Noise Block Cleaning and Main Content Block Extraction from Dynamic Web Page

Pan Ei San, Nilar Aye
University of computer Studies, Yangon
paneisan1985@gmail.com

Abstract

Web Information Extraction systems becomes more complex and time-consuming. Web page contains many informative blocks and noise blocks. Noise blocks are navigational elements, templates and advertisements that are not the main content blocks of the web page; it can be defined noisy blocks or boilerplate text. This boilerplate text typically is not related to the main content, may deteriorate search precision and thus needs to be detected properly. This paper proposes a Web Page cleaning and main content block extraction approach and purposes of improving the accuracy and efficiency of web content mining. The system uses structural features and the shallow text features as such as number of words, link density, and average word length can be used to classify the main content or boilerplate text from the web page. And then the system extracts main content block using three parameters such as Title keyword, Keyword Frequency based Block selection and position features. The relevant content blocks are identified as the high important level by similarity of block contents to other blocks. Experiments show that Web Page cleaning based on shallow features lead to more accurate and efficient classification results for boilerplate or other content than existing approaches.

Keywords: Boilerplate Detection, Decision Tree, Shallow Text features, Web Content Mining

1. Introduction

The rapid expansion of the Internet has made the WWW a popular place for disseminating and collecting information. Extracting useful information from Web pages thus becomes an important task. Usually, apart from the main content blocks, web pages usually have such blocks as navigation bars, copyright and privacy notices, relevant hyperlinks, and advertisements, which are called noisy blocks. Although such information items are functionally useful for human viewers and necessary for the Web site owners, they often hamper Web page clustering, classification, information retrieval and information extraction. Therefore how to identify and separate main content blocks from noisy blocks is a crucial issue. A number of approaches have been introduced to automatic this distinction, using a combination of heuristic segmentation and features. The most popular features used for boilerplate detection: number of words and link density - leads to a simple classification model that achieves competitive accuracy. We focus on detecting and eliminating local noises in Web pages to improve the performance of Web mining. For removing local noise or boilerplate, other systems use segmentation method that is based on different level of granularity. And then other uses VIPS (vision based method) that is computationally expensive and requires the external resources. The few existing work on web page cleaning delete noise blocks with exact matching contents but are weak at detecting new duplicate blocks, characterized by items like navigation bars.

Elimination of noisy and irrelevant contents from web pages has many applications, including web page classification, clustering, web featuring, proper indexing of search engines, efficient focused crawlers, cell phones and PDA browsing, speech rendering for the visually impaired, improving the quality of search results and text summarization [7]. In our system, classifying and mining noise-free web pages will improve on accuracy of search results as well as search speed and may benefits web page organization applications (e.g., keyword-based search engines and web categorization applications).

The main contributions in our work are:

- The system reduces the complexity of Information extraction system that removing the "noise" or "irrelevant" block from web page is performed as intermediated step based on shallow text features and Decision Tree method to classify the main content or boilerplate.
- A block important model is proposed to assign important weight to different regions in the web page using shallow text features and position features.

The remainder of this paper is organized as follows. The next section presents the related work. Section 3 describes Background theory of our system and Section 4 describes our proposed system and detail explains. Section 5 describes evaluation using remarkable precision, recall and F-score in our system. Finally, we conclude the paper by suggesting possible directions for future work to improve the proposed system in Section 6.

2. Related Work

Jan Pomikalek, [3] addressed two important problems that relevant to creation of large Web

Corpora- Cleaning boilerplate and removing duplicate and near-duplicate content. The part of the work devoted to boilerplate cleaning proposes a novel heuristic based algorithm called jusText. The algorithm processes the input in two stages: first of which classifies each text block on dependently based on its length, number of links and proportion of function words. The second stage then uses the context (the class of surrounding blocks) to assign classes to those blocks which could not be reliably classified in the first stage. A novel algorithm for removing near-duplicate content has been created, which only needs to store the duplicate n-grams in memory(rather than an inverted index on the duplicate n-grams).This reduced memory requirements and increases scalability of the method. To save even more memory, a memory efficient data structure Judy array is employed. M.Lavanya and Dr.M.Usha Rani [5] approved an approach to vision-based deep web data extraction is proposed for web document clustering. The proposed approach comprises of two phases: 1) Vision-based web data extraction and 2) web document clustering. In phase 1, the web page information is segmented into various chunks. From which, surplus noise and duplicate chunks are removed using three parameters, such as hyperlink percentage, noise score and cosine similarity. Finally, the extracted keywords are subjected to web document clustering using Fuzzy c-means clustering (FCM). This proposed system's experimental results showed that the proposed VDEC method can achieve stable and good results for both datasets (GDS and SDS). Christian Kohlschutter [1] presented a large-scale aggregate analysis of textual Web content, corroborating statistical laws from the field of Quantitative Linguistics and analyzes the idiosyncrasy of template content compared to regular “full text” content and derives a simple yet suitable quantitative model.

The annotators were instructed as follows in Clean Eval Standards [4] : (1) removing all HTML/JavaScript code and “boilerplate” (headers, copyright notices, link lists, materials repeated across most pages of a site, etc.); (2) adding a basic encoding of the structure of the page using a minimal set of symbols to mark the beginning of headers, paragraphs and list elements. Ruihua Song [7] investigated how to find a model to automatically assign importance values to blocks in a web page. They define the block importance estimation as a learning problem. First, we use the VIPS (VIsion-based Page Segmentation) algorithm to partition a web page into semantic blocks with a hierarchy structure. Then spatial features (such as position, size) and content features (such as the number of images and links) are extracted to construct a feature vector for each block. Based on these features, learning algorithms, such as SVM and neural network are applied to train various block importance models. In this approach’s experiments, the best model can achieve the performance with Micro-F1 79% and Micro-Accuracy 85.9%, which is quite close to a person’s. Roberto Panerai Velloso [6] proposed a novel and fully automatic algorithm that uses a tag path sequence (TPS) representation of the web page. The TPS consists of a sequence of symbols (string), each one representing a different tag path. The proposed techniques searches for positions in the TPS where it is possible to split it in two regions here each region’s alphabet do not intersect, which means that they have completely different sets of tag paths and, thus, are different regions. The results show that the algorithm is very effective in identifying the main content block of several major websites, and improves the precision of the extraction step by removing irrelevant results. By weighting terms by their block importance they significantly improve accuracy over the baseline BM25. They show that densitometric features, which can be computed efficiently online, without resorting to global frequencies, also significantly improve retrieval accuracy. Dheeraj Raja opal [2] proposed an approach for effective multi-word common sense expression extraction from unrestricted English text, in addition to a semantic similarity detection technique allowing additional matches to be found for specific concepts not already present in knowledge bases.

3. Background Theory

3.1. Web Page Features

Many features that can be used for the classification of Web page segments. It is generally accepted that the combination of several features can be used to identify text fragments as headline, full text, enumeration, navigation, disclaimer notice etc., which can then be separated into content and boilerplate text. Features may be extracted at four different levels: Individual text blocks (elements), the complete HTML document (a sequence of one or more text blocks plus structural information), the rendered document image (the visual representation as in a Web browser) and the complete Web site (i.e., the collection of documents which share a common layout and wording). While the former two levels can apparently be examined for each document locally, the latter two require external information, such as images and CSS (Cascading Style Sheet) definition for the rendering process and, in order to statistically determine site-level templates and boilerplate text, a sufficiently large number of pages from the same website.

Using features from the two external levels may be highly beneficial to the classification accuracy if the corresponding data is available. However, there are two major drawbacks. First, rendering pages for classification is a computational expensive operation. Second, template statistics need to be learned

separately for each web site, they usually cannot be re-used for another website layout. Using this feature we can identify phrases commonly used in boilerplate [3].

3.1.1. Structural Features

Many approaches for Web page segmentation and intra-document text classification utilize structural features in Web pages, such as individual HTML tags (headline, paragraph, anchor text link, image, etc.) or sequences/nested subtrees of HTML tags as well as the presence of particular CSS classes and styles. Of note, the more CSS is used, the less important the semantics of an HTML tag becomes it is perfectly legal to only use DIV tags and describe the "semantics" of a particular division using style-sheet classes. As we want to avoid site specific signals (which may lead to overfitting to a particular data set or domain) as well as a costly rendering of pages, we only examine the following structural features: The presence of a particular headline tag (H1, H2, H3, H4, H5, H6), a paragraph tag (P), a division tag (DIV) and the anchor text tag (A) as an HTML element that closes a particular text block [3].

3.1.2 Shallow Text Features

Because boilerplate detection does not inspect text at the topical level but rather at the functional level, we do not consider the bag of words as classification features. An evaluation at token-level may provide skewed results that describe a particular domain only. Instead, shallow text features can be examined at a higher, domain and language independent level, which have been discussed in the field of Quantitative Linguistics: Average word length (in our definition words are white-space delimited character sequences which at least contain one letter or digit), average sentence length (the sentence boundaries are identified by a simple pattern-based heuristic checking for the presence of full stops, question or exclamation marks as well as semi-colons) and the absolute number of words.

Another important source for the classification task is the local context, i.e., the absolute and relative position of a text block in the document. If the segmentation granularity is high, it is likely that full-text is followed by full-text and template is followed by template. Moreover, when there is a significant amount of boilerplate text, the main content usually is surrounded by boilerplate (header, footer, left-navigation, right-navigation etc.), not vice versa (i.e., even if the very last text block contains a sentence, if it is a copy-right or disclaimer notice, it is regarded boilerplate).

A few heuristic features are examined: the absolute number of words that either start with an uppercase letter or are completely upper-case as well as the ratio of these words compared to the total number of words and the ratio of full stops to the overall number of words, the number of date/time-related tokens and the number of vertical bars "|" (these characters can sometimes be found in navigational boilerplate text). Moreover, they also compute the link density (called anchor percentage) as the number of tokens within an A tag divided by the total number of tokens in the block; for this computation they do not regard the A tag as a block separator [3].

3.1.3. Densitometric Features

Besides the link density measure, the text density of each particular block is evaluated. Text density was derived from the pixel-based text density of Computer Vision-based approaches and transformed to character level. Basically, the text density is calculated by dividing the number of token by the number of lines.

3.2. Clean Eval Competition Gold Standard

The Clean Eval guidelines instruct to remove boilerplate types such as:

1. Navigation
2. Lists of links
3. Copyright notices
4. Template materials, such as headers and footers
5. Advertisements
6. Web-spam, such as automated postings by spammers
7. Forms
8. Duplicate material, such as quotes of the previous posts in a discussion forum

The format of the gold standard files is a plain text with simple markup (headers, lists, paragraphs) where the boilerplate parts are removed. Number of words and link density lead to a simple classification model achieves competitive accuracy. The Clean Eval's specification of boilerplate is more conservative and may be appropriate for information retrieval, but less so for creating Web corpora. While most boilerplate types can be identified based on structural features (surrounding HTML mark-up) and shallow text features (such as the number of tokens in a text block), a fundamentally different approach is required for detecting both Web-spam and duplicate content [4].

3.3. Classification of the boilerplate or Other Content using Decision Tree Algorithm

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is one of the predictive modeling approaches used in statistics, data mining and machine learning. More

descriptive names for such tree models are classification trees or regression trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. It performs well with large datasets. Large amounts of data can be analyzed using standard computing resources in reasonable time.

4. Proposed System

In our proposed system, there are two phases. The first phase is to define boilerplate or not using three steps: (1) segmentation step, (2) content free classification and (3) context sensitive classification. The second phase is to extract main content after removing boilerplate. Finally the decision tree (C4.8) is used to calculate the accuracy and performance of our proposed system.

4.1. Architecture of system components

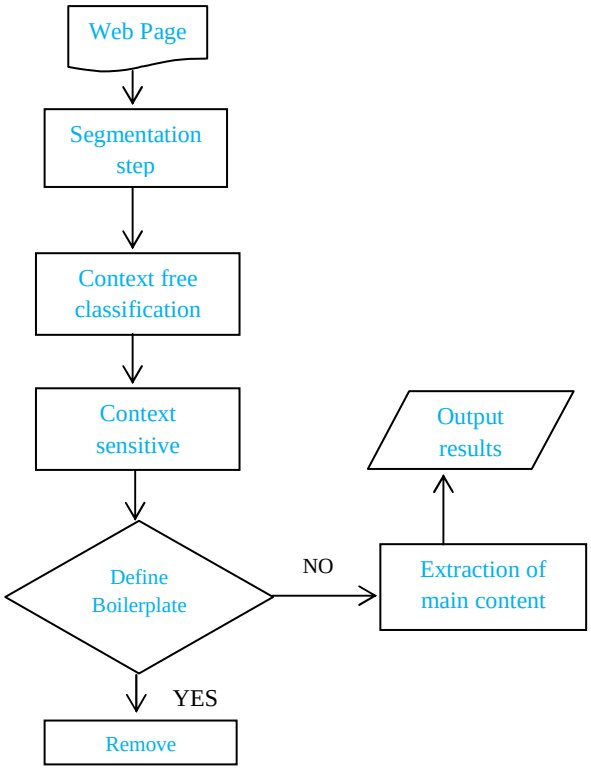

Figure 1: Proposed System of Content Management System

4.1.1. Preprocessing or segmentation step

In the preprocessing stage, the system defines as boilerplate and removes these tags such as<style>, <script> and <select> tag and copyright symbol. The full list of the used block-level tags includes: BLOCKQUOTE, CAPTION, CENTER,COL, COLGROUP, DD, DIV, DL, DT, FIELDSET, FORM, H1, H2, H3, H4, H5, H6, LEGEND,LI, OPTGROUP, OPTION, P, PRE, TABLE, TD, TEXTAREA, TFOOT, TH, THEAD, TR, UL.A sequence of two or more BR tags also separates blocks. The web pages are segmented into atomic text blocks, which are then annotated with features such as number of words, link density, average word length, the proportion of capital letters and the proportion of full stops and on this basis classified into content or boilerplate. Atomic text block are sequences of character data which are separated by one or more HTML tags, except for *A* tags-in order to compute the link density. Our proposed system considers that long blocks and some short blocks can be classified with very high confidence. All the other short blocks can then be classified by looking at the surrounding blocks. Several observations can be made about the blocks created in this way:

1. Short blocks which contain a link are almost always boilerplate.
2. Any blocks which contain many links are almost always boilerplate.
3. Long blocks which contain grammatical text are almost always good whereas all other long blocks are almost always boilerplate.
4. Both good (main content) and boilerplate blocks tend to create clusters, i.e. a boilerplate block is usually surrounded by other boilerplate blocks and vice versa.

4.1.2. Define Boilerplate or not

4.1.2.1 Context-free classification

After the segmentation and preprocessing, context free classification is executed which assigns each block to one of four classes:

- Bad-boilerplate blocks
- Good- main content block
- Short- too short to more reliable decision about the class
- Near-good- somewhere in –between short and good

The classification is done by the algorithm 1. The token count is the number of space-separated items and the link density is defined as the proportion of tokens inside <a> tags. The stop words density is simply the proportion of stop list words. The algorithm1 takes two integers (LENGTH_LOW, LENGTH_HIGH) and three floating point numbers (MAX_LINK_DENSITY, STOPWORD_HIGH, STOPWORD_LOW) as parameter. The former two set the thresholds for dividing the blocks by tokens count into short, medium_size and long. The latter two divide the blocks by the stop word density into low, median and high.

```
if link_density> MAX_LINK_DENSITY;
return 'bad'
#short block
if token_count< LENGTH_LOW;
if link_density>0;
return 'bad'
else:
return 'short'
#medium and long block
if stopwords_density> STOPWORD_HIGH;
if token_count> LENGTH_HIGH;
return 'good'
else:
return 'near_good'
if stopwords_density> STOPWORD_LOW;
return 'near_good'
else:
return 'bad'
```

Algorithm 1: Classification of Boilerplate or main content

4.1.2.2 Context-sensitive classification

The goal of the context-sensitive part of the algorithm is to re-classify the short and near-good block as either as good or bad serve as base stones in this stage. The pre-classified blocks can be viewed as sequences of short and near-good blocks delimited with good and bad blocks. Each such sequence can be surrounded by two good blocks, two bad blocks or by a good block at one side and a bad block at one side. The former two cases are handled easily. All blocks in the sequence are classified as good or bad respectively. In the latter case, a near-good block closest to the bad block serves as a delimiter of the good and bad area. All blocks between the bad block and the near-good block are classified as bad. All the others are classified as good. If all the blocks in the sequence are short (there is no near-good block) they are all classified as bad.

The idea behind the context-sensitive classification is that boilerplate blocks are typically surrounded by other boilerplate blocks and vice versa. The near-good blocks usually contain useful corpus data if they occur close to good blocks. The short blocks are typically only useful if they are surrounded by good blocks from both sides. In the description of the context-sensitive classification, one special case has been intentionally omitted in order to keep it reasonably simple. A sequence of short and near-good blocks may as well occur at the beginning or at the end of the document. This case is handled as if the edges of the documents were bad blocks as the main content is typically located in the middle of the document and the boilerplate near the borders.

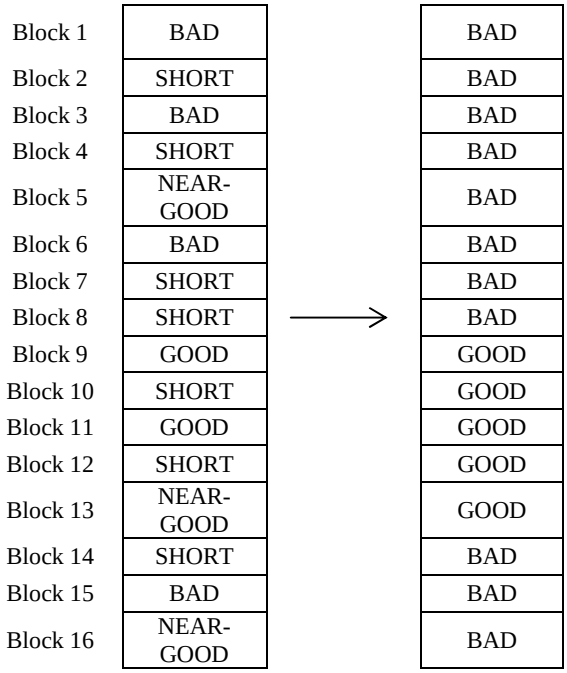

Figure2: Example of context-sensitive classification

In the following example, we will issue that Block 1 is previous block, Block 2 is current block and Block 3 is next block. The system extracts the link density, text density and word count features from these block. And then the system uses the context sensitive classification to change 4-classes to 2-classes. For example, in our classification, the system defines different linkDensity threshold (0.3 and 0.5) and stopwords_Density threshold (4, 9, 10, and 11) on each level to get more validate accuracy. Here the linkDensity value is higher, the block is more boilerplate. And the text density value is higher; the block is more content block. The following example describes how to classify the boilerplate or content by using decision tree.

Example of classifying the content or boilerplate

```
If (Curr-linkDensity<=0.3)
  If ( Prev-LinkDensity<=0.5)
    If(Curr-textDensity<=9)
      If ( Next-textDensity<=10)
         Prev-textDensity<=4; BOILERPLATE/BAD
         Prev-textDensity>4; CONTENT/GOOD
       Else(Next-textDensity>10);CONTENT/GOOD
   Else( Curr-textDensity>9)
        next-textDensity=0; BOILERPLATE/BAD
         next-textDensity>0; CONTENT/GOOD
  Else(Prev-LinkDensity>0.5)
         next-textDensity<=11; BOILERPLATE/BAD
         next-textDensity>11; CONTENT/GOOD
Else(Curr-linkDensity>0.3);BOILERPLATE/BAD
```

The system compared a number of structural and textual features on a dataset of 621 manually annotated news articles from 408 web sites. It used the information gain measure in order to identify the features with the highest ability to distinguish between main content and boilerplate. Simple features, such as number of words, link density, average word length, the proportion of capital letters and the proportion of full stops produced very high scores.

4.1.3. Exaction Important Block

4.1.3.1. Removing Duplicate Block Using Cosine Similarity

In this step the system calculates the block importance based on the level of block similarity, block position and link percentage. Here, the system reduces the duplicate block using equation (2). Cosine Similarity is one of the most popular similarity measures applied to text documents. Given two blocks B1 and B2, their cosine similarity is

$$\text{Cosine Similarity, SIM } B1, B2 = \frac{|B1.B2|}{|B1| * |B2|} \quad (2)$$

Where B1, B2 are weight of keywords in block B1 and B2.

4.1.3.2. Extraction Main Content Block

In the previous step, the system obtained a set of blocks after removing the noise blocks and duplicate blocks in a web page. After extracting content blocks, the system extracts essential keywords for the calculation of each Block Importance. In our system, the system has concentrated on the three parameters: title word relevancy, keyword frequency based block selection and position features which are very significant. Each parameter has its own significance for calculating sub-block weightage. The following equation computes sub-block weightage of all noiseless blocks.

$$B_W = \alpha T_k + \beta K_f + \gamma PF_s \quad (3)$$

Where γ,β and α are constants.

For each noiseless block, the system has to calculate these unknown parameters. The representation of each parameter is as follows:

(i) *Title Keyword* (T_k): Primarily, a web page title is the name or title of a Web site or a Web page. If the more number of title words are there in the particular block, the more it is important. This parameter of Title Keyword means the percentage of all the title keywords present in a block. It is computed using following equation.

Title word Relevancy,

$$T_k = 1 - \left[\frac{m_k}{\left[m_k + \sum_{i=1}^{m_k} F(m_k)^{(i)}\right]}\right] \quad (4)$$

Where, m_k=Number of Title Keywords

T_k=Title word relevancy

$F(m_k)^{(i)}$=Frequency of the title keyword n_t in a block

(ii) *Keyword Frequency based Block selection* (K_f): Basically, Keyword frequency is the number of times the keyword phrase appears on a deep Web page chunk relative to the total number of words on the web page. In our system, the top-K keywords of each and every block were selected and then their frequencies were calculated. The parameter keyword frequency based block selection is calculated for all sub-blocks and is computed using the following equation.

Keyword Frequency based Block selection

$$K_f = \sum_{k=1}^{K} \frac{f_k}{N} \quad (5)$$

Where, f_k =Frequency of Top ten keywords

K_f= Keyword Frequency based block selection

k=Number of Top-K Keywords

(iii) *Position Features (PF_s)*: Generally, these blocks are always centered horizontally and for calculating, the system calculates the ratio (T) of the size of the block to the size of whole web page instead of the actual size. In our experiments, the threshold of the ratio is set at 0.7, that is, if the ratio of the horizontally centered region is greater than or equal to 0.7, then the region is recognized as the data region. The system need to calculate the position feature in order to extract the important block from all blocks and is computed using the following equation.

$$PF_s = \begin{cases} 1 & 0.7 \geq T \\ 0 & \text{Otherwise} \end{cases}$$

Where,

$$T = \frac{Number\ of\ keywords\ in\ Data\ Region\ block}{Number\ of\ Keywords\ in\ Whole\ web\ page} \quad (6)$$

Thus, after calculating the values of T_k, K_f and PF_s by substituting the above mentioned equation. By substituting the values of T_k, K_f and PF_s in eq (3), the system obtains the sub-block weightage.

4.1.3.3. Block Weightage for Main Block

We have obtained sub-block weightage of all noiseless blocks from the above process. Then, the main block weightage is selected from the following equation.

$$B_i = \sum_{i=1}^{n} \propto B_W^{(i)} \quad (7)$$

Where, B_i=Main Block weightage, $\propto$ is constant and $B_W^{(i)}$is the i^{th} sub-block weightage of main-block. Thus, finally the system obtains a set of important blocks and extracts the keywords from the above obtained important blocks for effective web document information retrieval. Finally, the system displays the main content block to the user.

5. Evaluation

The class distribution is strongly dominated by Boilerplate and main content. We examine the per-feature information gain and evaluate machine learning classifiers based on Decision Trees as well as Support Vector Machines (SMO). We measure classification accuracy by Precision, Recall, F-Score; all scores are normalized based on the number of words in a block, i.e., large blocks are weighted higher than small blocks. The information gain is defined as the change in information entropy from the prior state (C) to a state that takes some information as given (C|A): IG(C, A) =H(C)-H(C | A) and helps identifying the most significant features. Generally, very simple features like Relative Position, Average Word Length and Number of Words of the current block appear to be strong indicators for class membership. We will test the classification accuracy of these highly ranked features separately. Interestingly, the text-flow capturing variants of these features (Number of Words Quotient, Text Density Quotient, Relative Position), which relate the value of the current block to the previous one, provide the highest information gain, indicating that the intra-document context plays an important role; this is also substantiated by the classification results.

The classification baseline is the ZeroR classifier, which in our case always predicts boilerplate and content. Due to the class weights this results in a precision of less than 50%. The 1R classifier determines the feature with the least error rate and partitions the corresponding numeric values to derive simple classification rules. 1R over all features resulted in a simple rule with an acceptable accuracy.

We get promising results from the C4.8-based decision trees, in order to avoid overfitting, the algorithm has been configured to only consider leaves matching at least 1000 instances. In our system, we analyse the classification features such as the link density, stop words density and number of tokens. The goal of the analysis is to confirm the intuition that all these features can be effectively used for distinguishing between the boilerplate and main content. Here, for link density, the blocks with the value above the threshold are classified as boilerplate, the other as main content.

For stop words density and number of tokens, the values below the threshold indicate boilerplate. We get a remarkable precision, recall and F-score of 95% for the 2-class (content and boilerplate) based on considering these shallow text features and used the following equations (8, 9 and 10).

$$precision = \frac{categories\ found\ and\ corrects}{total\ categories\ found} \quad (8)$$

$$recall = \frac{categories\ found\ and\ corrects}{total\ categories\ correct} \quad (9)$$

$$F\ score = \frac{2 * precision * recall}{precision + recall} \quad (10)$$

Table1: Classification Accuracies for the Google News Collection (2 class)

Algorithm	Precision	Recall	F-score
Zero R (baseline, predict content)	40.7	63.8	49.7
Only Average Word length	80.2	77.0	77.5
Only link density	88.5	87.8	87.4
1R: Text Density	87.8	87.9	87.9
C4.8 Link Density	91.1	91.1	91.0
C4.8 Number of words	91.1	90.8	90.9
C4.8 Number of words + Link Density	92.2	92.2	92.2
C4.8 Text Density+ Link Density	93.9	93.8	93.9
C4.8 our proposed shallow text features	95.1	95.0	95.1

6. Conclusion

In this paper, two important problems have been addressed the relevant to creation of large Web corpora cleaning boilerplate and removing duplicate. The system processes the input in two stages first of which classifies each text block independently based on its length, number of links and proportion of function words. In the second stage, the system extracts the main content which could be exacted the similarity result for user's query. It has a positive impact on fragmentation of the cleaned documents and prevents the texts taken out of content to be sent to output. To implement our proposed method, we developed an information extraction system capable of parsing a large corpus of web page, to describe accurately information retrieval result. The future work may also be extended for the specific content search. Webpage classification can help improve the quality of web search and improve SEO skill.

References

[1] Christian Kohlschütter, L3S/Leibniz Universität Hannover Appelstr.9a, 30167 Hannover, "A Densitometric Analysis of Web Template Content", Copyright: the author/owner(s).WWW2009, April 20–24, 2009, Madrid, Spain. ACM978-1-60558-487-4/09/04.Germany.

[2] Dheeraj Rajagopal, Eril Cambria and Daniel Olsher, National University of Singapore," A Graph-Based

Approach to Commonsense Concept Extraction and Semantic Similarity Detection", The International World Wide Web Conference Committee(IW3C2).WWW 2013 Companion, May 13–17,2013,Rio de Janeiro,Brazil.ACM978-1-4503-2038-2/13/05.

[3] Jan Pomikalek, "Removing Boilerplate and Duplicate content from Web Corpora", 2011.

[4] Marco Baroni, Francis Chantree, Adam Kilgarriff and Serge Sharoff, University of Trento,Lexical Computing Ltd,University of Leeds: "CleanEval:a competition for cleaning webpages",WAC3,the third Web-as-Corpus workshop

[5] M.Lavanya and Dr.M.Usha Rani, Sree Visyanikethan Engineering college, "Vision-Based Deep Web Data Extraction for Web Document Clustering" , Volume 12 Issue 5 Version 1.0 March 2012.Type:Double Blind Peer Reviewed International Research Journal. Publisher: Global Journals Inc. (USA).

[6] Roberto Panerai Velloso,Carina F.Dorneles, "Automatic Web Page Segmentation and Noise Removal for Structured Extraction Using Tag Path Sequences", Universidade Federal de Santa Catarina, Brazil, Journal of Information and Data Management,Vol.4,No.3,October 2013,Pages 173–187.

[7] Ruihua Song, Haifeng Liu, Ji-Rong Wen and Wei-Ying Ma: "Learning Block Importance Models for Web Pages", 2004, May 17-22, 2004, New York, NY USA.ACM.