Content Extraction Using Diverse Feature Sets

Matthew E. Peters
SEOmoz
119 Pine St., Suite 400
Seattle, WA 98101
matt@seomoz.org

Dan Lecocq
SEOmoz
119 Pine St., Suite 400
Seattle, WA 98101
dan@seomoz.org

ABSTRACT

The goal of content extraction or boilerplate detection is to separate the main content from navigation chrome, advertising blocks, copyright notices and the like in web pages. In this paper we explore a machine learning approach to content extraction that combines diverse feature sets and methods. Our main contributions are: *a)* preliminary results that show combining feature sets generally improves performance; and *b)* a method for including semantic information via `id` and `class` attributes applicable to `HTML5`. We also show that performance decreases on a new benchmark data set that better represents modern chrome.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Content Extraction; Boilerplate Removal; Template Detection

1. INTRODUCTION AND RELATED WORK

Content Extraction (CE) algorithms have been well studied ([3] provides recent review), and there are many interesting and commercially useful applications.[1] In our case, we embed a CE engine as a filtering step inside a large scale information retrieval system.[2] Accordingly, our final application constrains the design of our CE algorithm. In particular we need it to be efficient and capable of running in a streaming fashion on only the crawled HTML.

Kohlschütter et al. [1] and CETR [4] are two interesting suitable approaches. Both algorithms work by splitting the HTML document into a sequence of "blocks", either by using the Document Object Model (DOM) in [1], or by line breaks as in [4]. They then classify the blocks into contentnon-content through supervised (decision trees; [1]) or unsupervised (k-means; [4]) learning. They both share the intuition that content blocks are more "dense" then non-content blocks, but differ greatly in the details of their approach. In this paper we combine these two approaches by using a single block definition and embedding the entire CETR algorithm inside a machine learning model.

We also introduce a new set of features heuristically designed to capture semantic information in the HTML code left behind by programmers. Empirically we note that many of the `id` and `class` attributes in modern HTML tags include tokens such as "comment", "header" and "nav". These descriptive names are used by programmers when writing CSS and Javascript and since they are chosen to be meaningful to the programmer, they embed some semantic information about the block's content.

Finally, we provide an updated set of 1381 HTML pages and associated gold standard. Web layouts have changed considerably in the last few years, and newer web pages have a wider variety and larger abundance of chrome including longer text passages such as article snippets. We benchmark existing algorithms on the new data and show that performance decreases as the task difficulty increases.

2. DATA SET

Our primary data set was collected in late 2012 and consists of English language pages from three sources: *i)* 999 pages randomly selected from popular RSS feeds with a large number of subscribers; *ii)* 204 pages from a selection of 23 large news sites (e.g. `bbc.com` and `nytimes.com`); and *iii)* 178 pages randomly sampled from a blog directory (`technoratti.com`) across all categories and authority ranges.[3]

The gold standard was extracted with a web browser by pasting it into a text file. We considered any article text including title, date and author information as well as any user generated comments to be content.

To benchmark against previous studies, we also use a portion of the data set from [4]. This includes both the Cleaneval data, as well as data from [2].

3. MACHINE LEARNING APPROACH

We follow [1] and split each HTML document into blocks using the DOM and a specified set of tags that modify the on

[1]For example, diffbot (www.diffbot.com)

[2]See `http://mz.cm/ZyIPCm` for an overview of the architecture.

[3]Our data and modeling code are available at `http://github.com/seomoz/dragnet`.

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

Table 1: Comparison of tokens in `class` attributes.

Token	Content : No Content	Percent of blocks
menu	1 : 373.6	2.2%
widget	1 : 314.1	4.6%
nav	1 : 68.9	3.3%
facebook	1 : 18.3	1.3%
top	1 : 13.3	1.9%
twitter	1 : 8.5	2.3%
title	1 : 3.3	10.5%
header	1 : 2.9	3.7%
comment	3.2 : 1	21.3%
author	4.9 : 1	7.9%
thread	5.0 : 1	3.0%
avatar	42.1 : 1	3.2%

screen layout (e.g. `<div>, <p>, <h1>`). We iterate through the DOM tree and create a new block each time one of these tags is encountered. Blocks without any text content are discarded. To build a labeled data set for machine learning we first split each document into blocks and associate each token in the full document with a single block. Then we solve the longest common sub-sequence problem to determine the correspondence of tokens in the gold standard with tokens in the full document so that we can compute the percent of each block's tokens extracted as content. Any block with more then 10% of the tokens extracted is tagged "content".

The data is randomly sampled into 70%/30% training/test split. We use L^2 regularized logistic regression, with the regularization parameter set via 5-fold cross validation in the training data.

4. MODEL FEATURES

The first set of features we include are the two most predictive shallow text (ST) features in [1], the text and link density. We follow their definition and compute the text density as the average number of tokens per 80 character line and the link density as the the anchor text percent. We also include the densities from the previous and next blocks as features.

Motivated by CETR [4], we also include the smoothed tag ratio (the ratio of the text length to the number of tags in the block) and the absolute smoothed derivative of the tag ratio. To preserve the spirit of their approach including the two-dimensional k-means clustering, we implement the entire CETR algorithm and and use the final prediction as a "feature" in our model.

4.1 Semantic features from `id` and `class`

We extract these features by first accumulating all tokens in the `id` and `class` attributes in each block, including the outer most tag that separates the block from the surrounding blocks. Then, we introduce a binary feature for each token in a specified list (collectively the IC features).

To create the list of tokens to include as features, we first list all tokens that occur frequently in our training data with their content to no-content odds ratio. We manually select tokens with a odds ratio larger then 2.5 that appear in more then 1% of either content or non-content blocks and have an obvious semantic interpretation. Table 1 lists a few selected tokens in the `class` attribute along with their odds ratios. Tokens in the upper portion of the table are more likely to occur in non-content blocks, while those in the bottom are more likely to occur in content blocks. Interestingly "facebook" and "twitter" are fairly common, and are more likely to occur in non-content (presumably associated with social plug-ins). All told, we include 8 `id` features and 24 `class` features.

Table 2: Model comparison using mean F_1-score

	Cleaneval-EN	Big 5	2012 Train	2012 Test
Baseline	0.899	0.625	0.621	0.623
CETR	0.919	0.794	0.741	0.735
IC	0.887	0.641	0.709	0.701
ST	0.904	0.854	0.817	0.809
ST+IC	0.896	0.858	0.836	0.824
ST+IC+CETR	0.907	0.877	0.848	0.836

5. RESULTS

Table 2 compares the mean F_1-scores for different feature combinations. We use the method in [4] to compute the F_1-scores, where each word in the document is distinct even if two words are lexically the same. To demonstrate the versatility the learning approach, we train only on the 2012 Train set and make predictions on the rest of the data.

In general, combining features does improve model performance, even if the individual model performance is poor.

Model performance decreases on the newer 2012 data when compared to the older data sets. Individually, the IC features give a small performance improvement over the baseline, and not surprisingly perform poorly on the older data when CSS was less popular. The low individual performance of the IC features may be attributable to the fact that we accumulate tokens in each block, but meaningful tokens may appear outside the block at higher levels in the DOM. The small train/test differences suggest we may be slightly overfitting.

6. CONCLUSIONS AND FUTURE WORK

Even though these are preliminary results, combining features does overall appear to improve model performance. This suggests that adding additional features (e.g. CETD [3]) or modifying the definition of the IC features will further improve performance. We'd like to benchmark performance against a wider variety of pages, including the more recent data in [3]. Finally, we plan to try a more sophisticated learning algorithm then logistic regression.

7. REFERENCES

[1] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of WSDM '10*, pages 441–450. ACM, 2010.

[2] J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of WWW '09*, pages 971–980. ACM, 2009.

[3] F. Sun, D. Song, and L. Liao. Dom based content extraction via text density. In *SIGIR*, volume 11, pages 245–254, 2011.

[4] T. Weninger, W. H. Hsu, and J. Han. Cetr: content extraction via tag ratios. In *Proceedings of WWW '10*, pages 971–980. ACM, 2010.