Content Extraction Methods

Gowtham Rangarajan R
University of Massachusetts, Amherst
ranga@cs.umass.edu

Abstract

This report is a summary of the techniques and ideas that are closely related to content extraction in web articles. It discusses and evaluates few popular open source software that can be used to extract Title, Date and content from news articles. The report is divided in 4 sections. Section 1 briefly restates the problem . Section 2 deals with the techniques proposed in the literature to extract content discussing few of them in detail. Section 3 discusses the data set and evaluation methods that have been used in the literature. Section 4 talks about some of the popular open source libraries and evaluates them for the task.

1 Problem Statement

A Web-news article contains a lot more information besides the main 'content' such as ads, navigation links, comment sections, related videos, etc.,. The goal of the techniques discussed in this report is to classify text elements present in a webnews article as content or noise. However, few open source software provide provisions to extract title and date along with content. In such cases, they have been evaluated on title date and content.

Terminology: tags refer to html tags that occur, eg., $\langle p \rangle$, $\langle title \rangle$. Nodes refer to the nodes in a DOM tree. Token refers to a single word when the webpage is read as plain text, eg., $\langle div \rangle$, class='checkbox', rocket, etc., block refers to a html block eg., $\langle p \rangle$ This is new $\langle p/ \rangle$.

2 Prior Work

Content extraction from HTML documents has been well studied over the past two decades. They have evolved from using DOM tree matching algorithms to heuristic based machine learning methods.

The term content extraction were introduced early in the last decade by Rahman et al., [1] and since then numerous techniques have been developed to extract relevant information from a webpage. 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. Potter [5] is an interactive system that uses regular expressions to identify relevant content based on user input. [6] [7] porvides an excellent survey of the methods used in early 2000s.

During the later half of the decade, the research progressed towards using intra domain DOM tree template matching algorithms . These algorithms require 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 [8] [9]

were proposed. These heuristics remain relevant, even today, as features for machine learning techniques. During the first half of this decade there has been interest in treating the task as supervised machine learning problem. [10] [11] [12]. Kohlschutter et., al [10] introduced shallow text features such as average word length, captilization ratio link density to distinguish content from boilerplate. Weninger et., al [12] introduced CETR content which computes the ratio of number of tokens to that of tags and uses them as features. Advanced DSC (ADSC) [11] uses 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 [10] [11] or from plain-text html [**?**]and are learnt using K-means [12] or decision trees [10]. Most of these features have been benchmarked by Weninger. et., al [13] recently. Conditional Random Fields have also been used to clean up webpages [14].

There has been recent interest in using vision based methods [15] [16] where the DOM tree is clustered based on the visual content of partially rendered pages [17]. 2D crfs [18] have also been used to predict structured content directly from webpages.

However, recent popular open source software use simple supervised learning methods and hand engineered regular expressions to keep up with the modern day web.

This following paragraphs discuss some of the methods mentioned above in detail.

There were a significant number of heuristic based methods that were proposed during the last decade. These methods typically involve computing properties of the blocks/nodes of every element in the HTML web page, and classifying them as content based on a threshold.

Document slope curve considers a html page as plain text and [9] assigns 1s to recognized html tags that may correspond to non content and 0s to other word tokens. An optimum window length is chosen for the document and a windowed average at each tag is obtained. This heuristic is then used to classify a token based on a threshold. BTE identifies the largest region of text in a webpage with minimal tags as content. The Largest Tag Count feature considers the number of tags contained in each sub-tree. A large number of tags in a sub-tree typically indicates that a particular region of text has sophisticated structure and thus potentially richer in content than a sub-tree with a small number of tags. The change in the tag count across siblings is also recorded as a feature. Tag ratio is computed as the smoothed ratio of number of html tags to the number of words in a line in a html file. The ratio along with its derivative (against line-number) is used to separate content from non-content. Fanout feature is based on the number of immediate children a subtree has. The larger the fanout, the more likely the sub-tree is the immediate parent of all of the data objects. Text density computes the ratio between the number of words per block to that of the number of lines (assuming 80 characters per line).

[19] Mckeown et al. 's multigen system collects sentences together using sentence level similarity. A vector representation of features is obtained through various word level, syntactic features, that uses WORDNET data which is converted to a similarity measure using a logisitic regression. This computed similarity between every pair of sentences is clustered to estimate the sentences that discuss about the same news article.

The CLEANEVAL contest winner [14] used a linear chain CRF with the following features :- tags : p,a,u, img, header, class-bold, .class-italic, class-list, class-form; Content based features:- alpha-rel, num-rel, punct-rel, white-rel, char.other-rel, sentence.count, sentence.avg-length,sentence-begin, sentence-end, first-duplicate, duplicate-count, regexp.url, regexp.date, regexp.time ,bullet, div-group.word-ratio, td-group.word-ratio, position, document.word-count, document.sentence-count, document.block-count, document.max-div-group, document.max-td-group. These features were used to classify each text block into header, paragraph, list, continuation and noise.

Zhu et., al [18] proposes a 2d crf where every element in the html page is mapped on to a grid based on the position of the element in the rendered page. They then uses virterbi decoding to to train (perceptron) and infer the latent labels of the 2d grid.

[20] proposes Web page segmentation with a structured prediction approach. It formulates the segmentation task as a structured labeling problem on a transformed Web page segmentation graph (WPSgraph). WPS-graph models the candidate segmentation boundaries of a page and the dependency relation among the adjacent segmentation boundaries. Each labeling scheme on the WPS-graph corresponds to a possible segmentation of the page. The task of nding the optimal labeling of

the WPS-graph is transformed into a binary Integer Linear Programming problem, which considers the entire WPS-graph as a whole to conduct structured prediction with SVM.

3 Software

This section discusses the techniques used by popular open source software for content extraction. It concludes by evaluating these software for the given task.

Popular open source software, used for content extraction rely on a mix of regular expression and heuristic based techniques to extract relevant content. The list of popular open source software that are discussed here are

- BoilerPipe
- Goose
- Readability
- NewsPaper
- LibExtract

3.1 Boilerpipe

Boilerpipe [21], written in java, was introduced in the late 2009 and was actively worked upon till late 2011. It was a consequence of the work of Kohlschutter et., al [10] on extracting shallow features for content extraction. The primary purpose of this software is to separate out content from non content in a html webpage. Though the software provides fields to extract title and date, they are not promised to work effectively.

This extractor is heavily (around 10) based on heuristics such as number of words per block, link ratio avoiding regular expressions altogether. The process by which they arrive the thresholds for these heuristics is unclear and it is perhaps the hardest to tweak for the task at hand. Boilerpipe considers each block independently and thus doesn't exploit the tree structure of an html page.

This software has a python binding [22] which is used for evaluation in this report.

3.2 Goose

Goose [23] was first written in scala and later ported to python that is maintained separately, was introduced in the late 2009 and it is being actively developed. The software promises a good extractors for the date and Title field unlike Boilerpipe.

The core of the extractor relies only upon link density (ie.,ratio of number of links to number of words) score and stop word count. They compute and update these scores upto 3 levels from a DOM node and drop the html tree with high scores. They also identify boilerplate using regular expressions. Goose remains relatively silent about the rationale behind calculating these scores.

The Date and title extractor exploits the semantic cues inside a webpage. It also uses regexes to clean the most probable title.

3.3 Newspaper

This software [24], released recently, borrows most of its code from goose. The content extractor, that we are interested in, deviates negligibly from goose. However, There have been few enhancements to improve title and date prediction.These improvements come in the form of updated regexps.

3.4 LibExtract

This software [25], released recently, uses xpath expressions and lxml's html parser to navigate the DOM tree. It extracts 5 most frequently occurring tags, excluding the script and style tag, in a html file and assigns them as content.

Method	Score(Title)	Score(Content)
Boilerpipe	-	0.38
Goose	0.92	0.69
Readability	0.90	0.63
Newspaper	0.91	0.85
LibExtract	-	0.27
NewsExtractor(previous Report)	0.96	0.78

Table 1: Title's Exact accuracy On 102 examples

3.5 Readability

This software [26], written in ruby and recently ported to python, appears to share a common core with Goose. Unlikely candidates, obtained by studying the candidate's html tag, are removed from the DOM tree. Remaining candidates are filtered using uses link density and word count scores. These scores of DOM nodes are bubbled up to 3 levels and the best candidate is calculated. The process and the code invovled is strikingly similar to that of goose, One notable difference between them is that stopword count heuristic is not used here. These scores appear to be similar to that of goose! The math behind calculating these scores are unclear here too.

4 Public Datasets and Metrics for Evaluation

There are few publicly available data sets [27] [10] [28] that annotate content from the web news article. Some data-sets [27] [10] contain markup information while some dont [28]. Typically, the ground truth data with the markup information, if present, is matched with the text elements in the web page using a variation of the longest common sub sequence problem. Unfortunately, there isn't a standard tool for this match that the author is aware of.

The first notable effort to standardize the content extraction problem, CLEANEVAL [27] (2007), an activity of ACL, was a shared task and competitive evaluation on the topic of cleaning arbitrary web pages, with the goal of preparing web data for use as a corpus, for linguistic and language technology research and development. This contest resulted in the acceptance of the edit distance metric to evaluate content extraction methods.

The content annotated in the dataset includes regions of webpage that doesn't necessarily contain information regarding the newsarticle, besides there is no distinction between title/date and content. Hence these datasets weren't used to train the classifiers which were proposed in the previous report.

4.1 Evaluation

CLEANEVAL [29] shared task proposed a metric to evaluate these tasks based on levinshetien distance. In this method the edit distance between ground truth and the article to be compared are found and the ratio between that and the maximum number of words between them is used. However, this metric is computationally expensive to evaluate forcing recent literature to opt for mean F1 scores to evaluate their models in large corpus. Here, each article is broken into word tokens and an Fscore using these tokens is computed for the document. The mean Fscore over the test set is reported. In this report, the evaluations are done based on the mean Fscore proposal [30].

The software mentioned above were used to collect content and Title from a dataset collect for this task. The results are in table 1

As it can be seen with better regular expression matching Newspaper does better than goose and NewsExtractor. However, NewsExtractor does better in extracting titles.

References

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

[2] Brad Adelberg. Nodose—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] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390, 2001.

[6] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.

[7] Alberto HF Laender, Berthier A Ribeiro-Neto, Altigran S da Silva, and Juliana S Teixeira. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2):84–93, 2002.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

[18] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2d conditional random fields for web information extraction. In *Proceedings of the 22nd international conference on Machine learning*, pages 1044–1051. ACM, 2005.

[19] Kathleen McKeown, Vasileios Hatzivassiloglou, Regina Barzilay, Barry Schiffman, David Evans, and Simone Teufel. Columbia multi-document summarization: Approach and evaluation. 2001.

[20] Lidong Bing, Rui Guo, Wai Lam, Zheng-Yu Niu, and Haifeng Wang. Web page segmentation with structured prediction and its application in web page classification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 767–776. ACM, 2014.

[21] OpenSource. Java boilerpipe. https://code.google.com/archive/p/boilerpipe/, 2007–2004.

[22] OpenSource. Python bindings for boilerpipe. https://github.com/ptwobrussell/python-boilerpipe/, 2007–2004.

[23] OpenSource. Goose. https://github.com/grangier/python-goose, 2000–2004.

[24] OpenSource. Newspaper3k. https://github.com/codelucas/newspaper, 2000–2016.

[25] OpenSource. liextract. https://github.com/datalib/libextract, 2000–2016.

[26] OpenSource. readability-redux. https://github.com/MHordecki/readability-redux/, 2000–2016.

[27] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. Cleaneval: a competition for cleaning webpages.

[28] Matthew E Peters and Dan Lecocq. Content extraction using diverse feature sets.

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

[30] CleabEvalSharedTask2007. Cleaneval scorer. http://cleaneval.sigwac.org.uk/, 2007–2004.