CoreEx: Content Extraction from Online News Articles

Jyotika Prasad
Stanford University
Stanford, CA-94305
jyotika.prasad@gmail.com

Andreas Paepcke
Stanford University
Stanford, CA-94305
paepcke@cs.stanford.edu

ABSTRACT

We developed and tested a heuristic technique for extracting the main article from news site Web pages. We construct the DOM tree of the page and score every node based on the amount of text, the number of links it contains and additional heuristics. The method is site-independent and does not use any language-based features. We tested our algorithm on a set of 1120 news article pages from 27 domains. Our algorithm achieved over 97% precision and 98% recall, and an average processing speed of under 15ms per page.

Categories and Subject Descriptors: I.7.5 [Document Capture]: Document analysis,H.3.1 [Content Analysis and Indexing]

General Terms: Algorithms, Experimentation, Performance.

1. INTRODUCTION

Automatic core content extraction from Web pages is a challenging yet significant problem in the fields of Information Retrieval and Data Mining. The problem arises particularly on the World-Wide Web, because Web pages are often decorated with side bars, branding banners, and advertisements.

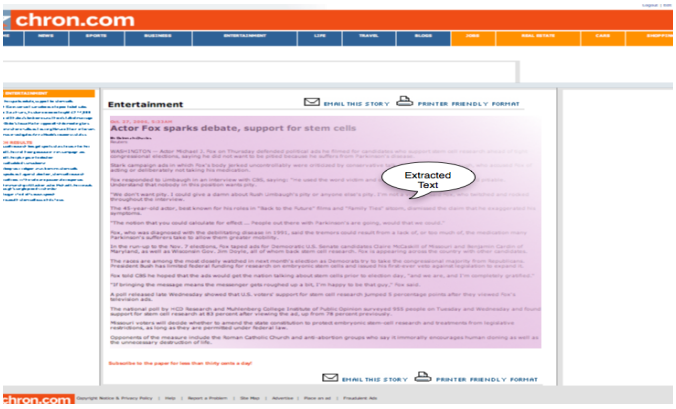

Figure 1: The shaded area is the text extracted by the algorithm

Large variations in page structure across domains make automatic extraction of core content difficult. Previous approaches have therefore constructed site-specific wrappers [2] , deployed machine learning [1] or used Natural Language Processing (NLP) techniques. These approaches, however, tend to require frequent updating or retraining to handle changes in structure, or employ computationally expensive steps.

We developed CoreEx, a simple heuristic technique that extracts the main article from online news Web pages. CoreEx uses a Document Object Model (DOM) tree representation of each page, where every node in the tree represents an HTML node in the page. We analyze the amount of text and number of links in every node, and use a heuristic measure to determine the node (or a set of nodes) most likely to contain the main content.

2. ALGORITHM

CoreEx is motivated by the observation that the main content in a news article page is contained in a DOM node, or a set of DOM nodes, with significantly more text than links. For each node in the DOM tree, we pick a subset of its children which have a high text-to-link ratio. We then calculate the score of the node as a function of the total text and total number of links contained in this subset. The algorithm selects the node with the highest score.

For every node in the DOM tree, we maintain a set of nodes S which stores a subset of its children, and calculate four counts: $textCnt$, $linkCnt$, $setTextCnt$ and $setLinkCnt$. Each $textCnt$ holds the number of words contained in one node. For interior nodes this count includes the sum of the words in the subtree to which the node is the root. Similarly, $linkCnt$ holds the number of links in or below the node. $textCnt$ of a node is thus recursively defined as the sum of the $textCnt$ of its children. $linkCnt$ follows a similar definition. To keep the score normalized between 0 and 1, a link is counted as one word of text.

We iterate through the node's children and add those whose text-to-link ratio is above a certain threshold to the previously fixed set S. $setTextCnt$ and $setLinkCnt$ are the sum of the $textCnt$ and $linkCnt$ of the nodes added to S.

More formally, once the DOM tree of a page is obtained, we walk up the tree, taking the following steps.

- For a terminal node T :
  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 **then**
  $textCnt(T) = 1$, $linkCnt(T) = 1$.
  else

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
ACM 978-1-59593-991-3/08/10.

$textCnt(T) = 0$, $linkCnt(T) = 0$.
end if

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

Once the counts are known for each node, the nodes are scored based on their *setTextCnt* and *setLinkCnt*. The DOM node with the highest score is picked as the *Main Content Node*, and the corresponding set S as the set of nodes that contains the content. 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.

The question then is how to compute a score that leads to good choices for core content.

2.1 Scoring

Our scoring function has two components. The first measures the ratio of the amount of text contained in the node to the number of links in the node. The second captures the fraction of the total text in the page that is contained in the node being scored. This component biases the score in favor of nodes which contain more text.

If $page_{text}$ is the total text the webpage contains, the score of node N is

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

$weight_{ratio}$ and $weight_{text}$ are weights assigned to both the components of the scoring function. Both the weights lie between 0 and 1, and have to sum to 1 (to keep the score between 0 and 1). $weight_{text}$ is much lower than $weight_{ratio}$, since we only require a gentle push towards the node that captures more text.

For an empirical determination of the weight constants and the threshold we ran a set of experiments with different values of $weight_{ratio}$, $weight_{text}$ and *threshold* on 500 news articles that were chosen at random from our test data (described in Section 3.1). The best performance was found to be $weight_{ratio} = 0.99$, $weight_{text} = 0.01$ and $threshold = 0.9$.

We perform two preprocessing steps before feeding the HTML into the algorithm. First, we ignore the following tags: *select, form, input, textarea* and *option*. This step is motivated by the observation that some pages contain text within these tags that is mostly not content. Secondly, we ignore all *script* nodes.

3. EXPERIMENT

3.1 Setup

Our test data consisted of 1620 manually labeled news article pages from 27 different sources, obtained from the MITRE corporation as per [1]. After using 500 pages for the one-time determination of our weight and threshold constants, we were left with 1120 pages for experimentation.

The tests were run on a workstation with four AMD Opteron 1.8 GHz dual core processors and 32 GB main memory . The CoreEx implementation was written in Java 1.5, and the system is single threaded. Our gold set was from [1]

3.2 Performance

We use a text-based performance measure, which works as follows.

Let the text in node n_i have a word count of w_i. We weight each node by w_i, so that missing a content node or including a non-content node in the result incurs a penalty of w_i. If $CorrectlyExtracted$ is the set of all nodes that the algorithm labeled correctly, $Extracted$ is the set of nodes extracted by our algorithm, and $Answer$ the set of nodes marked as the correct answer in the gold set, then the precision is

$$\frac{\sum_{n_i \in CorrectlyExtracted} w_i}{\sum_{n_i \in Extracted} w_i}$$

while the recall is

$$\frac{\sum_{n_i \in CorrectlyExtracted} w_i}{\sum_{n_i \in Answer} w_i}$$

The final system performs remarkably well, with 97% precision and 98% recall, yielding an F1 measure of 0.98. The simple node-based precision and recall, which does not take into account the amount of text in a node, are 93.0% and 92.5% resp. The average processing time per page was 13ms.

We found the performance to be consistent across the 27 domains, with 3 exceptions. The errors were mostly due to unusual HTML structures. Other caveats include image captions, which CoreEx labels as content but the gold set ignores. Finally, in certain pages when the article header appears far away from the body, CoreEx tends to ignore it.

4. CONCLUSION

We presented CoreEx, a simple, heuristic approach that extracts the main content from online news article pages.

CoreEx requires no training. It works fast and does not use any computationally expensive steps. For details see [3].

5. REFERENCES

[1] J. Gibson, B. Wellner, and S. Lubar. Adaptive web-page content identification. In *WIDM '07: Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management*, pp. 105-112, New York, NY, USA, 2007.

[2] D.-K. Kang and J. Choi. Metanews: An information agent for gathering news articles on the web. In *KDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 588-593, New York, NY, USA, 2002.

[3] J. Prasad and A. Paepcke. CoreEx: Content extraction from online news articles. Technical Report 2008-15, Stanford University, May 2008.