A Machine Learning Approach to Webpage Content Exraction

Jiawei Yao
Department of CS
Stanford University
Email: jwyao@stanford.edu

Xinhui Zuo
Department of MS&E
Stanford University
Email: xzuo@stanford.edu

***Abstract*—Apart from the main content, a webpage usually also contains boilerplate elements such as navigation panels, advertisements and comments. These additional elements are typically not related to the actual content and can be treated as noise that needs to be removed properly to improve the user's reading experience. This is a difficult problem as HTML is loose in semantics and flexible in structure. In this paper, we model the webpage content extraction problem as a classification problem and employ machine learning method to solve it. For each text block in the HTML document, we select a set of relevant features, based on which an SVM classifier is used to predict whether this text block is content or non-content. The results show that our approach can achieve performance comparable to some popular existing algorithms in this field.**

***Keywords*—text mining, content extraction, web semantic, algorithm**

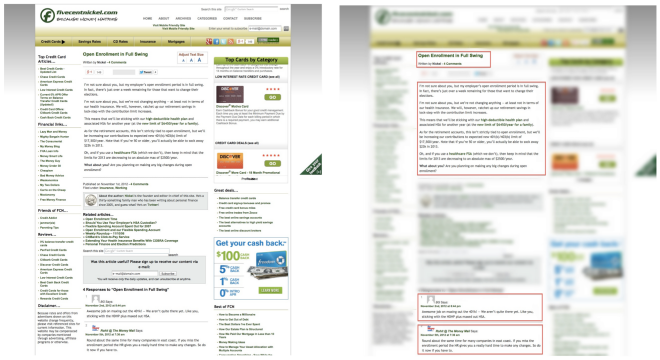

Fig. 1: Example webpage: original on the left, annotated main content on the right

I. Introduction

Over the years, the Internet has become lots of people's primary source of information and people are spending more and more time on web browsing. The news sites, blogs or other information-rich websites, for various reasons, displays many boilerplate information such as advertisements and navigation panels alongside the main content (see Fig. 1 for example). It is desirable that these boilerplates be removed because: 1. generally they are a distraction to the users and overload of them greatly breaks reading experience, 2. they hamper information retrieval and the removal of them will benefit search engines, and 3. not all websites are optimized for hand-held devices which have relatively limited screen space and extraction of main content can provide a hand-held device friendly view. The popularity of many read-it-later services like Instapaper, Pocket and Readability demonstrates the important of content extraction.

However, the removal of these noisy elements is nontrivial, because HTML is not a semantically strict language and enforces no structural constraints, so content providers are free to compose HTML in whatever way they like. Although the latest standard, namely HTML5, promotes semantics of HTML by introducing tags like `<article>`, `<nav>` and `<aside>`, the standard has not been widely adopted. To improve this situation, webpage Content Extraction (CE) algorithms were introduced and CE has been an active research area for the past decade. Many early efforts rely heavily on heuristics about webpages with main content, such as text density, text-to-tag ratio, etc, which are not universally applicable and might break randomly because of the flexibility of HTML.

In this paper, we put webpage content extraction in the machine learning setting and introduces a supervised learning approach to content extraction. We break an HTML page into text blocks which is structure-independent and extract features from the text blocks. An SVM classifier is trained on the text blocks and will be later used to predict the content of test documents. We also do semantic analysis based on `id` and `class` attributes associated with text blocks using Naive Bayes algorithm. The result of semantic analysis can be incorporated into features of text blocks to improve classification accuracy. Experiments are conducted on two datasets from 2008 and 2012, which represent characteristics of webpages from several years ago and webpages with more recent trends respectively.

The rest of this paper is organized as follows: Section II gives an overview of works related to airplane model recognition problem. In Section III, our approach of airplane model recognition is explained in detail. In Section IV, the dataset, design and evaluation metrics of the experiment are described. Results of the experiment and discussion are presented as well. Finally, Section V gives conclusion of this paper and puts forward some future work.

II. Related Works

Some of the existing algorithms [1][2][3][4] are based on the observation that compared to the content fragments, the non-content fragments of an HTML document are usually highly formatted with more tags and also contain less text and shorter sentences. As a result, non-content fragments have a higher tag density while the content fragments have a higher text density. Specifically, a number of features on the block level are evaluated according to their information gain

(Kullback-Leibler-divergence) in [1], among which the number of words and link density are proved to be the most relevant ones.

Alternatively, when a web page is divided into a tree whose nodes are visually grouped blocks, spatial and content features may also be used to detect the non-content nodes in this tree as in [5]. This approach is called Vision-based Page Segmentation (VIPS) technique and it suffers from high computational cost to render a page in order to analyze it. There are also some other approaches like template detection algorithms which classify the identical parts found in all webpages as noisy components [6]. Apparently, the application of these algorithm is limited to webpages from the same website, and thus it would be cumbersome to build models and templates for different websites or different versions of one website.

In this project, we adopted the definition of blocks in [1] and model webpage context extraction as a classification problem on the block level. For this classification problem, we have selected three types for features: text features, relative position and `id` & `class` token feature and used SVM as classifier. The blocks classified as content would then be merged to construct "clean" webpages, which are compared with the gold standard. The results of our machine learning approach is comparable to the performance of the algorithm in [4].

III. Our Approach

Our approach is, like many other popular content extraction methods, text block based. We blockify training documents and train an SVM classifier based on features extracted form the blocks. For test documents, the same blockify algorithm is applied and the blocks classified as content are extracted to construct the main content.

A. Blockifying

Although many previous works are block-based, they do not document the blockifying process clearly. In our approach, we use HTML parser to traverse the HTML DOM tree and accumulate relevant information such as text and links as traversing, when the beginning or end of a block-level element[1] is reached, we output the accumulated text as a text block.

B. Feature Extraction

As we are transforming the content extraction problem into a classification problem, we wish to use features that are strongly indicative whether a text block is content or not. In our approach, we select three types of features: text features, relative position, and `id&class` token feature.

The first type is called text features, which are extracted based on text properties inside a text block. According to the evaluation of block features based on their information gain (Kullback-Leibler-divergence) in [1], for each block we have selected seven most relevant features:

- number of words in this block and the quotient to its previous block
- average sentence length in this block and the quotient to its previous block
- text density in this block and the quotient to its previous block
- link density in this block

The definition of features such as number of words and average sentence length are intuitive and trivial, and the link density is calculated as the ratio of number of words within all `<a>` tags associated with the text block to the total number of words in the block. As in [1], we assum that the there are 80 characters in a line, and the text density of a block is defined as the average number of words in one line, which is calculated as the ratio of the total number of words to the total number of lines.

The second type of feature is the relative position of a block in the webpage. To be specific, for a document divided into N blocks, we discretize their position into M relative positions, and the relative position of the n^{th} block is

$$relative\_pos(n) = \lfloor \frac{n}{N} \times M \rfloor$$

The last type of feature is the `id&class` token feature. In modern HTML documents, the `id` and `class` attributes capture the semantic information in the HTML left behind by programmers[4]. For example, tokens such as `ad` and `nav` usually indicate that the associated elements are non-content. But of course we don't want to manually analyze and collect such indicative tokens, which will be not only heuristic-based but also non-efficient. Thus we employed the event-model Naive Bayes algorithm to learn the top N (which we set to 10) `id` and `class` tokens that are the surest indicators of non-content blocks[2]. We compute the probability and score of k-th token as follows (assuming there are m blocks):

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i + |V|}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i + |V|}$$

$$score_k = log\frac{\phi_{k|y=0}}{\phi_{k|y=1}} \times \#blocks - with(k)$$

The first two equations are multinomial event model with Laplace smoothing and the third equation is weighted score of k-th token. We select tokens with 10 largest scores[3] and add 10 binary features indicating whether a block has the corresponding token or not.

[1] Block-level elements are formatted with a line break before and after the element (thereby creating a stand-alone block of content). Note that we also treat `<br>` as block-level element as it modifies text flow.

[2] With preliminary analysis, we found that indicative tokens for content blocks vary but for non-content blocks are relatively stable, so we use indicators for non-content blocks only.

[3] A sample list of such tokens from our are: menu, module, nav, widget, sidebar, footer, right, dropdown, cat, nagivation.

C. SVM Training and Classification

After feature extraction, we label each block as content or non-content based on whether they appear in the gold standard. Then we employ SVM to this binary classification problem on the block level with eighteen features as decried in the previous section. Following the suggestions in [7], we first scale all the attributes' value to the same range so that the features with a higher absolute value wouldn't dominate the value of the kernel output. Since we are using large datasets and the dimension of our feature set is relatively small, radial basis function (RBF) would be a good choice as the kernel. We also tried parameter tuning as described in [7] but the improvement is negligible so in our final evaluation we just use default parameters of C and γ.

After the SVM classifier is obtained, when a test document is given, we first use the same blockify method to break test document into blocks and then use the classifier to label the blocks as content or non-content. Texts from all the content blocks are then merged to form the extracted content.

IV. Evaluation

A. Datasets

Two datasets were used to evaluate our approach. The first one is the L3S dataset, which consists of 621 manually classified Google News articles from 408 different web sites in 2008 from [1], and the second one is the Dragnet dataset, which consists of 1380 webpages from RSS feeds, news websites and blogs in 2012 from [4]. Each dataset have either manually marked content or manually extracted content, which can be used as gold standard.

With some inspection, documents from the L3S dataset are generally simple-structured and most text are main content, while documents from the Dragnet dataset are more complex-structured with heavy boilerplate elements, some even containing long comments. Therefore, both datasets represent webpage design trend of their respective times. As we aim to build a generic content extraction tool, a well handling of both datasets would be highly desirable.

B. Performance metrics

The evaluation metrics are based on both the bag of words and longest common subsequence of gold standards and extracted contents. Specifically, precision, recall and F_1 scores are calculated. For evaluation with bag of words, we have the precision and recall as

$$P_1 = \frac{|W_P \cap W_L|}{|W_P|}$$

$$R_1 = \frac{|W_P \cap W_L|}{|W_L|}$$

where W_P is the set of words we have classified as content and W_L is the set of words in the gold standard.

For evaluation with longest common subsequence, where each word in the document is distinct even if two worlds are lexically the same. This metric is more meaningful as what users finally see are paragraphs of text instead of discrete words. Let $LCS(P, L)$ denote the longest common subsequence between the predicted content and gold standard, we have the precision and recall as

$$P_2 = \frac{LCS(P, L).length}{P.length}$$

$$R_2 = \frac{LCS(P, L).length}{L.length}$$

And the F_1 score is calculated as

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

We define the metric as token-level F_1 if $P = P_1, R = R_1$, or LCS F_1 if $P = P_2, R = R_2$. When evaluation, we use both token-level F_1 and LCS F_1 as evaluation metric.

C. Experiment Design

In evaluation, we randomly chose 600 documents from each datasets and run 5-fold cross validation on it. For each run we collect evaluation metric described as before and use the average as the final metric. We apply different feature set combinations to train the classifier:

- text features only
- text features + relative position features
- text features + relative position features + `id&class` token features

We use method from [4] as baseline, which uses combination of features from [1], [2] and CSS features and is method with the best performance we are able to find.

D. Results and Discussion

The experiment results on the L3S dataset is in Fig. 2 and results on the Dragnet dataset is in Fig. 3. The blue bars are token-level F_1 and the green bars are LCS F_1.

Firstly, we can see that for both datasets our approach is comparable to the baseline method (and even outperforms it a bit in the L3S dataset). This is a good indicator that our approach can accurately extract main content from documents with diverse characteristics. It is noticeable that on the L3S dataset both F_1 measures are close to 0.9 while on the Dragnet dataset they are around 0.8. This aligns with our previous observation that documents from Dragnet datasets are more recent and complex-structured, which makes content extraction much harder.

Secondly, considering different feature sets, we can see that on the L3S dataset, the differences in F_1 measures using different feature combinations are very small. But on the Dragnet dataset, adding `id&class` token features gives a non-trivial increase in F_1. This may result from the trend that HTML documents contain more semantic `id` & `class` tokens in recent years as semantic HTML has been promoted. Looking forward, we believe that with HTML5 document become more semantically rich, the `id&class` token features (or other equivalent features capturing semantic part of a document) will play a more important role in content extraction.

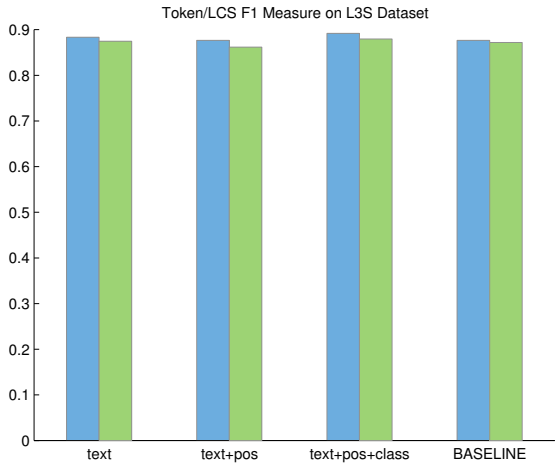

Fig. 2: Results on L3S dataset

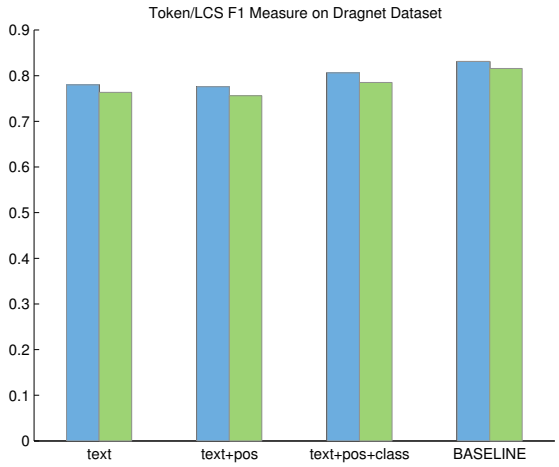

Fig. 3: Results on Dragnet dataset

Thirdly, we also manually inspected extracted content of some document as we believe what really matters is what the user actually sees. To our disappointment, although the metrics are good, some of the extracted content misses some parts of actual main content, some in the beginning, some in the end, and others in between. From a realistic setting, extracting extraneous part is bearable, but missing some critical content is never acceptable. Maybe we can add an evaluation measure which record whether all text in gold standard appears in extracted content and apply some technique to ensure that in the future.

Last but not least, LCS F_1 is consistently lower than token-level F_1 in each experiment, which is expected as LCS F_1 is a more rigorous measure. However, this also reminds us that for evaluation's sake, token-level F_1 can be a reasonable proxy of LCS F_1, which is more expensive to compute.

V. Conclusion and Future Work

Webpage content extraction is a difficult and interesting problem to tackle. In this paper, we put this problem into a machine learning perspective and introduced a supervised learning approach which can solve this problem to some degree. With preliminary evaluation using two F_1 measures, our method can achieve comparable performance to the state-of-the-art method.

However, our approach is far from perfect (actually, content extraction problem is an AI-complete problem so there might never be a perfect solution), for example it may extract more content than necessary or even miss part of main content. Some techniques shall be added to ensure the completeness of the extracted content - many previous works assume the continuity of main content, which may not be 100% true but is applicable to most cases.

At this stage, we used a set of features that are reported to have the highest information gain (Kullback-Leibler-divergence) in [1]. In the future, other feature selection methods, such as forward/backward feature selection, can be employed to evaluate a larger set of relevant features.

Moreover, at this moment we only have the data set from [1], which has been manually classified to set the gold standard on the block level. When data set with gold standard at a larger scale is available to us, instead of classification problem defined on the block level as in [1], we would like to explore the possibility of intra-document analysis. We expect the HTML DOM tree structure and semantics to have potential contribution to content and non-content classification problem.

Acknowledgement

We would like to thank the CS299 staff and others for reviewing our poster and offering constructive suggestions. We greatly appreciate Mr. Matt Peters for kindly offering us the Dragnet dataset for evaluation.

References

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

[2] T. Weninger, W. H. Hsu, and J. Han. *CETR: Content Extraction via Tag Ratios*. In Proceedings of WWW '10, pages 971-980. ACM, 2010.

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

[4] M. Peters, and D. Lecocq. *Content Extraction Using Diverse Feature Sets*. In Proceedings of WWW '13, pages 89-90, 2013.

[5] R. Song, H. Liu, J. Wen, and W. Ma. *Learning block importance models for web pages*. In Proceedings of WWW '04, pages 203-211, 2004.

[6] L. Chen, S. Ye, and X. Li. *Template detection for large scale search engines*. In Proceedings of SAC '06, pages 1094-1098, 2006.

[7] C. Hsu, C. Chang, and C. Lin. *A Practical Guide to Support Vector Classification*. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003.