Web Content Extraction Through Machine Learning (Project Milestone) *

Ziyan Zhou
ziyanjoe@stanford.edu

Muntasir Mashuq
muntasir@stanford.edu

ABSTRACT

Web content extraction is a key technology for enabling an array of applications aimed at understanding the web. While fully automated web extraction has been studied extensively, they often focus on extracting structured data that appear in multiples on a single web page. This project aims to extract not-so-structured web content, like news articles, that appear only once in a noisy web pages. Our approach tries to classify text blocks using a mixture of visual and language independent features. We uses clustering to automatically label a large dataset, then train our model to classify each data point.

Keywords

Content Extraction; SVM; DBSCAN; Web Scraping

1. INTRODUCTION AND RELATED WORK

Content extraction algorithms have been well studied and there are many interesting and useful applications[1][3]. A survey by Laender at el.[5] systematically covers a wide range of techniques used for this purpose. These tools range from wrapper development to NLP-based and modeling-based approach. Miao at el.[6] proposed a fully automatic tag path clustering approach to extract structures on a single page. Sun at el.[7] applied a language independent approach by looking at text density at each area of the document.

We wanted a better approach to this problem, to train a SVM classifier that is language independent, that incorporates visual features like font size, color and style, line spacing and block size etc., while also maintaining site and domain agnostic.

For this milestone, we wanted to obtain a sizable dataset with desired visual features and correct labeling in a short amount of time. Manual labeling is therefore not a viable option. So a method is devised to automatically label obtained dataset through clustering.

*Source code for this project is avaiable on GitHub under the MIT License at `https://github.com/ziyan/spider`

Table 1: Collected Dataset

Site	Language	# of Pages
NPR	English	25
Prothom Alo	Begali	24
QQ	Simplified Chinese	25
Sina	Simplified Chinese	25
TechCrunch	English	16
The Verge	English	16
USA Today	English	20

2. DATA COLLECTION

The dataset is extracted from serveral popular English, Simplified Chinese, and Bengali news article sites on the Internet.[1] See *Table 1.*

Each page is downloaded and rendered in a virtual webkit browser.[2], restoring its original layout intended for human audience. Javascript is then injected into the page to efficiently inspect DOM elements. Text enclosing blocks are identified. For each DOM element containing text, the algorithm finds its closest parent element that are displayed as block. Inline links and other text decorators are not individually identified. Instead, their common parent element, which might be a paragraph or a div, is identified as a whole.[2]

For each chosen block, a set of features are then extracted, including the size and position of the block, the contained text, the font configurations, color, line height, tag path, etc. In fact, there are over 300 different CSS properties for each block. Our algorithm automatically extracts those with a non-default value. Therefore, the number of features varies from block to block in the data collection phase.

3. CLUSTERING BLOCKS

We noticed that text blocks on the page are not necessarily clustered according to a known distribution (i.e. Gaussian

[1]The collected dataset is available at `https://github.com/ziyan/spider/tree/master/data`

[2]See a screenshot rendered by the virtual browser with extracted text blocks outlined by red boxes: `https://raw.github.com/ziyan/spider/master/docs/milestone/blocks.png`

distribution). Rather similar text blocks, like navigational buttons, horizontally distributed across the top of the page, can cluster together in any shape. We also noticed that some noisy elements only appears once on some of the pages. Furthermore, it is unclear that how many cluster there will be, since it depends heavily on the design and layout of the web page. Therefore, we have chosen DBSCAN[4], density-based spatial clustering of applications with noise, as our clustering algorithm. This algorithm handles the the problem of unknown number of clusters, unknown shape of clusters and noise quite nicely.

The clustering result shows that, the body text of the articles across multiple pages, which usually consist of blocks, are clustered together in a single cluster. The navigational links are also clustered together, as well as the links in the footers. Other common elements, like similar sidebar, advertisement, comments, etc., are also successfully clustered. Given all the visual features represented by the CSS properties, the algorithm is quite effective in discerning the different visual elements on the pages.

4. AUTOMATIC LABELING

Although all the blocks have been clustered quite precisely based on their visual properties, it is not trivial to find out which cluster contains the content text. The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. The number of clusters varies from website to website. Thus there is no way to know which clusters contain the content text we need. For our dataset, the number of output clusters ranges from a couple of clusters to more than 20.

Fortunately, most websites have some metadata stored in meta tags in order to accommodate Google's web crawler or Facebook's Open Graph. So we can extract the description of an article by parsing the meta tags in the page. The description usually contains a brief summary or just the first few words of the article content. With that, we are able to calculate the similarity between each cluster and the description using the Longest Common Subsequence (LCS) algorithm. The longer the common subsequence (we call this number "relevance score"), the more likely is the cluster to contain the content text. Note we used LCS instead of Term Frequency Inverse Document Frequency to ensure language independency as word segmentations are done differently for different languages.

At first we tried to automatically label the blocks by finding the best cluster local to each page on a website. After training the SVM on this labeling, we found that for some websites, the precision on the test set was not ideal. On closer examinations, we found that on some rare pages, the best cluster that most relevently matches the page description would be user generated contents like comments. To fix this issue, we implemented *Algorithm 1* which scores the blocks across the entire website (rather than working locally on a single page).

This algorithm perform very accurately and removes the occasional labeling hiccups on a single page. After training the SVM with this updated labeling, the precision on the test dramatically improved.

```
score := array of zeros with length equal to # of clusters;
for each cluster i of a website do
    for each block j under that cluster do
        score[i] := score[i] + relevance score of block j;
    end
end
Pick the cluster(C) with the highest similarity score;
Label all the blocks of in the same cluster C as 1;
Label all other blocks as 0;
```

Algorithm 1: Labeling from global score in a website

5. SVM AND RESULT EVALUATION

We constructed a support vector classifier with a linear kernel. Due to the inbalance between the number of content blocks and the number of non-content blocks on a web page, we applied class weights to give our positive (content) examples higher weights.

For this milestone, we trained a different classifier for each site using only labeled text blocks from that particular site. And we perform a 4-fold cross validation to evaluate the performance of our classifier.

6. FEATURE SELECTION

In order to better evaluate our approach, we experimented with different feature sets.

6.1 Text Length

A common naive approach to web content extraction is to find the longgest continuous text in a web page. Therefore, to establish a baseline, we've experimented with using only text length as our feature set. The text length of each text block is simply the number of non-whitespace characters in the block. We normalize this number by scaling it to have a zero mean and variance of one.

The results show that this approach works for some sites but performs quite poorly for others. See *Table 2*.

6.2 Tag Path

The second approach is to use tag path of the block. Tag path is the path from the root of the document to the text block. For this milestone, we only incorporated the element type in the tag path. For example, a paragraph inside a div container in the body of the HTML page would have a tag path like *"body > div > p"*. We treat each different tag path uniquely and vectorized this discrete feature then scaling it to $\{-1, +1\}$.

The results show that this approach works quite well in general but fails for some particular site. We found that in the failure cases, the tag paths are indeed the same for content and some non-content. Therefore, this feature alone cannot sufficiently distinguish the two classes of blocks. See *Table 3*.

6.3 CSS Properties

The third approach is to use all visual related CSS properties (color, font-size, font-style, line-height etc.) as features. These features are treated as discrete features and are vectorized and scaled to $\{-1, +1\}$.

Table 2: Result using Text Length only

Site	Class	Precision	Recall	F_1
NPR	non-content	98.16%	95.00%	96.55%
NPR	content	78.99%	91.34%	84.70%
Prothom Alo	non-content	99.03%	95.93%	97.45%
Prothom Alo	content	42.14%	75.01%	53.31%
QQ	non-content	98.60%	91.62%	94.97%
QQ	content	31.95%	75.43%	44.70%
Sina	non-content	96.52%	96.80%	96.66%
Sina	content	62.92%	60.63%	61.69%
TechCrunch	non-content	99.30%	93.51%	96.32%
TechCrunch	content	32.84%	83.18%	46.96%
The Verge	non-content	99.47%	97.06%	98.25%
The Verge	content	53.56%	86.86%	66.08%
USA Today	non-content	93.72%	90.40%	92.00%
USA Today	content	75.45%	82.98%	78.83%

Table 3: Result using Tag Path only

Site	Class	Precision	Recall	F_1
NPR	non-content	100.00%	99.94%	99.97%
NPR	content	99.74%	100.00%	99.87%
Prothom Alo	non-content	100.00%	100.00%	100.00%
Prothom Alo	content	100.00%	100.00%	100.00%
QQ	non-content	100.00%	96.41%	98.17%
QQ	content	58.68%	100.00%	73.88%
Sina	non-content	100.00%	99.50%	99.75%
Sina	content	94.61%	100.00%	97.22%
TechCrunch	non-content	100.00%	100.00%	100.00%
TechCrunch	content	100.00%	100.00%	100.00%
The Verge	non-content	100.00%	99.68%	99.84%
The Verge	content	92.34%	100.00%	96.00%
USA Today	non-content	100.00%	100.00%	100.00%
USA Today	content	100.00%	100.00%	100.00%

The results show that similar to tag path, these features work quite well in general but not ideal for some sites. Especially for sites that do not visually distinguish content with other parts of the page, these features are not sufficient. See *Table 4*.

Table 4: Result using CSS Properties

Site	Class	Precision	Recall	F_1
TechCrunch	non-content	100.00%	99.75%	99.88%
TechCrunch	content	94.27%	100.00%	97.03%

Note: rows that are entirely 100% omitted.

6.4 Combined Features

When we combine text length, tag path and CSS properties, the results show that we can classify all text blocks within our dataset perfectly (even in each iteration of the 4-fold cross validation).

7. REMAINING WORK

For this rest of this quarter, we will be working on a few improvements to this project.

7.1 Accurate Labeling

Our automatic labeling method is not perfect. While it correctly labels some content blocks, it misses our content blocks are slightly different (blockquote or a list within an article). We will work on improving the labeling method or even employing manual labeling.

7.2 Better Features

Some of our features have room for improvement. For example, tag path can include element ID and CSS classes. It can even include positional information to acurrately pinpoint a specific element in the web page. Using these additional information, perhaps tag path as a feature will sufficiently distinguish content vs non-content.

On the other hand, some CSS properties can be bucketized or even transformed to linear values. For example, color of the text is currently treated as discrete values, but they are indeed continuous in the color space.

7.3 Demo

We are planning to create a Chrome extension to allow anyone to use our classifier on the webpages they visit. This will dramatically expand our dataset and will also allow human assisted labeling of a more diversified dataset, resulting in a better evaluation of our approaches.

8. REFERENCES

[1] Diffbot: Extract content from standard page types: articles/blog posts, front pages, image and product pages. `http://www.diffbot.com/`.

[2] Phantomjs is a headless webkit scriptable with a javascript api. `http://www.phantomjs.org/`.

[3] Readability turns any web page into a clean view for reading now or later on your computer, smartphone, or tablet. `http://www.readability.com/`.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[5] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. In *ACM SIGMOD Record*, pages 84–93, 2002.

[6] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the web using tag path clustering. In *Proceedings of the 18th international conference on World wide web*, pages 981–990, April 2009.

[7] F. Sun, D. Song, and L. Liao. Dom based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 245–254. ACM, 2011.