**Adam Standke**
**CSC498**

# A DOCUMENT CLASSIFICATION PROBLEM OF IDENTIFYING A NEWS ARTICLE WITHIN A WEB PAGE?

**Abstract:**

This paper focuses on the document classification problem of identifying whether a web page contains an online news article. Contained within this paper is background information about how machine learning algorithms when used in combination with information extraction algorithms/techniques can identify online news articles. Additionally, this paper contains an experimental design along with a proposed solution description for identifying online news articles.

## 1. Introduction

Currently, many news based websites include content other than news articles. Often times, a news web page contains no articles at all but other forms of content such as videos, images, and/or slides. Consequently, identifying news articles by automatic means is a nontrivial task. Simple heuristics that may have worked to identify news articles in the early days of the web have become outdated in today's web landscape. In order to effectively solve the problem of identifying news articles a multidisciplinary approach must be taken. This project takes such an approach by combining fields such as information retrieval, data mining, and machine learning to solve this classification task.

In doing so, this project will use the following definition of an article to classify online news articles: an article is a contiguous, coherent work of prose on a single topic or multiple closely related topics that alone comprises the main informational content of the page[5]. For the news domain, in addition to the general requirements for an article, a news article must be a story or report at least two paragraphs and eight total sentences in length[5].

## 2. Literature Review

Topics that deal with information retrieval, data mining, and machine learning will be subsequently discussed. These topics include the following: link target identification, analysing the content and/or visual features of web pages, and the C4.5 machine learning algorithm.

### 2.1 Link Target Identification

The link analysis field studies the relationship between web pages[1]. An area of research that really flourished after the proposal of the page rank algorithm[1]. The page rank algorithm assesses the importance of a web page according to the number of external links that point to it[1]. However, the page rank algorithm does not classify the content of web pages based on link structure[1]. In link target identification several features are extracted from the root link within a web page[1]. The features extracted are then used to determine whether the root link points to relevant content (eg., other news web pages that contain articles) or irrelevant content (eg., advertisement web pages)[1].

To create such a classification six specific features based on the link structure are used[1]. The first feature checks to see if the link contains a number in the name of the link[1]. Links related to news usually have an id in their structure[1]. For example, a hyperlink such as https://abcnews.go.com/story?id=62025268 would be classified as a news article since its link

contains a number and an id variable. The second feature checks to see if the link contains a date[1]. In general it is more likely that a news article contains a date in its link, more so than advertising and/or navigation menus[1]. The third feature judges whether a link contains a news article by measuring a link's length; generally, longer link structures usually represent news web pages, since the title of a news article is oftentimes contained within the link[1].

The fourth feature leverages the fact that navigation menus usually end with a slash mark[1]. For example, a hyperlink such as https://news.yahoo.com/ would be classified as a navigation type object since the link ends with a forward slash mark. For the fifth feature a number of reserved words are used to filter out multimedia news[1]. The words gallery, video, image, photo, slideshow, episode, and player are a list of reserved words that the fifth feature takes into account to determine whether the web page pointed by such a link represents irrelevant content[1]. The last feature counts the number of slashes in the link address which indicates the location of a target web page within a file system[1].

**2.2 Analyzing the Content and/or Visual Features of Web Pages**

This form of analysis consists of examining the content and/or visual features within a web page. By transforming the content of a web page into a document object model tree (ie., DOM tree), the content of a web page can be efficiently extracted and traversed by different information retrieval algorithms. One such algorithm that uses a heuristic approach to extraction of online news articles is CoreEx.

**2.2.1 CoreEx**

CoreEx analyzes the amount of text and number of links of every node in the DOM tree and then applies a heuristic to determine the node (or set of nodes) that most likely contain the main content[3]. For every node in the DOM tree, CoreEx maintains a set of nodes $S$ which stores a subset of the node's children and calculates four counts: textCnt, linkCnt, setTextCnt, and setLinkCnt[3]. TextCnt holds the number of words while linkCnt holds the number of links contained in one node[3]. To keep the score normalized between 0 and 1 a link is counted as one word of text[6]. SetTextCnt and setLinkCnt are the sum of textCnt and linkCnt[3]. CoreEx sets these two values only if a child-node's text-to-link ratio is above a certain threshold[3].

$$
\begin{aligned}
&For\ a\ terminal\ node\ T:\\
&\quad -\ If\ T\ is\ a\ text\ node\\
&\qquad textCnt(T)\ =\ word\ count\ of\ text\ in\ T\\
&\qquad linkCnt(T) = 0\\
&\quad -\ If\ T\ is\ a\ link\ node\\
&\qquad textCnt(T)\ = 1\\
&\qquad linkCnt(T) =\ 1\\
&\quad -\ In\ all\ other\ cases\\
&\qquad textCnt(T)\ = 0\\
&\qquad linkCnt(T) =\ 0
\end{aligned}
$$

**Figure 1**

$$
\begin{aligned}
&For\ a\ non-terminal\ node\ N:\\
&\quad textCnt(N)\ = 0\ and\ linkCnt(N) = 0\\
&\quad S(N)\ is\ an\ empty\ set\\
&\quad setTextCnt(N)\ = 0\ and\ setLinkCnt(N)\ = 0
\end{aligned}
$$

**for** *each child of N* **do**
    $textCnt(N) = textCnt(N) + textCnt(child)$
    $linkCnt(N) = linkCnt(N) + linkCnt(child)$
    $childRatio = \frac{textCnt - linkCnt}{textCnt}$
    **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* $setLinkCnt(N)$

**Figure 2**

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

**Figure 3**

The CoreEx algorithm can be split up into two parts; namely, nodes that are terminal and nodes that are non-terminal[3]. Both parts of the algorithm are shown in figures 1 and 2[3]. Figure 1 details CoreEx's handling of terminal nodes (ie., T). If T is a text node (ie., only contains text) its textCnt is determined by how many words are contained in T[3]. In all other cases where T is not a text node a value of 1 or 0 is assigned to it[3]. Figure 2 details CoreEx's handling of non-terminal nodes (ie., N). Initially, N's textCnt, linkCnt, and the set *S* are empty[3]. Then for each child of N, N's textCnt and linkCnt are updated[3]. If a child of N satisfies a pre-set threshold (which has been empirically determined to be 0.9) the child will be added to the set *S* and setTextCnt and setLinkCnt will be updated for N[3]. This process repeats until all non-terminal nodes have been evaluated[3].

Once the counts are known for each non-terminal node, the nodes are scored based on their values for setTextCnt and setLinkCnt[3]. Figure 3 details CoreEx's scoring function for non-terminal nodes (ie., N). $Page_{text}$ is the total amount of text that the webpage contains[3]. $Weight_{ratio}$ and $weight_{text}$ are weights assigned to the function which have been experimentally determined to be 0.99 for $weight_{ratio}$ and 0.01 for $weight_{text}$ [3]. Using this scoring function the DOM node with the highest score is picked as the main content node and the corresponding set of nodes in the set *S* contains the news article[3].

**2.3 The C4.5 Machine Learning Algorithm**

C4.5 is a machine learning algorithm that constructs classification models[4]. The classification task consists of assigning things to categories or classes as determined by their properties[4]. In doing so, C4.5 constructs classification models through induction by generalizing from specific examples[4]. C4.5 is a descendant of an earlier machine learning algorithm called ID3, however, C4.5 provides additional capabilities[4]. In its simplest form, 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[4]. To construct a decision tree C4.5 relies on partitioning a set of training cases T into subsets of cases until a single-class collection of cases is achieved[4].

$$info(T) = -\sum_{j=1}^{k} \frac{freq(C_j,T)}{|T|} \times log_2\left(\frac{freq(C_j, T)}{|T|}\right) bits$$

**Figure 4**

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)\ bits$$

**Figure 5**

$$gain(X) = info(T) - info_X(T)$$

**Figure 6**

Similar to the ID3 algorithm, C4.5 relies on a criteria called gain to effectively partition the training cases[4]. The information theory that underpins the gain criterion relies on the notion that the information conveyed by a message depends on its probability and can be measured in bits[4]. Figures 4, 5, and 6 detail the gain criteria.

Figure 4 shows the function that is used to compute the (im)purity of an arbitrary collection of training examples[2]. This function when applied to a set of training cases measures the average amount of information needed to identify the class of a training case in T[4]. The quantity $freq(C_j, T)$ calculates the number of training cases in T that belong to class $C_j$ [4]. The other function, shown by figure 5 represents the information gained after partitioning a set of training cases by a certain attribute[4]. Each $T_i$ is a subset of T for which attribute X has the value i[2]. Together, both these functions make up the gain criterion as shown by figure 6[4]. The gain criterion selects attributes that maximize this gain criterion[4]. In doing so, the set of training cases are partitioned into a corresponding decision tree[4].

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times log_2\left(\frac{|T_i|}{|T|}\right)$$

**Figure 7**

$$gain\ ratio(X) = gain(X)/split\ info(X)$$

**Figure 8**

C4.5 improves upon the gain criterion by introducing an additional gain criteria called gain ratio[4]. The gain criterion is inherently biased, since it favors attributes with many different outcomes[4]. The bias inherent in the gain criterion is rectified by C4.5 through a normalization procedure in which the apparent gain attributable to an attribute with many outcomes is adjusted[4]. Figures 7 and 8 show C4.5's calculation of the gain ratio criterion[4]. Figure 7 calculates the split information for an attribute X. Where $T_1$ through $T_n$ represent n subsets after the training set T is partitioned by n different values of attribute X[2]. After doing so, figure 8 expresses the proportion of information generated by the attribute that is useful for classification[4].

In addition to C4.5's gain ratio, two additional features inherent to C4.5 improve its classification performance; namely, retrospective pruning and biased windowing[4]. C4.5 modifies the recursive partitioning of the training cases by removing, retrospectively, some of the original structure of the original decision tree[4]. Generally, this is done by replacing subtrees with leaves by comparing the total number of predicted errors[4]. Even though C4.5 cannot

determine exactly the probability of a predicted error at a given leaf, it is able to bound an upper limit on this probability based on a given confidence level[4]. C4.5 then equates the predicted error rate with this upper limit to then compute the total number of predicted errors[4]. By removing parts of the tree that do not contribute to the accuracy of unseen cases, a pruned tree generally has a lower error rate on unseen cases than an un-pruned tree[4].

C4.5 also supports a mechanism called windowing, in which C4.5 biases the selection of training cases (called the window) from which the initial decision tree is constructed from[4]. C4.5 biases the selection by making sure the distribution of classes in the initial window is as uniform as possible[4]. This type of windowing not only supports faster construction of decision trees, but also more accurate trees[4]. Windowing allows multiple trees to be grown and the tree with the lowest predicted error rate is chosen[4].

**3. Primary Objective**

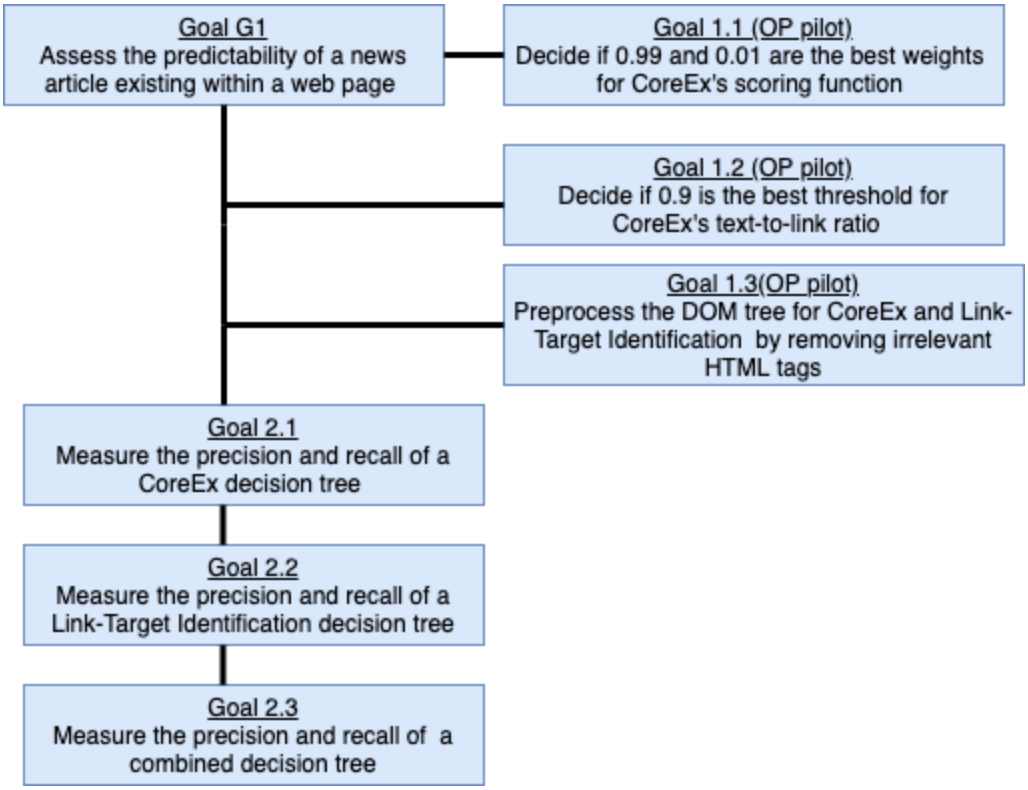
The primary objective of this project is to assess the predictability of a news article existing within a web page. By constructing decision trees using the C4.5 machine learning algorithm from features extracted from link-target identification and CoreEx a simple binary classification system will be constructed that will hopefully be able to accurately classify online news articles.

**4. Solution Description**

To achieve the primary objective several tools will be used for this project. First, Google's headless browser called Headless Chrome will be used to run Google Chrome in a headless environment. This type of environment will allow fully rendered web pages to be saved in the same layout and format as they would appear on Google Chrome. Once all of the web pages have been downloaded, an HTML tree parser will be constructed using the Java library called JSoup. JSoup is an application programming interface that allows HTML files to be parsed and manipulated using DOM traversal and/or CSS selectors.Through JSoup, the Java language, and the Eclipse Integrated Development Environment (IDE), link-target identification and CoreEx will be implemented and applied to each saved web page. Lastly, the eighth release of the C4.5 machine learning program will be used to construct decision trees from the information gathered from link-target identification and CoreEx.

**5. Hypotheses with Goal Tree**

This project proposes two hypotheses. Hypothesis one consists of the following declaration: link-target identification will perform better than CoreEx at identifying news articles. On the other hand, the null hypothesis would be that link-target identification will not perform better than CoreEx at identifying news articles. The second hypothesis makes the following claim: the decision tree constructed from features extracted from link-target identification and CoreEx will have the best classification performance of identifying news articles. Alternatively, the null hypothesis would be that the combination of link-target identification along with CoreEx would not have the best classification performance at identifying news articles.




**6. Experimental Design**

The dataset will contain a total of 330 fully rendered web pages. Using the technique of cross-validation, 330 web pages will be randomly shuffled and split up into 10 groups of 33 web pages. Of the ten groups, nine will be used to train the model and one will be used to test the model for each round. After ten rounds of training and testing, the accuracy score for a given trial will be the average of the ten accuracy scores obtained during cross-validation.

| Factors | Values |
|---|---|
| **Link-Target Identification** | { does the root link contain a number/Id in the name of the link, does the root link contain a date, does the root link contain a reserved word, does the root link end with a forward slash mark, the length of the root link, the number of slashes within the root link} |
| **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. |

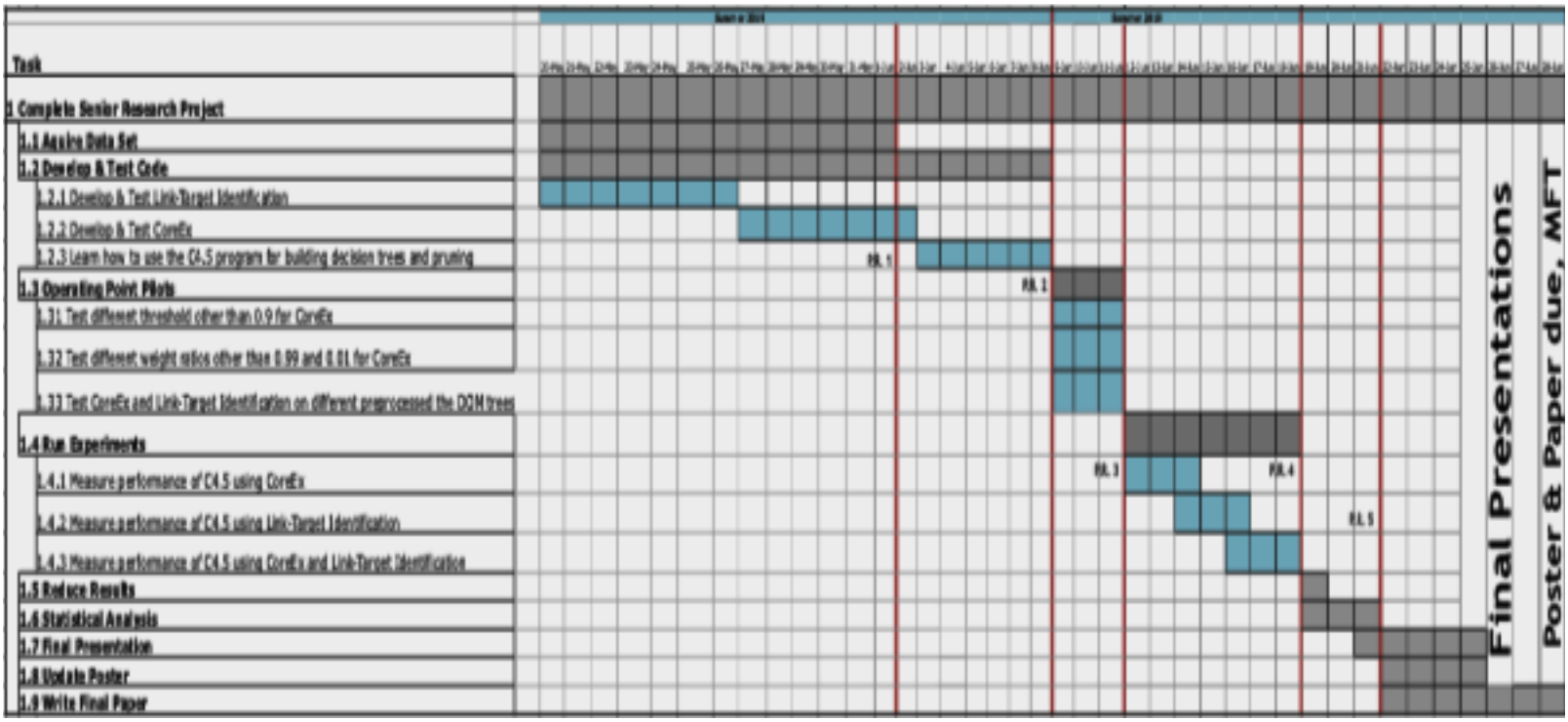
**Table 1**

| | Decision tree built from CoreEx | Decision tree built from Link-Target Identification | Decision tree built from Link-Target Identification and CoreEx |
|---|---|---|---|
| **297 training pages/33 test pages** | **X** | **X** | **X** |

**Table 2**

The web pages that make up the dataset will consist of an equal split (ie., 33 web pages from each website) of these popular news and non-news websites: CNN, Yahoo News, Food Network, Reddit, 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. Table 1 shows the experiment factors and Table 2 shows the experiment block design.

**7. Schedule**



## 8. Conclusion

In conclusion, different types of web information retrieval algorithms/techniques have been developed to mine HTML documents for patterns. Accordingly, this project seeks to find the specific pattern of an online news article within a web page. By transforming HTML documents into their respective DOM trees, CoreEx and Link-Target Identification will be used to extract information relevant to online news articles. This information will then be inputted into the C4.5 machine learning algorithm, which will then analyze how different and/or combined information retrieval algorithms perform in identifying online news articles.

## 9. Works Cited


[1]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.

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

[3]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.

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