WebNewsExtraction

Gowtham Rangarajan R
College of Computer and Information Sciences
University of Massachusetts, Amherst
ranga@cs.umass.edu

1 Introduction

This report presents a solution to extracting Title, Published Date (referred to as date elsewhere in the report) and Content elements from web-news articles using machine learning methods. The explosive growth of internet has resulted in an exponentially growing number of news articles every year. This vast resource of news-items could be mined to learn more about the society. Consequentially, there has been growing interest in automatically extracting structured information from them. To this end, we recognize the importance of a text extraction system that provides a structured representation of web news article identifying an article's title, publish Date and main content. Extracting content may not be straightforward using heuristic based approach because of the evolving nature of web. Further more, far less time can be spent in training statistical machine learning classifiers than coming up with carefully handcrafted if-else / regular expressions statements that keeps up with the evolving web and that works across a wide array of web articles. The are two main contributions, of this report, to tackle the problem mentioned above are

- Introduction of a web annotation tool and an alignment tool to annotate HTML pages and convert them into feature vectors that can be used for learning
- A classifier that uses the above set up to identify title, published date and main content from a webnews article

The report is organized into 5 sections . Section 2 discusses related work in a close but related field of content extraction. Section 3 briefly presents an overview of the proposed solution and describes the dataset that was collected . Section 4 discusses the features and the learning methods that were used for this task. Section 5 attempts to be a documentation for the software proposed. Section 6 concludes with the results obtained for the task and discusses about possible useful extensions to the work.

2 Related work

Content extraction techniques from HTML documents have been well studied over the past two decades. They were introduced early in the last decade by Rahman et al., [1] and since then numerous techniques have been developed. These techniques initially focused on creating regular expressions on plain text to identify content [2] , creating custom extractors for websites [3], exploiting the DOM tree representation for clues [4] etc.,. The biggest disadvantage with these is the need to design intelligent heuristic/regular expressions to capture non-content blocks of text that works well across domains. Moreover, the evolving nature of web poses a significant amount of effort in extending the shelf life of such systems. Despite these potential drawbacks few popular open source software [Goose] use heuristic techniques and regular-expressions to extract relevant content.

During the later half of the decade, the research progressed towards using intra domain DOM tree template matching algorithms [5]. These algorithms required a significant deal of analysis for a specific domain before being production ready. Their failure to adapt to unseen websites restricts their applicability severely in the modern web. During the same period, heuristics that capture content [6] were proposed. These heuristics remain relevant as features for machine learning techniques to determine content. During the recent half of the decade there has been interest in treating the task as supervised machine learning problem [7] [8] and proposing relevant features [**?**]. Kohlschutter et., al [7] introduced shallow text features such as average word length, captilization ratio link density to distinguish content from boilerplate. Weninger et., al [9] introduced CETR content which computes the number of words under a particular tag and uses them as features. Document Slope Curves were introduced by Gottron. He later introduced an Advanced DSC (ADSC) [8] in which a windowing technique is used to locate document regions in which word tokens are more frequent than tag tokens. These Features are either extracted from the DOM node [7] [8] or by considering line breaks [**?**]and are learnt using K-mean [9] or decision trees [7]. Most of these features have been benchmarked by Weninger. et., al [10] recently. Conditional Random Fields have also been used to clean up webpages. [11].

There has also been recent interest in using vision based methods [12] [13] where the DOM tree is clustered based on the visual content of partially rendered pages [14]. However, recent popular open source software use supervised learning methods (readability, boilerpipe) or hand engineered regular expressions (goose) to keep up with the modern day web.

3 Dataset

3.1 Overview

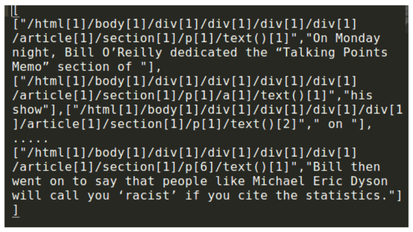
A webpage could be represented as a DOM tree, whose leaf nodes contain text that may or may not appear on the screen. Our goal is to classify these leaf nodes into 4 classes namely title, content, date and boilerplate. We pose the above problem as a supervised learning problem, where we collect the ground truth data that contains text nodes and its corresponding class. We obtain the ground truth DOM nodes by manually annotating web news articles and collecting them using a chrome extension developed for this project. We perform an inorder traversal of the DOM nodes and collect the text nodes to be classified and match them with the ground truth data. We use this path information obtained during the inorder traversal (which is assumed to be the order in which an HTML page appears on the screen) of the DOM tree and use the properties of the node along with other relevant information from the DOM tree and extract features for each of these nodes. These labeled examples along with their feature vectors, that are represented as matrices, are used with different machine learning algorithms to find the best estimator for the task.

3.2 Dataset

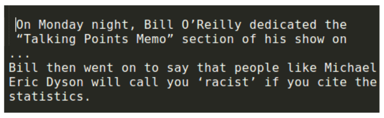
Around 250 webpages were annotated with title, content and published date (date). Each annotated webpage consisted of a list of tuple containing the xpath expression from the root of the DOM tree and the textcontent of selected nodes that correspond to title, content and date along with the selected text for these classes (1) . Both these information were collected since there was a possibility that the selected text may not entirely match the text present in a text node.

The collected dataset is different from the datasets [15][16] that are used for a similar task, where only the text information is collected which is later matched with the list of textnodes using an LCS procedure, because we wanted to locate the ground truth nodes exactly rather than approximately. The implementation details of the extension and the aligning procedure is postponed till section 5.

The dataset released **??** contains train.csv having 3 columns namely fileids, archived URL (in hobbes) and the domain URL (just for exploratory purposes). The fileid column acts as a key that is used to match the collected data with the archived url. The dataset contains a *FullDataset* folder which is parent to 3 different folders : *Title*, *Publish_Date* and *Body*. These folders contain the correspondingly annotated webnews article in the following pattern *fid.groundtruth* and *fid.WithNode.groundtruth*, (as the names suggest they are annotations with and without node information). The archived url is not considered to be annotated if and only if there isn't a body, title and



```
[
["/html[1]/body[1]/div[1]/div[1]/div[1]/div[1]
/article[1]/section[1]/p[1]/text()[1]","On Monday
night, Bill O’Reilly dedicated the “Talking Points
Memo” section of "],
["/html[1]/body[1]/div[1]/div[1]/div[1]/div[1]
/article[1]/section[1]/p[1]/a[1]/text()[1]","his
show"],["/html[1]/body[1]/div[1]/div[1]/div[1]/div[1
]/article[1]/section[1]/p[1]/text()[2]"," on "],
.....
["/html[1]/body[1]/div[1]/div[1]/div[1]/div[1]
/article[1]/section[1]/p[6]/text()[1]","Bill then
went on to say that people like Michael Eric Dyson
will call you ‘racist’ if you cite the statistics."]
]
```



```
On Monday night, Bill O’Reilly dedicated the
 “Talking Points Memo” section of his show on
...
Bill then went on to say that people like Michael
Eric Dyson will call you ‘racist’ if you cite the
statistics.
```

Figure 1: Dataset: Groundtruth examples collected. Left: example fid.WithNode.groundtruthdata, Right: example fid.groundtruthdata

Folder	Filename	Description
train/	field.csv	columns having fileid archived_url and url
FullDataset/Title FullDataset/Title	fileid.groundtruth Title/fileid.WithNode.groundtruth	Title text json array of list of xpathexpression, textvalue at node
FullDataset/Publish_Date FullDataset/Publish_Date	fileid.groundtruth fileid.WithNode.groundtruth	Date text json array of list of xpathexpression, textvalue at node
	Body/fileid.groundtruth Body/fileid.WithNode.groundtruth	Body text json array of list of xpathexpression, textvalue at node

Table 1: Dataset: Description

a Date label with the corresponding fileid in the dataset. The absence of either a title class or a date class may indicate the absence of a good match for these classes in the webpage. Please refer to the Table 1 for more details

3.3 Exploratory analysis

The dataset consists of **269** webpages that were annotated from **239** unique domains, with an average of **1.13** news articles per domain. It has **65194** text nodes out of which only **266** nodes were classified as Title, **3803** represented content and **204** represented date. In other words, on an average, 250 text nodes are present in a webpage out of which 15 nodes are classified as content and 1 as title and 1 as Date. Some of the domains that were visited include alaskadispatch.com ,dailytelegraph.com.au itv.com ,japantimes.co.jp etc.,.

It has to be noted that the annotated news articles were in English. The data was annotated by a single person over a span of 7 hours.

A naive classifier which classifies all text nodes as content leads to an Fscore of **0.11**, a better baseline that uses words count (of a node) (logistic regression) as a feature leads to an Fscore of **0.562** for the content class. The code for this analysis can be found in the ipython notebook that is present with the data set.

4 Feature Engineering and Learning

A combination of features were used and analysed while implementing these classifiers. These features broadly fall into three categories: Language based features that analyse the content inside a text node, Location based features that measure the absolute or relative position of a node and appearance base features that drop hints about the appearance of a text node. These sets of features can be viewed as answers to the question where is the node located ? how does it appear ? what does it contain ?. Language and appearance features corresponding to the close neighbours (previous/next leaf node) were also used.

4.1 Location features

These features, as the name indicates, measure the position of a node from a fixed or a variable reference. Two features were to obtain this measurement was introduced under this , one is the relative distance between the possible title of a webnews article and the node, the other one being the absolute position of the node. To calculate the first feature one needs to locate the title of an article. We can find the title by computing the edit distance between the text that is present under the DOM tree's Title (not to be confused with the title class) or headline tag and the text node and identifying the text node (which is appears on the screen) with the lowest score as the Title of the webnews article. Once we ascertain the location of the title we can compute a node's relative position from the 'title text node'.

4.2 Appearance features

These features encode the appearance of a text node. We rely on semantic cues provided by html tags like h,b and the variable names used in a div to determine the appearance of the text. The appearance of a text node is obtained from the html tag that surrounds the text. The html tag and its properties relating to font/heading/header/footer are extracted.

4.3 Language features

These features are expected to capture the language details that distinguish content from boilerplate. Features that belongs to this category include functional word ratio, Word count, character count, isMonth, isdateregexp, isyear etc.,

4.4 Learning

4.4.1 Title Classifier

The goal of the title classifier is to figure out the best textnode that can be a title in a web news article. As discussed previously, one plausible candidate for this position is the one which has the least edit distance from the 'Title' or headline tag that is embedded in the html webpage. A text nodes html tag is also used as a feature. Hence, a simple classifier using these two features is proposed.

4.4.2 Publish Date Classifier

The goal of the Published Date classifier is to figure out the date (as it appears on the screen). Though there are meta tags that can be directly used to extract the published time they may not be reliable or they may not be present in a webpage. To handle this scenario, we introduce a machine learning based classifier that can extract the publish date of the article. Location features like a node's distance from the title etc., are used.

4.4.3 Content Classifier

The goal of the Content classifier is to figure out the main content (as it appears on the screen). We define the main content of a web news article to be a domain agnostic article related to the headline. Thus copyright and author information are excluded from the body. Locational features are currently not included in the current model.

5 Implementation

This section discusses briefly about the software implementation [17] that accompanies with this report. The software package consists of an extension that can be used to annotate data using the chrome web browser. Using this extension the user can select a region of text and classify it as

Method	Mean Fscore	std Fscore
RandomForest	0.584	0.038
LogisticRegression	0.365	0.030

Table 2: Date Classifier : Cross validated Accuracy on 270 examples

Method	Mean Fscore	std Fscore
RandomForest (100 examples)	0.542	0.031
LogisticRegression (100 examples)	0.642	0.020
RandomForest (270 examples)	0.668	0.038
LogisticRegression (270 examples)	0.697	0.002

Table 3: Content Classifier (appearance and language features only): 5 fold Cross validated Accuracy

Title or Body or Date. The extension collects the set of nodes that are selected and sends them to a dropbox location using a deprecated dropbox API. Text data is stored in the utf8 format.

The collected data is placed in the data subfolder and it is renamed to **FullDataset**. Inorder to use this annotated data , the user executes **prepare_data.sh 'featurename'**. This bash script internally calls **prepare.py** which aligns the output from the parser with the groundtruth annotations. The alignment function (**match()** in **prepare.py**) matches the every text value in the collected text node with the ground truth data. If multiple exact matches are found, all of them are considered to be valid classes except in the case of content where the entire node is considered to be boilerplate. There could be a failure to match the text nodes, and this is due to the parser's failure entirely. It is interesting to note that around $\frac{1}{10}^{th}$ of the web news items were not uniquely parse-able. After alinging, the parser generates an inorder traversed list of possible text nodes belonging to the webpage that needs to be classified. The aligned ground truth information is placed inside a temporary folder: **temp**.

At this point the bash script calls **helper.py** with the aligned folder and the featurename. Now, every text node along with the webpage's DOM tree is identified and passed to **extract_feature()** in **helper.py**. Here, **extract_feature()** creates the required feature instance (based on 'featurename') using **Feature.py**. This instance is used to calculate the set of features for every text node. This instance, on calling **extract_features** returns a dictionary of activate features which is converterd to a vector using sklearns **dictvectorizer**. At the end of this stage we obtain a Matrix where the rows denote a node and the columns denote the feature vector for that node. The first column in the generated matrix contain the classlabel (1 for Title, 2 for Content and 3 for Published_Date). The feature matrix, stored in **data/'featurename'/feature/train**, of a webpage can be identified using the fileid present in **data/train/field.csv** along with the **vectorizer**. This data is used to learn a machine learning model for the problem and these models are stored in **NewsExtractor/models/** once they are complete and renamed to ensure they are appropriately loaded during classification time.

During classification time, the classifier **Newspaper.py** loads these models and computes the title date and Content.

The software uses xpath expressions to navigate the DOM tree and extract features. It can support any parser as long as the parser can access DOM elements through xpath expressions. Currently,the software requires the webnews article to be a UTF8 encoded unicode.

6 Results and Discussion

Randomforest classifier was chosen as the best learning algorithm for the task. It is important to mention here that this an Fscore and recall score of 0.5 translates into roughly getting the date right once in two webnews articles and predicting only one extra node as a Date. This could be a good score given the scarcity of the training data because a date occurs only once in 400 nodes!

Sentence	assigned label	true label
. The story was updated on July 26 that year.	0	1
from Colorlines	0	1
, from an ongoing	0	1

Table 4: Content Classifier (appearance and language features only): 5 fold Cross validated Accuracy

The precision and recall scores for the logistic regression classifier was **0.804** and **0.607**.

The most informative features shown in the figure 2 are surprisingly local neighbourhood appearance features! This clearly indicates we could much better language features would help boost the recall. If we look at the kinds of errors the classifier makes, which are mostly recall errors, we expect a CRF to do better since A CRF takes the (net) effect of the content classifier on the neighbouring node into consideration.

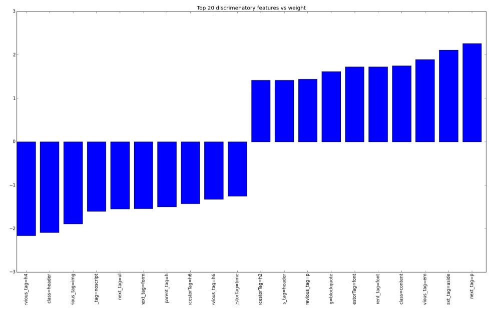


Figure 2: Content Classifier : Discriminatory Features

6.1 Future Work

This work can be extended to solve this problem in a semi supervised fashion. One straightforward method is to accept user supervision on data points that are near the decision boundary of the classifier. These datapoints could be classified for a better accuracy. The software might request the user for labelling specific points.

A more sophisticated structured prediction algorithm could be applied to the problem. To detect multiple articles.

References

[1] AFR Rahman, H Alam, and R Hartono. Understanding the flow of content in summarizing html documents.

[2] Brad Adelberg. Nodose&mdash;a tool for semi-automatically extracting structured and semistructured data from text documents. *SIGMOD Rec.*, 27(2):283–294, June 1998.

[3] Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Accordion summarization for end-game browsing on pdas and cellular phones.

[4] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. Dom-based content extraction of html documents. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 207–214, New York, NY, USA, 2003. ACM.

[5] Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from web documents. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 588–593. ACM, 2002.

[6] David Pinto, Michael Branstein, Ryan Coleman, W Bruce Croft, Matthew King, Wei Li, and Xing Wei. Quasm: a system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 46–55. ACM, 2002.

[7] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA, 2010. ACM.

[8] Thomas Gottron. Content code blurring: A new approach to content extraction. In *19th International Conference on Database and Expert Systems Application*, pages 29–33. IEEE, 2008.

[9] Tim Weninger and William H Hsu. Text extraction from the web via text-to-tag ratio. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, pages 23–28. IEEE, 2008.

[10] Tim Weninger, William H Hsu, and Jiawei Han. Cetr: content extraction via tag ratios. In *Proceedings of the 19th international conference on World wide web*, pages 971–980. ACM, 2010.

[11] Michal Marek. Web page cleaning with conditional random fields. 2007.

[12] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 580–591, New York, NY, USA, 2002. ACM.

[13] Liang Chen, Shaozhi Ye, and Xing Li. Template detection for large scale search engines. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 1094–1098, New York, NY, USA, 2006. ACM.

[14] Dandan Song, Fei Sun, and Lejian Liao. *Knowledge and Information Systems*, 42(1):75–96, 2015.

[15] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. Cleaneval: a competition for cleaning web pages.

[16] Matthew E Peters and Dan Lecocq. Content extraction using diverse feature sets. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 89–90. International World Wide Web Conferences Steering Committee, 2013.

[17] Gowtham Rangarajan Raman. Newsextraction. 2016.