5,27,33) and (L_1, R_1)=(0,65)

T=(5,27,33) which corresponds to text segments $\vec{S}$= ({3},{5,10,7}, {11,15,13,14})

CoreEx

- CoreEx is a simple heuristic based algorithm that extracts the main article from online news Web pages

- CoreEx analyzes the amount of text and number of links in every node in the DOM tree and uses a heuristic measure to determine the node (or set of nodes) most likely to contain the main content

- Two parts to the algorithm:
  - 1. Determining the text-to-link ratio for terminal nodes and non-terminal nodes
  - 2. Scoring of non-terminal nodes

CorexEx

- *textCnt*: holds the number of words contained in one node
- *linkCnt*: holds the number of links in or below the node
- S: set of nodes that contains the main content
- *setTextCnt*: holds the sum of textCnt for nodes added to set S
- *setLinkCnt*: holds the sum of linkCnt for nodes added to set S
- *Threshold*: determines whether a node's child should be added to set S
    - Empirically determined to be 0.9

CoreEx - Terminal nodes

-if T is a text node **then**
    textCnt(T) = word count of text in T
    linkCnt(T) = 0
-else if T is a link node
    textCnt(T) = 1
    linkCnt(T) = 1
-else
    textCnt(T) = 0
    linkCnt(T) = 0
end if

[6]

CoreEx - Non-terminal nodes

For a non-terminal node N:

textCnt(N) = 0 and *linkCnt(N)* = 0
S(N) is an empty set
setTextCnt(N) = 0 and *setLinkCnt(N)* = 0
for each child of N **do**
textCnt(N) = *textCnt(N)* + *textCnt(child)*
linkCnt(N) = *linkCnt(N)* + *linkCnt(child)*
childRatio = $\frac{textCnt(child) - linkCnt(child)}{textCnt(child)}$
if *childRatio* > *Threshold* **then**
add child to S(N)
setTextCnt(N)= *setTextCnt(N)*+*textCnt(child)*
setLinkCnt(N)= *setLinkCnt(N)*+*linkCnt(child)*
end if
end for
Store S(N), *textCnt(N)*, *linkCnt(N)*, *setTextCnt(N)*, and *setLinkCnt(N)*

[6]

CorexEx - Scoring

Scoring function:

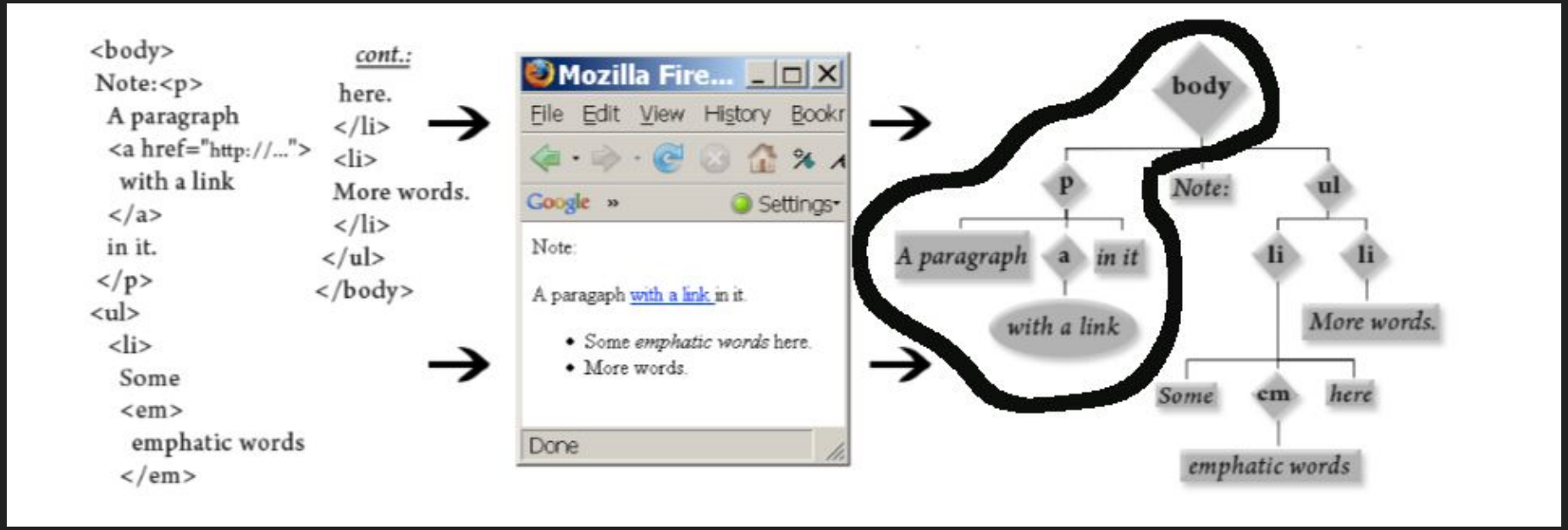
$$weight_{ratio} \times \frac{setTextCnt(N) - setLinkCnt(N)}{setTextCnt(N)} + weight_{text} \times \frac{setTextCnt(N)}{page_{text}}$$

[6]

CoreEx

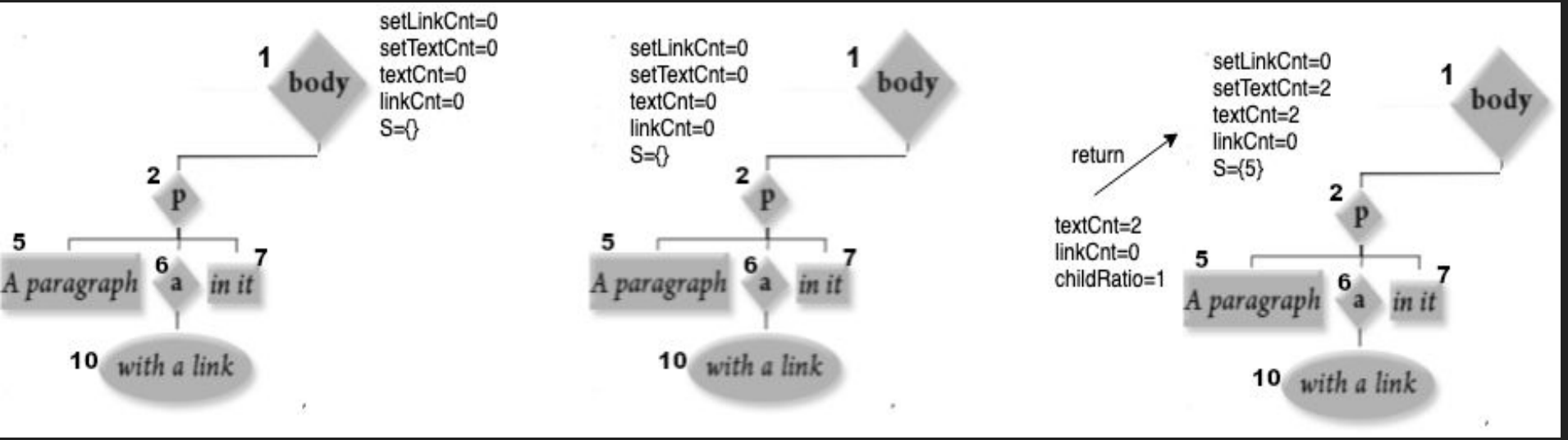
- $weight_{ratio}$ and $weight_{text}$ are weights assigned to both parts of the scoring function
- Both weights have been empirically determined to be: 0.99 for $weight_{ratio}$ and 0.01 for $weight_{text}$
- $page_{text}$ is the total text the web page contains

Example: CorexEx - Left Sub-tree Traversal

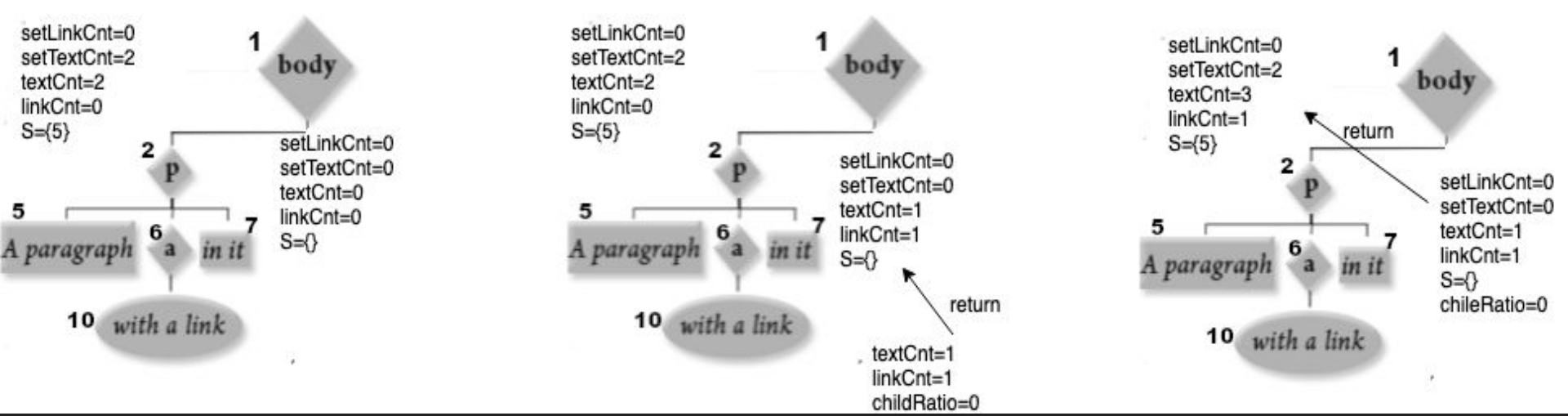


[6]

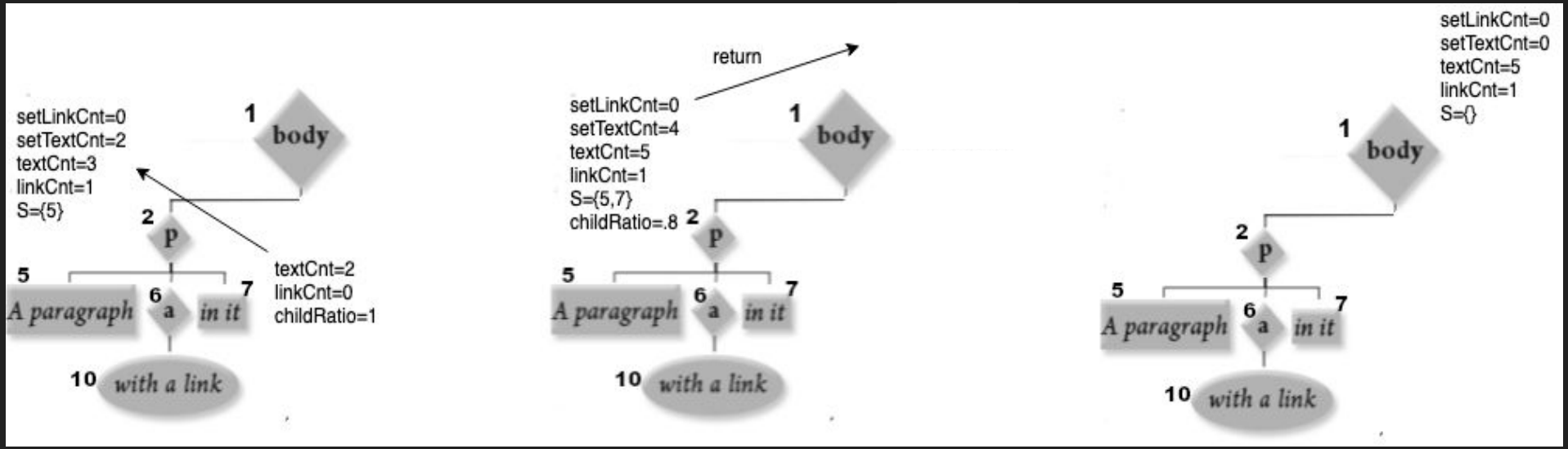
Example: CorexEx - Left Sub-tree Traversal

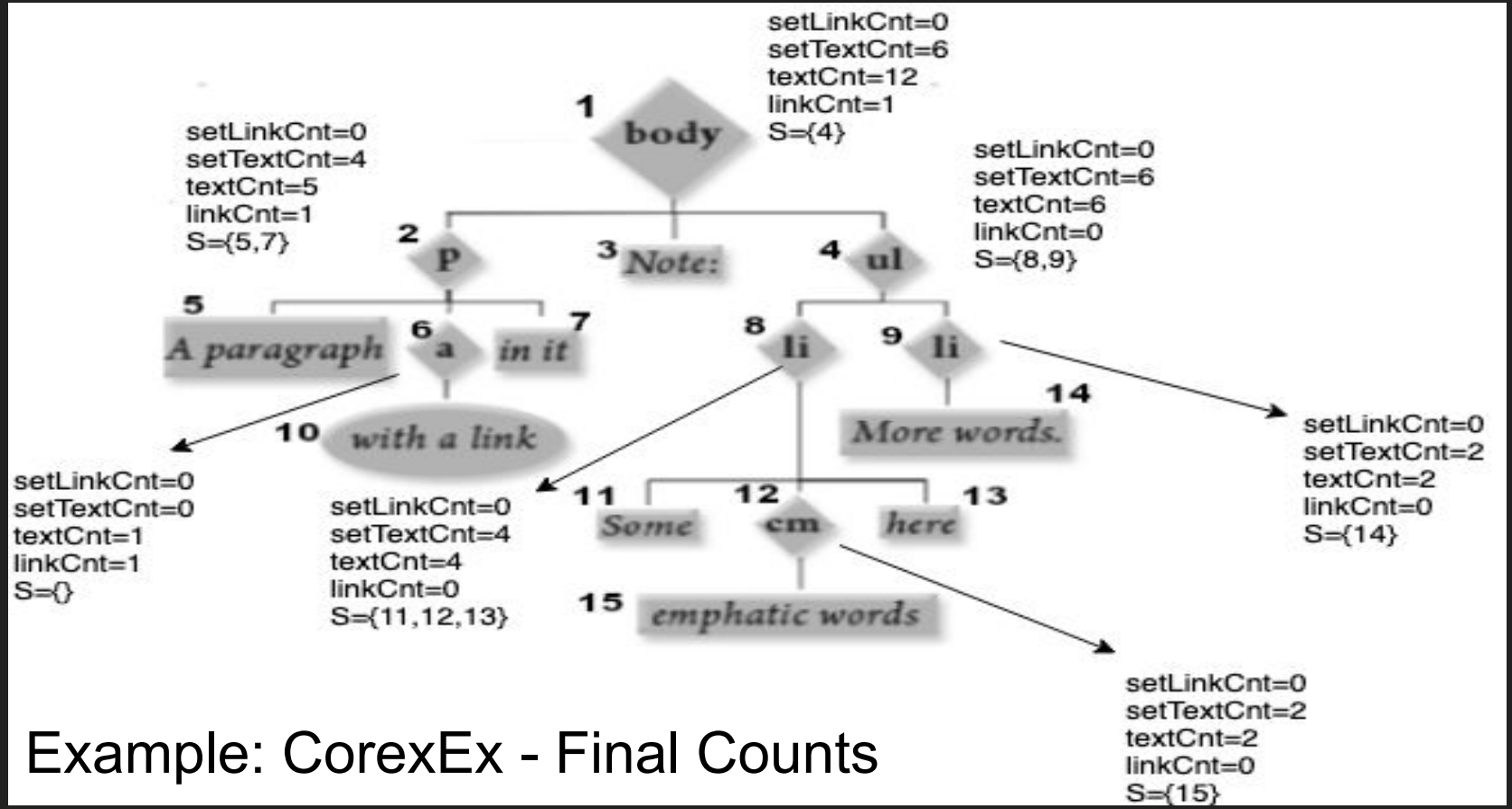


Example: CorexEx - Left Sub-tree Traversal



Example: CorexEx - Left Sub-tree Traversal





Example: CorexEx - Final Counts

Example: CorexEx - Scoring

- $score(node6) = 0.99 \times \frac{0}{0} + 0.01 \times \frac{0}{12} = 0$
- $score(node2) = 0.99 \times \frac{4-0}{4} + 0.01 \times \frac{4}{12} = .9933$
- $score(node12) = 0.99 \times \frac{2-0}{2} + 0.01 \times \frac{2}{12} = .9916$
- $score(node8) = 0.99 \times \frac{4-0}{4} + 0.01 \times \frac{4}{12} = .9933$
- $score(node9) = 0.99 \times \frac{2-0}{2} + 0.01 \times \frac{2}{12} = .9916$
- $score(node4) = 0.99 \times \frac{6-0}{6} + 0.01 \times \frac{6}{12} = .9950$
- $score(node1) = 0.99 \times \frac{6-0}{6} + 0.01 \times \frac{6}{12} = .9950$

- How CoreEx Breaks Ties:
  - If two nodes reach identical scores, the node higher in the DOM tree is selected. If two siblings are tied, a random choice is made among them.
- Main Content Node: <body>
- Node(s) that contain the Main Content:

C4.5 Machine Learning Algorithm

- C4.5 generates a classification model in the form of a decision tree, where each leaf indicates a particular class and each non-leaf represents a decision node that specifies some test to be carried out on an attribute

- C4.5 relies on a criteria called gain to effectively partition the training cases into subsets of cases until a *single-class collection of cases* is achieved

C4.5 Machine Learning - decision trees

- C4.5's gain criteria is used to split a set of training cases into subsets

$$gain(X) = info(T) - info_X(T), \text{ where } \begin{cases} info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} \times log_2(\frac{freq(C_j, T)}{|T|}) \\ info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i) \end{cases},$$

[7]

C4.5 Machine Learning - decision trees

- $gain(X)$ = measures the information that is gained by partitioning a set of training cases by a given attribute
- $info(T)$ = calculates the average amount of information needed to identify the class of a training case
- $info_X(T)$ = represents the expected information after the set of training cases has been partitioned by an attribute
- C_j = represents a class
- T = represents a set of training cases
- X = represents an attribute

Example: Decision tree

T	Value of Highest Scoring Subsequence (MSS)	HTML Tag of main content node (CorexEx)	Class
1	15	<body>	Not Article
2	65	<body>	Article
3	70		Article
4	80		Article
5	50	<iframe>	Not Article

Example: Decision tree

$$info(T) = -\frac{3}{5} \times log_2(\frac{3}{5}) - \frac{2}{5} \times log_2(\frac{2}{5}) = 0.44 + 0.53 = 0.97$$

- Sort continuous values for MSS:

15	50	65	70	80
-	-	+	+	+

- Calculate midpoint:

$$\frac{50 + 65}{2} = 57.5$$

- Pick Threshold(s):
  57.4

Example: Decision tree

$$Gain(MSS) = 0.97 - \frac{3}{5} \times info(MSS_{>57.4}) - \frac{2}{5} \times info(MSS_{\leq 57.4})$$

$$info(MSS_{>57.4}) = -\frac{3}{3} \times log_2(\frac{3}{3}) - \frac{0}{3} \times log_2(\frac{0}{3}) = 0$$

$$info(MSS_{\leq 57.4}) = -\frac{0}{2} \times log_2(\frac{0}{2}) - \frac{2}{2} \times log_2(\frac{2}{2}) = 0$$

$$0.97 - \frac{3}{5}(0) - \frac{2}{5}(0) = 0.97$$

Highest MSS Value:

[3+, 0-] for MSS Value > 57.4

[0+, 2-] for MSS Value <= 57.4

Example: Decision tree

$$Gain(CoreEx) = 0.97 - \frac{2}{5} \times info(<body>) - \frac{2}{5} \times info(<div>) - \frac{1}{5} \times info(<iframe>)$$

$$info(<body>) = -\frac{1}{2} \times log_2(\frac{1}{2}) - \frac{1}{2} \times log_2(\frac{1}{2}) = 1$$

$$info(<div>) = -\frac{2}{2} \times log_2(\frac{2}{2}) - \frac{0}{2} \times log_2(\frac{0}{2}) = 0$$

$$info(<iframe>) = -\frac{0}{1} \times log_2(\frac{0}{1}) - \frac{1}{1} \times log_2(\frac{1}{1}) = 0$$

$$= 0.97 - \frac{2}{5}(1) - \frac{2}{5}(0) - \frac{1}{5}(0) = 0.57$$

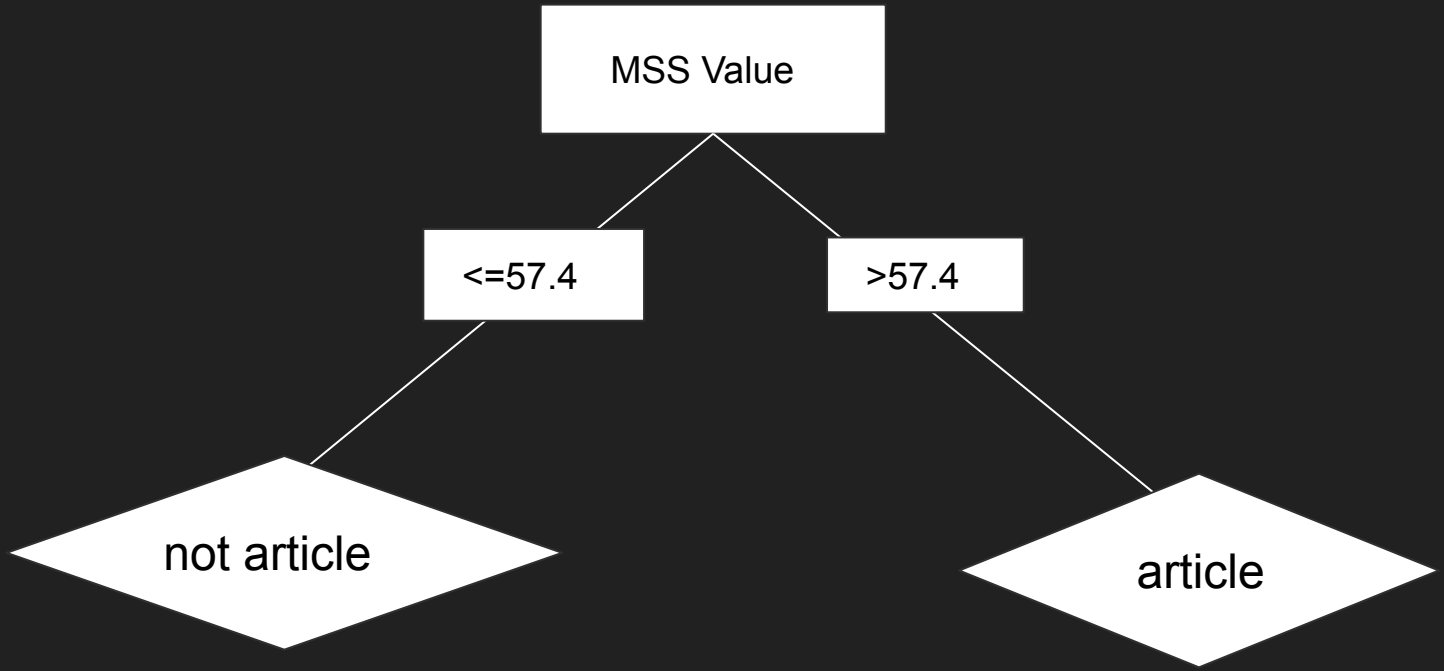
CoreEx Main Content Tag:

[1+, 1-] for <body> tag

[2+, 0] for tag

[0, 1-] for <iframe> tag

Example: Decision tree



Primary Objective

- Primary Objective: The primary objective of this project is to assess the predictability of a news article existing within a web page
- Solution: Decision trees using the C4.5 machine learning algorithm will be constructed using features extracted from link-target identification, article extraction using maximum scoring subsequence, and CoreEx
- Limitation: Summer Term A of 2019

Hypothesis

- Adding the five features from link-target identification will improve overall performance of identifying news articles than just using the two article extraction algorithms (ie., article extraction using MSS and CoreEx)

Experiment Design

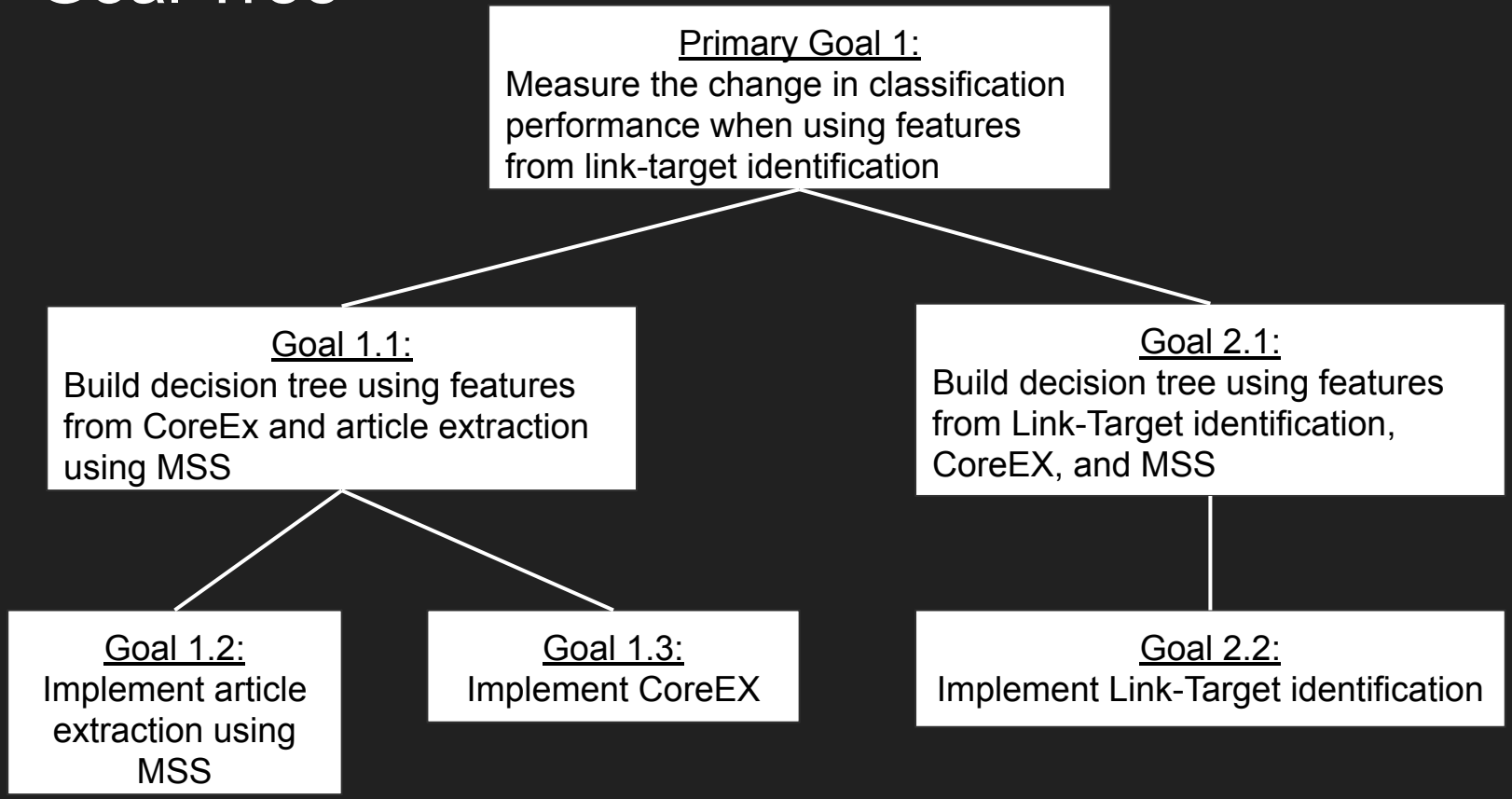
- Data:
  - The dataset will contain a total of 330 fully rendered web pages from the following popular news websites: CNN, New York Times, Washington Post, Yahoo News, Fox News, BBC, The Verge, Huffington Post, ABC News, and CBS News.

  - Each web page will be manually labeled, with approximately 50% being labeled as an article and approximately 50% being labeled as not an article.

Experiment Design Cont.

- Google’s headless browser will be used to download 330 fully rendered DOM web pages
- A DOM tree parser will be constructed using the JSoup library to implement the three information retrieval systems
- The C4.5 machine learning program will be used to construct decision trees
  - F1 scores will be used to judge overall performance of decision trees

Goal Tree


Primary Goal 1:
Measure the change in classification performance when using features from link-target identification
Goal 1.1:
Build decision tree using features from CoreEx and article extraction using MSS
Goal 2.1:
Build decision tree using features from Link-Target identification, CoreEX, and MSS
Goal 1.2:
Implement article extraction using MSS
Goal 1.3:
Implement CoreEX
Goal 2.2:
Implement Link-Target identification

Block Design - Experiment Factors

Factors	Values
Link-Target Identification	{ # of links that contain a number in the name of the link, # of links that contain a date, # of links that contain a reserved word, # of links that end with a forward slash mark, # of links that are smaller than predetermined threshold }
Article Extraction using Maximum Scoring Subsequence (MSS)	{ an integer that represents the highest scoring subsequence }
CoreEx	{ the main content node's HTML tag, the HTML tag with the highest frequency count in the set S, an integer that represents the score for the main content node }
Sample	cross-validation will randomly split up 330 web pages into 10 groups of 33 web pages. Nine of the ten groups will be used to train the model and one group will be used to test the model for each round. This process will be repeated for 20 trials.

Block Design - Experiment

	Decision tree built from CoreEx and Article Extraction using MSS	Decision tree built from Link-Target Id, CoreEx and Article Extraction using MSS
297 training pages/33 test pages	X	X

Citations

[1]Fan, Jian, et al. “Article Clipper - A System for Web Article Extraction.” *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 11*, 2011, doi:10.1145/2020408.2020525.

[2]Ferreira, Rodolfo, et al. “Appling Link Target Identification and Content Extraction to Improve Web News Summarization.” *Proceedings of the 2016 ACM Symposium on Document Engineering - DocEng 16*, Sept. 2016, pp. 197–200., doi:10.1145/2960811.2967158.

[3]Luo, Ping, et al. “Web Article Extraction for Web Printing: a DOM+Visual Based Approach.” *Proceedings of the 9th ACM Symposium on Document Engineering - DocEng 09*, Sept. 2009, doi:10.1145/1600193.1600208.

[4]Mitchell, Tom M. *Machine Learning*. McGraw Hill, 2017.

[5]Pasternack, Jeff, and Dan Roth. “Extracting Article Text from the Web with Maximum Subsequence Segmentation.” *Proceedings of the 18th International Conference on World Wide Web - WWW 09*, 2009, doi:10.1145/1526709.1526840.

[6]Prasad, Jyotika, and Andreas Paepcke. “Coreex.” *Proceeding of the 17th ACM Conference on Information and Knowledge Mining - CIKM 08*, Oct. 2008, doi:10.1145/1458082.1458295.

[7]Quinlan, John Ross. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 2006.

[8]Ruzzo, Walter and Martin Tompa. “A Linear Time Algorithm for Finding All Maximal Scoring Subsequences.” *Proceeding of the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 234–241.