EXTRACTING THE MAIN CONTENT FROM WEB DOCUMENTS

By

Samuel Louvan

0641162

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
EINDHOVEN UNIVERSITY OF TECHNOLOGY
EINDHOVEN
AUGUST 2009

Table of Contents

Abstract

The rapid growth of World Wide Web has been tremendous in recent years. With the large amount of information on the Internet, web pages have been the potential source of information retrieval and data mining technology such as commercial search engines, web mining applications. However, the web page as the main source of data consists of many parts which are not equally important. Besides the main contents, a web page also comprises of noisy parts that can degrade the performance of information retrieval applications. Therefore, a method to identify and extract main content is needed to alleviate this problem.

In this thesis, we address the problem of extracting the main content from web documents. Numerous approaches to do this task exist. Most of the previous approaches used heuristic rule sets to locate the main content. Our contribution in this work is mainly the development of web content extraction module which uses a hybrid approach that consist of *machine learning* and our own developed heuristic approaches namely *Largest Block String*, *String Length Smoothing*, and *Table Pattern*. Our work differs in the following ways: the operational settings of the content extraction module, the features used, the dataset type, the type of heuristic and the evaluation method.

According to our experiments, the combination of *machine learning* and *heuristic approach* gives encouraging result and it is a competitive content extraction method compared to the current state of the art web content extraction methods.

Acknowledgements

I would like to thank Mykola Pechenizkiy, my supervisor, for his many suggestions and guidance during this master thesis work. I am also thankful to all the people in Teezir especially Arthur van Bunningen, Thijs Westerveld, and Stefan de Bruijn for their valuable inputs and motivational feedback that encourage me to always do better in this work. Also thanks to Paul de Bra, Toon Calders for their valuable comments and suggestions on my research.

I want to thank Thomas Gottron for his kindness to share his evaluation dataset which is very useful in the experiments of this work.

I should also mention the TU/e-ASML Scholarship which was awarded to me for the period 2007–2009 was crucial to the successful completion of this work.

Finally, I would like to express my gratitude to my family: my mother Essy Lie and my sister Darlene for their constant support throughout my life. Last but not least, I would like to thank my girlfriend, Hani, who gives extra motivations and strengths during the long weeks of writing and re-writing this report.

Eindhoven Samuel Louvan
August, 2009

Chapter 1

Introduction

1.1 Motivation from Business Perspective

Since its birth in the beginning of the 1990s, the World Wide Web (WWW) has been undergoing remarkable growth. Originated as a hypertext system for accessing many forms of documentation at CERN, the WWW rapidly grow as it is accessible for public use through the web browser. Along with its tremendous growth, the web has been experiencing many changes; one of them is related to how its content is presented to the user. Typically, a modern web document comprises of different kinds of content. As illustrated in Figure 1.1, a news page, for instance, besides the article posting as the *main content* it also contains other *noisy contents* such as user comments, navigational menus, headers, footers, links to other news page, advertisements, copyright notices, privacy policies which scatter over the page. Considering the fact that a web document contains various forms of contents, it influences the way human browses the document. When browsing a particular web document, most of the time users typically focus on the main content and ignore the additional contents. For human, this behavior can be done relatively fast and accurate because they can

use their knowledge, visual representation and layout of the web pages to distinguish the main content from other parts. In the other hand, since computer software is not as intelligent as human to distinguish between the *main content* and the *noisy content*, this becomes the challenge for commercial search engines, web miners and other kinds of applications that use web document as a data source. A search engine, for instance, typically indexes the whole text of a web page. As a result, the noisy contents which is useless information remains in the index. The presence of noisy contents may degrade the performance of such Information Retrieval applications for example the quality of the search result, accuracy of information extraction, and the size of the index.

In order to alleviate this problem an approach to *extract only the main content* from web documents during data acquisition (e.g. crawling process) is needed. This task is needed to clean the web document from noisy contents. To the best of our knowledge, there is no commonly agreed terms which describe this task. However, some [1] [14] describe this task as a *web content extraction* task.

1.2 Operational Settings

The web content extraction operates on the data acquisition phase of the Information Retrieval system. As for problem domains, in this thesis we select three domains for web content extraction namely *news*, *blog*, and *forum* websites.

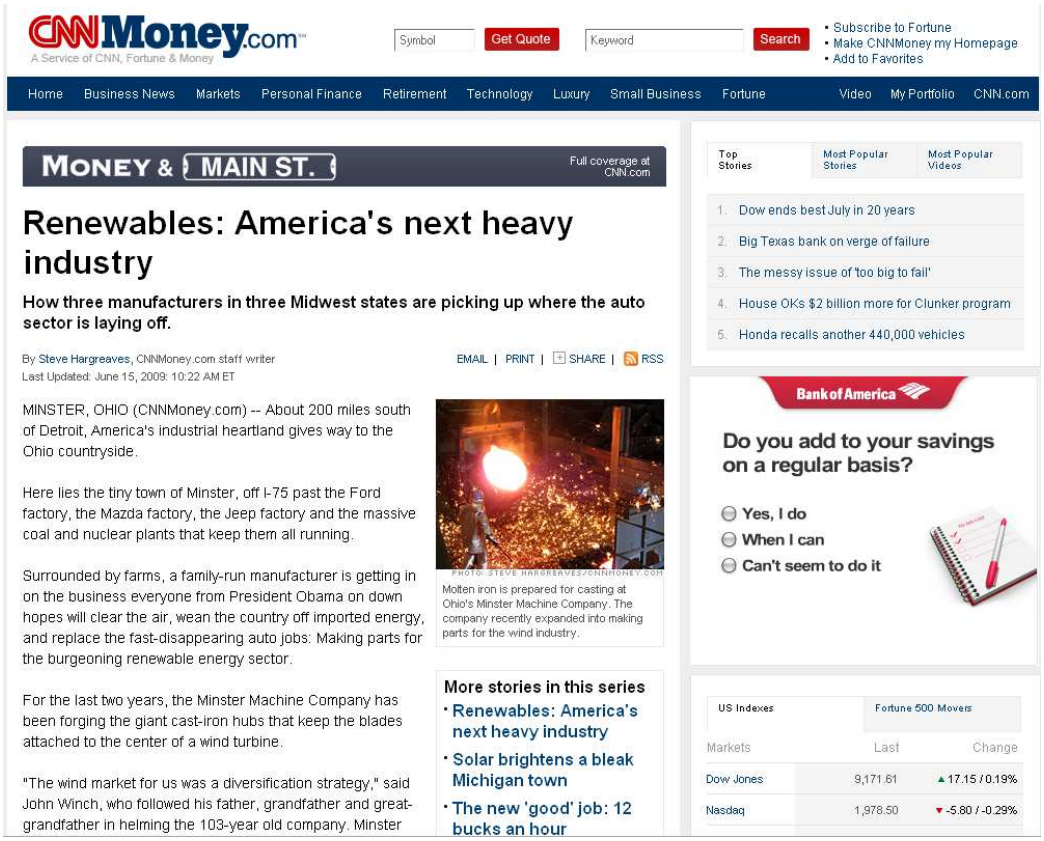

Figure 1.1: Typical example of commercial web page

1.3 Thesis Objective and Methodology

The objective of this thesis is:

Develop a web content extraction method that, given an arbitrary HTML document, should be able to extract the main content and discard all the noisy content.

In order to accomplish our objective, we conduct a study of existing approaches to web content extraction. The problem of web content extraction has been investigated by researchers and many kinds of content extraction methods have been proposed.

In general, there are two issues which can be observed. The first issue is the source base of the content extraction. Typically, a content extraction method uses either the Document Object Model (DOM)[1] representation or the plain HTML source code. The second issue is related to the general approach to do content extraction. Gottron [5] divides the content extraction approach in two categories namely *single document extraction* and *multiple document extraction*.

Since our operational settings require that our content extraction module runs during data acquisition, we need a relatively lightweight and fast method to do extraction. In general, single document extraction is relatively faster rather than multiple document extraction because it only considers the document at hand during extraction without looking into other documents from the same host

Most of the existing methods in single document extraction operate based on certain heuristic in order to perform content extraction. For instance, by examining certain features such as the number of hyperlinks, the text density, the ratio between HTML tag to text etc.

For DOM-based methods, usually the existing methods perform *web page segmentation* prior to content extraction.

Definition 1.1. *Web page segmentation is a task which breaks down the structure of a web page into smaller segments of certain granularity.*

The web page segmentation process is needed because there are dozens of DOM nodes in a single web document and we need to focus on DOM nodes in certain granularity.

[1]The Document Object Model is a cross-platform and language independent convention for representing and interacting with objects in HTML,XML, and XHTML documents. [9]

In this thesis, we use the DOM tree representation as the base since we can obtain many kinds of features by accessing the DOM nodes. Also, by using DOM nodes, we still have the information of the document structure which can be helpful for some reasons e.g. to detect certain pattern structure in the document, to traverse to the other part of the document etc.

As the main approach to perform the content extraction, we applied a *hybrid* approach consist of *machine learning* and *heuristic methods*. The reason to incorporate machine learning is that the existing heuristic-only methods usually use few features and it is quite difficult to determine the threshold value for the feature parameters. With the machine learning approach we can use many kinds of features and the learning algorithm may learn the parameter automatically. Moreover, by using this approach we may apply different learning results for different types of websites. In addition, we also developed other heuristic methods for certain type of website which improve the content extraction result.

Regarding the machine learning part, we used it to perform two tasks namely the *segment classification* and the *content classification*.

Definition 1.2. *Classification is the task of learning a target function that maps each attribute set to one of the predefined class labels [17]. In our case, the class labels are main content, noisy content, good segment, and bad segment. The attribute set consist of DOM properties and the features that we can derive from DOM properties.*

The *segment classification* returns the DOM nodes which are classified as *good segments*. The *content classification* takes the result of the segment classification to further classify the DOM nodes as *main content* or *noisy content*. We named this two classification processes as *two-phase classifier*. Another configuration for the

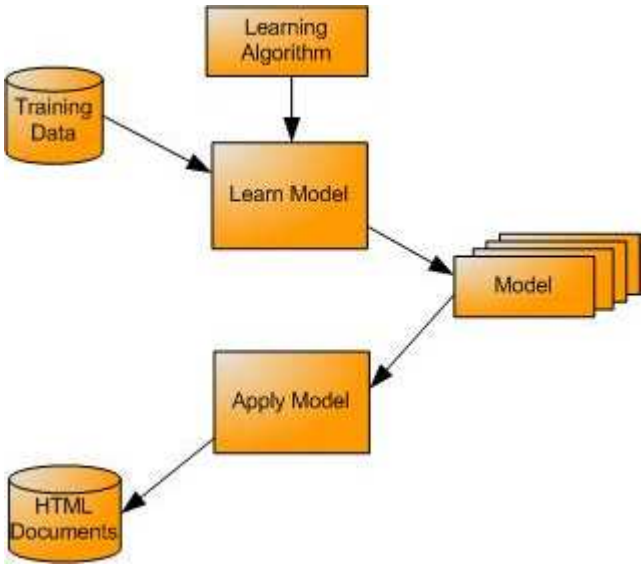

Figure 1.2: High level overview of the classification task

classification that we use is by performing classification only once, we named this as *single-phase classifier*. For the single-phase classification we use two class labels namely main content and noisy content.

In our context, the classification is aimed to make predictive modeling which means the classifier model is used to determine the class label of unseen instances. The example representation of input and output of this task is shown in Table 1.1.

Table 1.1: Example input and output of the classification task

DOM Node	Features	Segment Classification	Content Classification
D_1	$\{f_1, f_2, ..., f_N\}$	good segment	main content
D_2	$\{f_1, f_2, ..., f_N\}$	good segment	noisy content
D_3	$\{f_1, f_2, ..., f_N\}$	bad segment	main content
D_4	$\{f_1, f_2, ..., f_N\}$	bad segment	noisy content

Figure 1.2 illustrated the high level overview of the classification task. The *training data set* consists of labeled training instances from the crawled web pages. For the

learning algorithms we conducted experiments with several algorithms to observe which classifier yields the best performance. Next, the best classifier model can be applied to the unseen HTML documents to perform classification.

In addition to classification tasks, we added some heuristic approaches as a complement. The reason is that only applying classification and extract the text nodes is not sufficient. Most of the time, inside a DOM node there are many embedded noisy contents such as embedded advertisement, user comments, related links etc. By utilizing the heuristic approaches, we expect to obtain more accurate content extraction as we perform further filtering to the DOM nodes.

1.4 Result

The result of this thesis is our research work on web content extraction. Specifically we developed a web content extraction method which consist of machine learning and heuristic approach. For the machine learning part,we discovered features that are important to identify the main content area in a web page, the generalization capability of certain classifier model, and the comparison of the *single-phase* and *two-phase classifier*.

As for the heuristic approach, we developed several methods as a complement for the machine learning part. We developed three heuristic approaches namely *Largest Block String (LBS), String Length Smoothing (SLS)*, and *Table Pattern (TP)*. These heuristics are intended to perform further content extraction after the machine learning process.

We compared our approach to the existing heuristic approaches and according to our experiments, the combination of machine learning and our heuristic approach

gives encouraging result in terms of precision and recall. Regarding the implementation, we integrated our method in a *web content extraction module* of an Information Retrieval System from Teezir B.V.

1.5 Thesis Structure

The structure of this thesis is organized as follows, Chapter 2 provides some background about Teezir B.V, Teezir IR framework, and how our module fit in the framework. Chapter 3 describes the related works on single document extraction. Chapter 4 explains about our approach to the web content extraction problem. Chapter 5 describes the evaluation method to measure the performance. Chapter 6 presents the experiments and evaluation of our approach. Finally, in Chapter 7 we draw the conclusions from our work and present possible improvements for future work.

Chapter 2

Background

2.1 Teezir B.V.

Teezir is a young and innovative technology company that develops comprehensive search solutions. The solution is built up in building blocks related to information retrieval and search technology. These building blocks can be applied and customized in order to create the best business solution for their clients.

Apart from Teezir's fully customized solution, they currently have four main solutions which drive their technology development. These solutions are expert finding, vacancy finding, opinion mining, and focused content. The solution quadrant is depicted in Figure 2.1. The explanations of each of the solutions are as follows:

Expert Finding Solution. This solution allows finding the right person which matches certain skills in an organization or in the market. The document search technology incorporates context and content of the information to enable people find relevant documents in a fast way. This solution based on the document that people write in order to determine the expertise.

Vacancy Finding Solution. This solution gathers data such as CV, resume, and vacancies from numerous job sites. In order to find the best candidate for a vacancy and vice versa, it performs matching between full-text vacancies and candidate profiles.

Opinion Mining Solution. This solution enables to gain insight about the effectiveness of product launches and marketing campaign. The solution measures the perception and opinions of people by crawling and analyzing relevant sources on the web. This allows monitoring the sentiment and polarity over time and also comparing it to the client's competitor. The unique characteristic of Teezir's solution to its competitor is that it can receive the query from the user directly and then perform sentiment analysis. While the competitors usually requires the list of the possible query in advance before they provide the sentiment analysis.

Focused Content Solution. Focused content provides a single point interface for accessing information around a specific topic, domain or audience. Content can be gathered from internal or external resources such as the websites, weblogs, databases and can vary from text format to image/video format.

Figure 2.1: Teezir main solutions

2.2 Teezir Framework

Teezir's search platform provides the functionality for the overall process of disclosing data, namely content gathering, analyzing document, index building, and searching of information. The overall of Teezir's technology building block is shown in Figure 2.2.

As our objective is to develop a content extraction module which operates during data acquisition stage, we mainly focus on content gathering part of Teezir's technology. One of the stages in content gathering is *crawling*. Crawling is an automated process of collecting data typically in hyperlinked documents. A crawler systematically follows the hyperlinks between documents and stores the local copies in the database. The behavior of this crawler is defined in a configuration file which enables to set up the parameters that we need such as the depth, maximum pages to fetch,

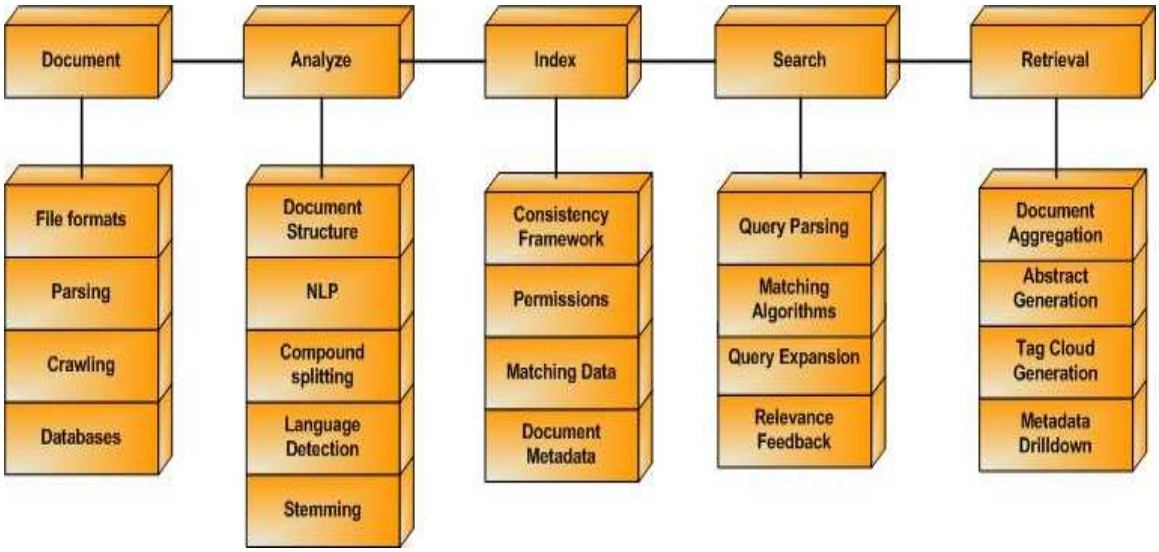

Figure 2.2: Teezir Framework

stop condition, time to re-crawl etc. In the process, a crawler needs to deal with heterogenic nature of content type. Teezir's crawler is able to handle most of HTML and Javascript that most websites have.

Currently, Teezir uses a sophisticated crawler technology that incorporates intelligence inside it. This kinds of intelligence will allow the crawler automatically follow the relevant links based on classification. For instance, given a web page, the crawler can detect the certain type of link such as paging link or job vacancy posting. By having this kind of knowledge, the behavior of the crawler will be more focused and the gathered data will only be the relevant data according to the client's needs.

2.3 Contribution to Teezir Framework

The contribution of our thesis work to Teezir Framework are the *web content extraction module* and the *training module*. The responsibility of the web content extraction module is to perform the *segment classification*, *content classification* and *heuristic approaches*. Meanwhile, the training module allows the user to generate training data set which can be used for machine learning experiments. We used the existing Teezir Framework to implement our modules. As illustrated in Figure 2.3, we used the existing web crawler and HTML parser from Teezir, and as an input for our content extraction module is the DOM tree of the HTML document. The final output of the content extraction module is the main content text string from a HTML document.

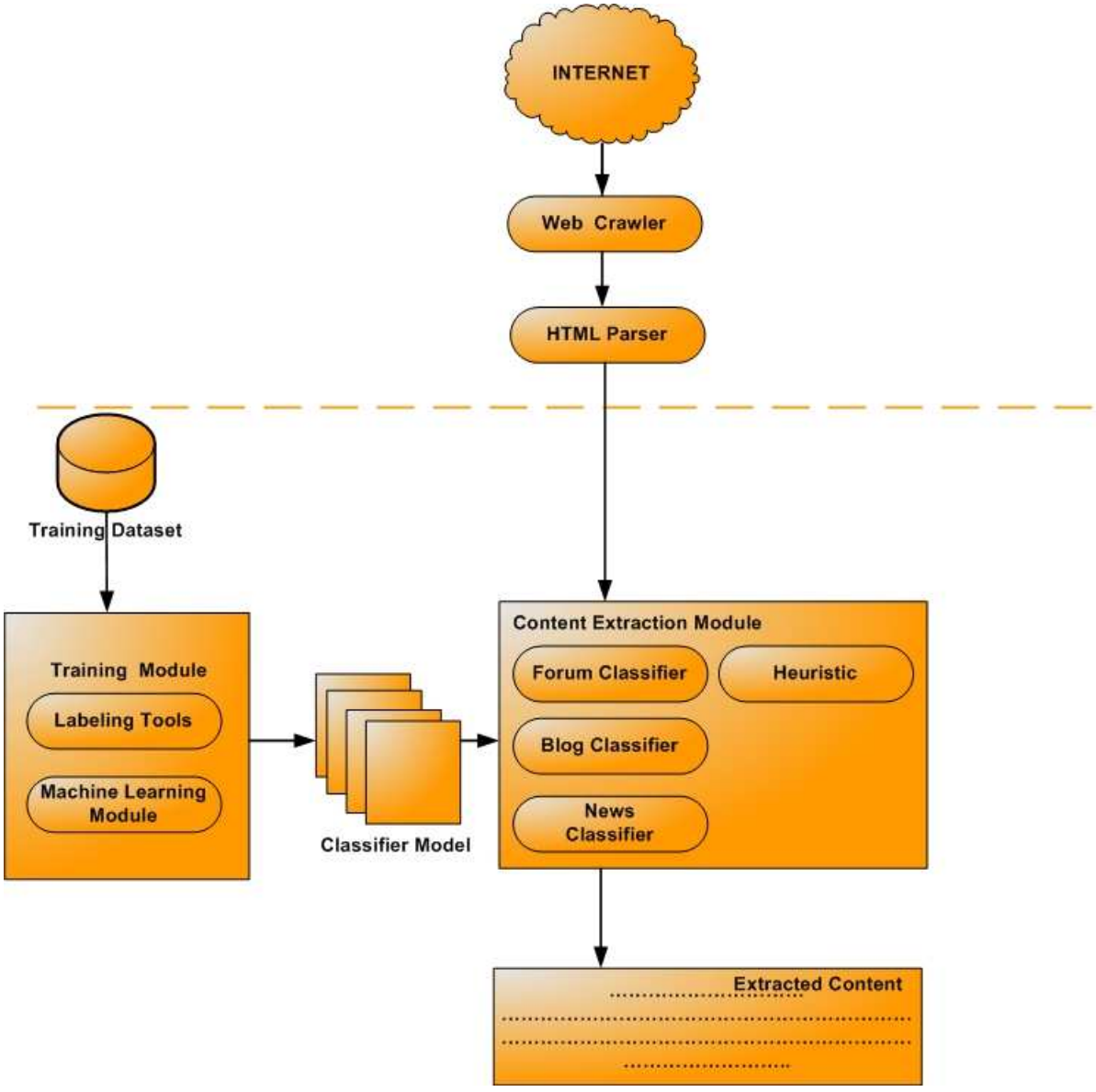

Figure 2.3: Contribution to Teezir Framework

Chapter 3

Related Work

According to Gottron [5], based on the number of pages needed to perform web content extraction there are roughly two different kinds of approaches namely *single document extraction* and *multiple document extraction*. The single document extraction uses heuristic approaches operating on single documents while multiple document extraction applies analysis of the template structures in document collections.

As described in Section 1.3, we focus on *single document extraction*. In addition to the heuristic approaches, the single document extraction can be also categorized based on source basis of the content extraction namely *DOM-based* and *non-DOM based*. For DOM based methods, they utilize the DOM tree which represents the HTML document structure and navigate through this tree to perform content extraction. For non-DOM based methods, they only use plain HTML source code to do the content extraction.

In this chapter we outline several approaches of single document extraction algorithms including both DOM-based and non-DOM-based ones.

3.1 Crunch

Crunch framework is a content extraction program developed by Gupta et al [20]. Instead of using raw HTML text, it uses the DOM tree representation of a web document. Receiving input of a HTML page, Crunch will parse the HTML string, construct the DOM, traverse the nodes recursively and filter out the non-informative content behind. Each of the filters can be turned on or off and customized to certain degree.

Algorithm 3.1 illustrates the main filters of Crunch. Generally there are two sets of filter in Crunch. The first set simply ignores specific HTML tag such as styles, links, and images. The second set consists of the advertisement remover, the link list remover, the empty table remover, and the removed link retainer. The advertisement remover maintains a list of advertisement server addresses to detect whether a DOM node contains advertisement elements. The link list remover employs a filtering technique by calculating the ratio between the number of links and non-linked words. This ratio will be compared to certain threshold which can be customized. The empty table remover simply removes table element which doesn't have substantial information inside it by looking at the number of characters. The removed link retainer adds link information back at the end of the document to keep the page browseable. After performing a series of filtering pipeline the final output of Crunch can be customized as a transformed HTML document or plain text format.

After developing the first version of Crunch, the framework was refined and some improvements were added [21] [22]. One of the advantage of Crunch is the genre detection. It uses a classifiers based on the distribution of keywords to classify a web site to one of the pre-defined categories before doing the classification such as

news websites, portal websites, shopping websites etc. According to Gupta et al [22], they intended to incorporate machine learning techniques for content extraction in addition to their web genre detection. However, until now there is no sign that they will release the new version. An disadvantage of Crunch is that it does not allow re-training process as the heuristic rules and threshold values are fixed.

Algorithm 3.1 Crunch Filtering algorithm

```
Input: D: DOM node
  nodeType ⇐ D.getNodeType()
  parent ⇐ D.getParentNode()
  if nodeType = ELEMENT_NODE then
    nodeName ⇐ D.getNodeName()
    / * First Filter * /
    if nodeName = DIV and settings.ignoreDivStyles = true then
      removeAttribute(D, "style")
    end if
    / * Second Filter * /
    if isAdLink(D) and settings.ignoreAds = true then
      parent.removeChild(D)
    else if nodeName = TD and settings.ignoreLinkCells = true then
      linkTextRatio ⇐ getLinkTextRatio(D)
      if linkTextRatio > threshold then
        parent.removeChild(D)
      end if
    else if isTextLink(D) = true and settings.ignoreTextLink = true then
      parent.removeChild(D)
      if settings.addLinksToBottom = true then
        addLinks(D)
      end if
    end if
  end if
```

3.2 Feature Extractor (FE)

The Feature Extractor (FE) algorithm by Debnath et al [18] is a content extraction algorithm which based on DOM block structures. The algorithm segments a web document into blocks and selects certain blocks to be extracted. A block here corresponds to the DOM subtree nodes. The algorithm will start working from the root node and recursively splitting the document into blocks. They defined a set of HTML tags which denotes a block namely `table`, `tr`, `hr`, and `ul`. FE uses the feature such as the presence of nested blocks, texts, images, applets, or contained JavaScript code.

FE will extract the blocks which is dominant in certain features. In the context of content extraction, for example, we can set the feature we need it's the text properties. As a result the blocks that will be extracted will be those which is rich with text. The complete algorithm of FE is shown in Algorithm 3.2.

The K-FeatureExtractor, the variant of FE, works a little bit different, instead of simply choosing one single winning block the blocks which are able to pass the first iteration are clustered using a k-means clustering algorithm. Afterwards, the cluster with the highest probability for the desired feature is chosen as the winner. FE allows many kinds of features to be incorporated in the content extraction process however we have to select the features manually and define the proper threshold values.

Algorithm 3.2 Feature Extractor algorithm

Input: D: DOM node (root node), T: set of block defining HTML elements
Output: B: Set of blocks

```
B ⇐ D
for all t ∈ T do
  for all b ∈ B do
    if b hasChildNode(t) then
      B^N ⇐ getBlocks(b, t)
      B ⇐ (B − b) ∪ B^N
    end if
  end for
end for
return B

function getBlocks(b, t);
B ⇐ ∅
C ⇐ descendants(b)
for all m ∈ C do
  if elementType(m) = t then
    B ⇐ B ∪ {m}
  end if
end for
```

3.3 Link Quota Filter (LQF)

LQF [21] is a simple heuristic method to remove link lists and navigational elements. There are many variations of LQF implementations, however generally it measures the ratio between hyperlinked contents to the non-hyperlinked contents in a DOM node. If the ratio is greater than certain threshold, the DOM node should be removed. LQF may work effectively to remove additional contents which consist of high frequency of hyperlinked text but most likely will fail to deal with additional contents such as headers, footers.

Algorithm 3.3 LQF algorithm

Input: D: Document, t: threshold
Output: D': modified documents without link list blocks

```
N ⇐ descendants(D);
for all n ∈ N do
  if isBlockNode(n) then
    if LinkQuota(n) > t then
      C ⇐ descendants(n);
      for all m ∈ C do
        if ¬isBlockNode(m) then
          if isTextNode(m) then
            deleteNode(m);
          end if
        end if
      end for
    end if
  end if
end for
return D
```

3.4 Largest Pagelet (LP)

LP [5] is an adaptation of page partitioning algorithm from Bar-Yossef and Rajagopalan [24]. As illustrated in Algorithm 3.4 the partitioning algorithm works based on the DOM structure of a web document and breaks down the DOM structure until a stop condition is met. The DOM node which satisfies the stop condition is called a *pagelet*. The stop condition of the algorithm is the *number of links* inside a pagelet.

In order to extract the main content, the LP algorithm only considers the pagelet which has the largest visible text portion. Any invisible text inside script and style are skipped. One of the important points in this algorithm is the value chosen for parameter k, the threshold of number of hyperlinks. Bar-Yossef and Rajagopalan used $k = 3$ but did not explain in detail how they arrived at this value. If the value k is

Algorithm 3.4 LP Page Partitioning algorithm

Input: D: HTML Document, k: threshold of number of hyperlinks per pagelet
Output: P:

```
q.append(rootNode(D));
P ⇐ ∅;
while ¬isEmpty(q) do
  n ⇐ q.pop();
  C ⇐ descendants(n);
  if ∃c ∈ C : countLinks(c) ≥ k then
    for all c ∈ C do
      q.append(c);
    end for
  else
    P ⇐ P ∪ {n}
  end if
end while
return P
```

chosen properly for a particular document, most likely LP will give good performance in terms of identifying which DOM node contains the main content. However, LP may suffer from embedded noisy contents such as user comments which usually have low frequency of hyperlinks. In addition to that, as LP only returns the content from one pagelet that contains the largest visible text, it is difficult for LP to extract the main content that is scattered in multiple pagelets.

3.5 Document Slope Curve (DSC)

DSC [2] is another content extraction method which only uses the HTML source code to extract main contents. DSC tokenizes the HTML source code and counts the cumulative tags. After that it will analyze the cumulative tag distribution to identify the main content in a web document. Suppose we have an arbitrary HTML

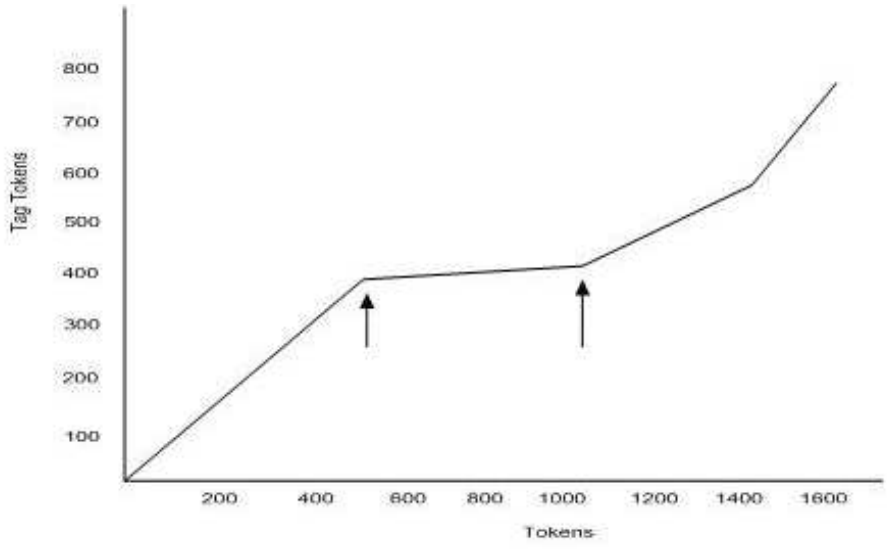

Figure 3.1: Cumulative tag distribution

document, DSC will tokenize the document and saves the frequency of tag token in every token position in the document. The result of this process can be seen as the graph of tag tokens vs token as illustrated in Figure 3.1.

In this example, the area which pointed by the arrows are the main content of the document. They assume that the long regions with low slope (contain less HTML tags) are the main contents. In order to check the average slope, they applied windowing technique to define the document section. If the average slope of the document section within the current window is less than 50 % of the average slope of the entire document, the document section is marked as low slope section.

One of the advantage of DSC is, it can retrieve more than one main content in a web page as low slope section may occur in a scattered manner in the document. However, there is a possibility for DSC to fail when deal with noisy contents such as user comments as most likely user comments will have low slope value.

3.6 Content Code Blurring (CCB)

CCB [5] is another web content extraction algorithm which borrows the idea of image blurring. The idea and aim of content code blurring is to locate those regions in a document which contain "a lot of content and and little code". All the tags in the source code corresponds to *code* while everything else corresponds to *content*. CCB tokenizes the HTML source code and store every character in a one dimensional vector called *content code vector*. Each entry in the vector is initialized with a value of 1 if the element represents type content and with a value of 0 for code.

To obtain the actual content code ratio which incorporates the surrounding element values, it calculates the weighted-average of the values in the neighbourhood of each entry. Gottron [5] defines the parameter radius r to control how far the algorithm should go to the sourrounding elements. If all the elements in a neighbourhood have a content code ratio of 1, the average neighbourhood will be 1. The same is valid for neighbourhood with the initial value of 0. If the element values in the neighbourhood are mixed then the average value will be between 0 and 1. For calculating the weighted-average values, CCB uses the Gaussian distribution function. The value of each element will be updated with the new value that obtained from this function (*blurring*). The blurring iteration is stopped as soon as the values stabilize.

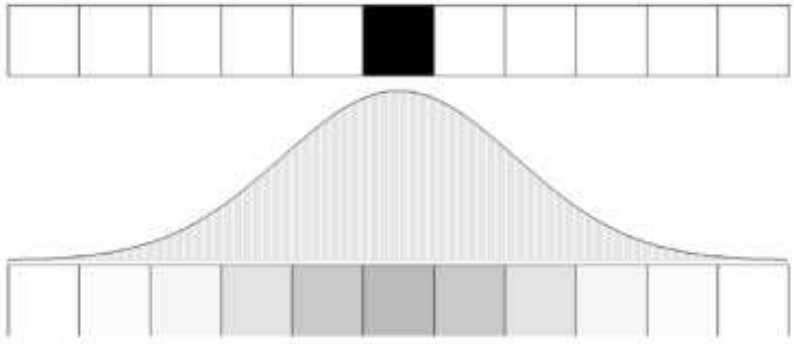

Figure 3.2: Content code blurring

Figure 3.2 illustrate the effect of blurring the content code vector. In the upper image, we can see the initial condition of the content code vector. After the blurring process, we can see in the below image that the value of a character will spread to its neighbours. Finally, character elements which have value above certain threshold will be extracted as content. In general, according to Gottron [5] we need to tune the parameter r and observe the extraction result in order to obtain good result from CCB. Essentially, the larger r we choose, the recall will be higher but lower the precision while smaller r will yield higher precision but lower recall.

3.7 Summary

In this chapter we have presented several related works to web content extraction, in particular for single document extraction. Generally, single document extraction utilizes heuristics and operates either on DOM tree representation or plain source code of HTML documents.

Typically, for heuristic rules they introduced one main feature to distinguish main content from noisy information. In addition to the heuristic rules, the existing approaches, except Crunch, basically treat all the web pages the same way. For example, there will not be any difference between extracting the main content from news web pages and blog web pages. Table 3.1 summarizes the related works that we presented in this chapter.

Table 3.1: Single document extraction summary

Method	Source of Extraction	Main Heuristics
Crunch	DOM tree	link to text ratio, static list of advertisement host server, presence of empty cell, removal of HTML style
Feature Extractor (FE)	DOM tree	Extract DOM nodes which are rich in certain feature
Link Quota Filter (LQF)	DOM tree	Remove DOM nodes which has high link to text ratio
Largest Pagelet (LP)	DOM tree	Performs page partitioning and extract DOM nodes which have the largest visible text
Document Slope Curve (DSC)	plain HTML	Extract characters from regions which have low frequency of HTML tag
Content Code Blurring (CCB)	plain HTML	Performs blurring and extract characters which have certain value of *content to code ratio*

Chapter 4

Our Approach

In this chapter we present our approach in web content extraction. Our approach is different in the following ways. First, it uses learning process to discover the main content area in the web page. By using this learning process, we can combine many kinds of different features and the learning algorithm may deduce the parameters for the features automatically. In this way, at least it will reduce the amount of work to specify appropriate feature parameter values. Second, by using this approach we may train the algorithm to extract content from different types of web pages.

As an overview, our approach can be seen as a one pipeline process. As shown in Figure 4.1, the input is the HTML document and the output are the text strings extracted from the HTML document. In between, there are two processes namely

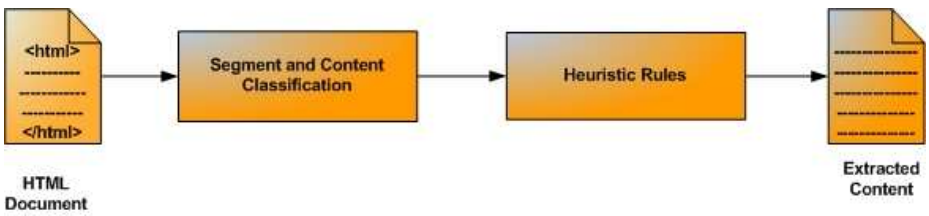

Figure 4.1: Overall approach of the content extraction

classification and heuristic rules. The classification process is used to identify the segments which contain the main content. The output of the classification task is one or more segments (DOM nodes) which classified as main content. These segments will be the input of the heuristic rules which performs further filtering to extract the text string of the main content. The following sections will explain about the classification and heuristic processes.

4.1 Classification

4.1.1 Segment Classification

The goal of segment classification is to break down the structure of a web document into smaller segments with certain granularity. In order to do this we based our segmentation process from the Document Object Model (DOM) of the web document. In this section we start by describing about the DOM structure and next we show how we use the DOM structure to perform segmentation of a web document.

According to W3C specification [9], DOM is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way how a document is accessed and manipulated. By utilizing DOM, one can construct document, navigate through the structures of the document, and perform operations such as add, update, delete the properties of the elements. As a standard programming interface, DOM is designed to be programming language independent. There are numerous languages binding for DOM such as Java and ECMA Script (an industry-standard scripting language based on JavaScript and JScript).

In the DOM, the logical structure of a document is represented as a tree structure. An example of a DOM tree representation is shown in Figure 4.2. From the illustration we can see the DOM tree of the respective web document rendered on the web browser. Essentially, every HTML elements rendered in the browser correspond with one node in the DOM tree. For example, the DOM node which is indicated by the arrow sign in Figure 4.2 corresponds to the side menu in the rendered web document.

The DOM based segmentation process starts from the body of the document and skips the HTML version information and header section since those two parts most likely will only contain scripting and style sheet declaration and they are not visible elements.

In the body part of the document, there can be dozens of HTML elements inside. In order to focus our segmentation process we only interested to HTML elements which define structural style in the document. These elements basically are used to define sections in the document. According to the HTML 4.01 specification [10], there are several HTML elements which define structural element namely block level elements (`div`, `span` tags), list elements (`ul`, `ol`, `li` tags), table elements (`table`, `tr`, `td` tags).

We traverse the nodes in the DOM tree in a depth first manner and for each node that we encounter; we check whether a node is a structural elements. If it is a structural element we need to check whether it is a *good* segment. The meaning of *good* in this context, is related to the content uniformity of the segments. For instance, a segment that contains only advertisements, only main content etc. In addition to that, a segment could be very coarse or fine-grained. In general, we do not prefer segments which are extremely coarse or fine grained.

Figure 4.2: DOM tree structure of a web page

To illustrate about this granularity issue, consider Figure 4.3. From the figure we can see that there are many structural elements with different granularities. For

Figure 4.3: Segment granularity in a web page

example, there is a segment which covers all the headers, content, title, and right menu bar. But inside this segment there are more segments which less coarse such as segments which only contain a right menu bar, a content, or a header. Even, inside the right menu bar, the segment can be broken down further into finer grained segment.

For example, in Figure 4.3, a segment "Best Gelezen" in the right side, can be broken down into five more segments. Generally, from this observation, what we

prefer as a segment with good granularity is a segment which represents uniform semantic unit. For human, it is very easy for them to distinguish the semantic of segments in a web document. In the other way around, it is difficult for machine to do this.

In order to select the granularity we will use supervised learning to train the algorithm to recognize a segment with a good granularity. We view this problem as a classification task whether a DOM node is a good segment or not.

4.1.2 Content Classification

After the segment classification takes place, the output which are DOM nodes classified as good segment, are used as inputs for content classification. The content classification basically takes the feature vector of a segment and then classifies it whether it is a main content or noisy content.

4.1.3 Classification Representation

As we want to apply machine learning to do the segment classification and the content classification process, we need to transform the representation of the DOM node segments into feature vectors for classification tasks. The representation of our training data is represented as a feature vector:

$$< X_1, X_2, X_3, ..., X_n, Y >$$

X_i The feature of the DOM node at index i in the vector.

Y The class label of the feature vector.

With this feature vector representation, the classification task maps each feature vector X into its class label Y. In our case, the classification model is used as a predictive modeling meaning that it is used to predict the class label of unknown instances.

4.1.4 Classification Configurations

Two Phase Classifier

In this configuration, we separate the segment classification and content classification tasks. First, given the DOM tree of an HTML document, the segment classification will return the DOM nodes which classified as good segments. After that, from these good segments, content classification is performed. In the end, DOM nodes which classified as good segments and main content are returned. We performed feature selection for segment classification and content classification so that there are differences in terms of features which are used during classification process.

Single Phase Classifier

Instead of separating the classification into two phase classification, it is more natural if we merge both classification tasks. This means that the segment classification and content classification is performed at the same time. In the single phase classification we defined the class labels as *main content* and *noisy content*.

4.1.5 Features

In order to do the classification tasks, we need to collect all necessary features from DOM nodes or other features than can be constructed from the existing features. Most of the features that we used are from DOM properties and similar to Song et al [19], they used the features to classify segment importance into three levels of importance. In addition to that, we added some new features other than DOM properties namely *stop word ratio*, *header around*, and *DOM height*. We also incorporate link to text ratio [22] to the features. The features that we present here are used for both single phase classifier and two-phase classifier.

As for values of the features, we *normalized* the values e.g. the *innerTxtLength* of certain DOM node is the length of the inner text in the DOM node divided by the total inner text length in the document.

innerTxtLength

The innerTxtLength is the length of all of the text string in all of the text nodes in a DOM node. The length of the inner text in a DOM node can be an indication that it contains main content.

innerHtmlLength

The innerHtmlLength is the length of all HTML string between start tag and end tag of an object. The length of the inner HTML in a certain DOM node can be used as heuristic to determine whether a DOM node is a good segment in segment classification or main content in content classification.

imgNum

The imgNum is the number of image elements inside a DOM node. Typically, headers and footers of web pages have images without text elements on it. Meanwhile, the main content sometimes has image in it as an illustration. Therefore, number of images can be used as a feature to determine whether a DOM node is a main content.

interactionNum

The interactionNum is the number of `input`, `select` elements inside a DOM node. HTML elements such as `input`, `select` usually used to provide interaction between the user and the application. These kinds of interaction components are rarely part of the main content. Therefore, it could be useful as a feature to distinguish whether an element is a main content or noisy content.

formNum

The formNum is the number of `form` element inside a DOM node.

optionNum

The optionNum is the number of `option` element inside a DOM node. The motivation to choose optionNum as a feature also similar to interaction components and forms since `options` only provide interaction between the user and the application which usually is not the part of the main content.

tableNum

The tableNum is the number of `table`, `tr`, `td` elements inside a DOM node. Before the extensive use of `div` tag and Cascading Stylesheet (CSS) in the modern web documents, tables are often used to define the layout structure of the document and as a placeholder for main contents. However, sometimes we still can find ad hoc web pages that still use tables for layout purposes. Therefore we use this as a feature to determine whether a DOM node is a good segment and whether it contains main contents.

paraNum

The paraNum is the number of `p` element inside a DOM node. In web documents such as blogs, news websites, the main content usually is placed inside the paragraph element.

linkNum

The linkNum is number of anchor element (hyperlink) inside a DOM node. If a DOM node contains quite a lot of links most likely it could be a navigational elements and not a main content.

divNum

The divNum is the number of `div` elements inside a DOM node.In the modern web documents, `div` tags are frequently used to define the layout structure of the document. The number of `div` tags inside a DOM node can be used as a hint whether it is a good segment and main content.

centerX, centerY

Suppose that we have a center point (X,Y) of a DOM node, centerX is the X component of the center point and center Y is the Y component of the DOM node. The visual position of the DOM node can be a good hint to determine whether a DOM node is main content or not. For example, headers and footers usually located on the top or bottom of the web page. Main content usually located in the center of the page. In the actual implementation of the web content extraction module, we did not use these feature as it is not trivial to make an HTML renderer to get the pixel coordinates of the HTML elements.

width, height

Assuming that a DOM node segment is a rectangle shaped, the width and the height denotes the width and the height of the rectangle that surrounds the DOM node. Combined with CenterX and CenterY, the width and the height properties may assist determining a good segment and main content. For example, if a DOM rectangle has large width and height most likely it's not a good segment and still contains many noisy contents. In the actual implementation of the web content extraction module, we did not use these feature as it is not trivial to make an HTML renderer to get the pixel coordinates of the HTML elements.

linkToTextRatio

The linkToTextRatio in a DOM node is the ratio between the length of the anchored text and the total length of the text in the DOM node. A DOM node which has high value of link to text ratio most likely would be navigational elements and not main contents.

stopWordRatio

The stopWordRatio is the number of stop words that contained in all of the text nodes of a particular DOM node. Based on our observation, typically, there are many stop words occur in the main content. However, there is a possibility that the stop words occur in the noisy contents such as recent comments section of blogs which rich of anchored text. Therefore, we introduce a penalty for stop words. Given the inner text length ($inTxtLen$), anchored text lengh ($ancTxtLen$), and number of stop words ($nbSw$), the stopWordRatio (swr) is calculated using the formula in Equation (4.1).

$$swr = nbSw \times \frac{inTxtLen - ancTxtLen}{inTxtLen} \tag{4.1}$$

tagName

The tagName is the name of the HTML tag of the DOM node. Different HTML tags usually have different purposes for document presentation. For example, `li` and `ul` tag usually used to make a linked list of hyperlinks or navigational elements while `div` and `table` used to define the document layout structure and content. Therefore, HTML tag can be used as a feature to identify whether a DOM node is a good segment and whether it contains the main content.

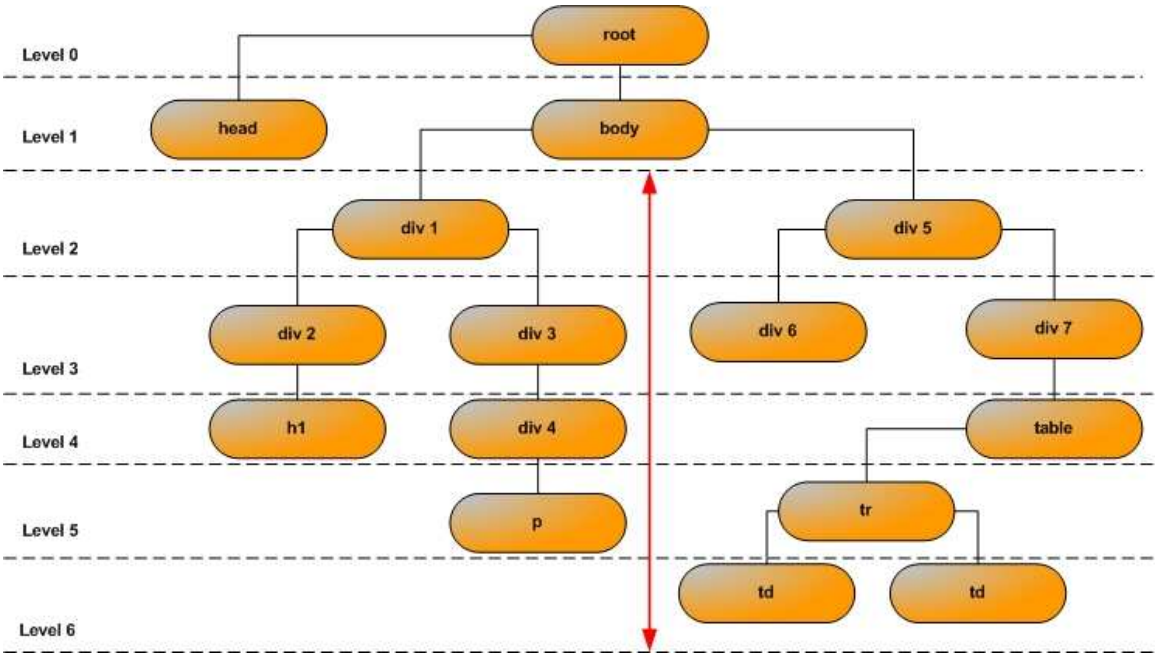

Figure 4.4: Height of DOM elements

domHeight

The domHeight is the maximum depth that can be reached a particular DOM node to a certain leaf node. The example of DOM tree with its associated depth level is shown in Figure 4.4, in the figure, the DOM height of element `body` is five. The explanation as follows, from `body` element, we can reach four leaves, namely `h1` (at Level 4), `p` (Level 5), `div 6` (Level 3), and two `td` elements (Level 6). Thus, the maximum depth is the path from `body` to `td` elements which give five units height.

In order to do segment classification, we should select a certain DOM node which has good granularity level. The DOM height could be used as a feature to determine whether a DOM node is a good segment or not.

headerAround

This feature indicates whether there are any header elements near a particular DOM node. In order to do this, we check the parent node, sibling nodes, and children nodes of the DOM node whether any header element exist. In this case, the header element is denoted by `h1`, `h2`, `h3`, `h4`, and `h5` tags. Based on our observation, most of the main contents such as news articles, blog posts have title around them. The title usually represented by header tags.

4.2 Additional Heuristic Approach

After the classification process is conducted, the returned DOM nodes may have problems. The example of these problems e.g. in many cases there are embedded noisy contents such as embedded advertisements, link to related articles, short snippets of the article etc. Comments from the web page visitor also may jeopardize the classification result. Therefore, it is not a wise decision to extract the content by only traversing all the text content in the DOM node and get the text string. In order to alleviate this problem we apply heuristic approaches.

As our problem domains consist of news, blogs and web forums we need to find suitable heuristic characteristics that applicable to those domains. From our observations, news and blogs are similar in the sense that in each web page contains a main content e.g. a blog post, a news article and surrounded by many noisy contents. However, for web forums the situation is different as in a web forum page typically it consist of many postings. Therefore, we will separate the heuristics used for news and blogs with web forums.

Figure 4.5: DOM Structure of a node which classified as main content

4.2.1 Largest Block of String (LBS)

Based on our observation, there is a common pattern of text nodes that contain main content in a news or blog page. The text strings are always placed contiguously in the same depth level of a DOM sub tree. Most likely, the text strings are placed inside a `p` tag but however it also can be placed inside other tags such as `div` or plain `#text` node.

As illustrated in Figure 4.5, the main content in this context is placed in a series of `p` tag. The way we find the largest block of string is by traversing the DOM tree depth first based and then for each level in the sub-tree we find a child node with the longest string. After that we look to the sibling nodes of the child node in order to discover the neighboring text string. We will add the sibling's text string if it contains `#text` node. In order to skip the embedded noisy information such as advertisements

or scripts, we check the content of the sibling's node. If it doesn't contain `#text` string as a child, we will skip it. We repeat this process until all the sub trees are examined. In the end, the output is the largest block of text string found in a sub tree.

In case more than one DOM node is returned from the classification process, we may use this LBS heuristic to filter out noisy DOM nodes by comparing their longest string length.

4.2.2 String Length Smoothing (SLS)

The String Length Smoothing (SLS) is inspired by the approach of Wenninger [23] and Moens [13]. In [23] they used text to HTML code ratio and then perform smoothing function and clustering. The difference with [13] is that they use windowing and cutoff factor relative to maximum string length to decide whether a string should be regarded as content. In this heuristic we tried to experiment with the length of string inside a DOM node which is classified as a main content. SLS work as follows, we traverse into all the text nodes and put the length of the string into a one dimensional array, *stringArray*. Every time we encounter the HTML tags that modify the structure such as `div`, `tr`, `td`, `table`, `ul`, `li`, we put an empty element into the array.

Figure 4.6 shows the length of value of the string inside a DOM node in one of the web page in our data set. It can be seen from the figure that in the middle of the chart, there are several strings with large length. Likewise, there are also small strings which are actually part of the main content. In order to detect these small strings, we perform the value smoothing to spread the value of large strings to the

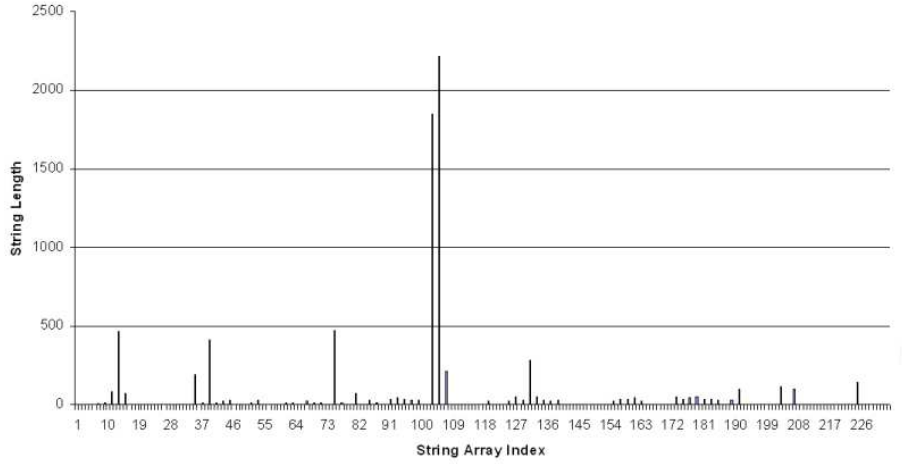

Figure 4.6: String length before smoothing

neighboring strings.

We used the formula in [23] to calculate the smoothing value. The formula is shown in Equation (4.2). We smooth the value in *stringArray* by computing the new value e_k for a given radius r for each element k. The resulting smoothing values are shown in Figure 4.7. It can be seen that there are increases of string length for the small strings which are located near the large strings.

$$e_k = \frac{\sum_{i=k-r}^{k+r} stringArray_i}{2r+1} \tag{4.2}$$

After we perform the smoothing of the string length, we extract the string which has the length larger than the threshold. We followed [23] to use the *standard deviation* of the string length as the threshold value.

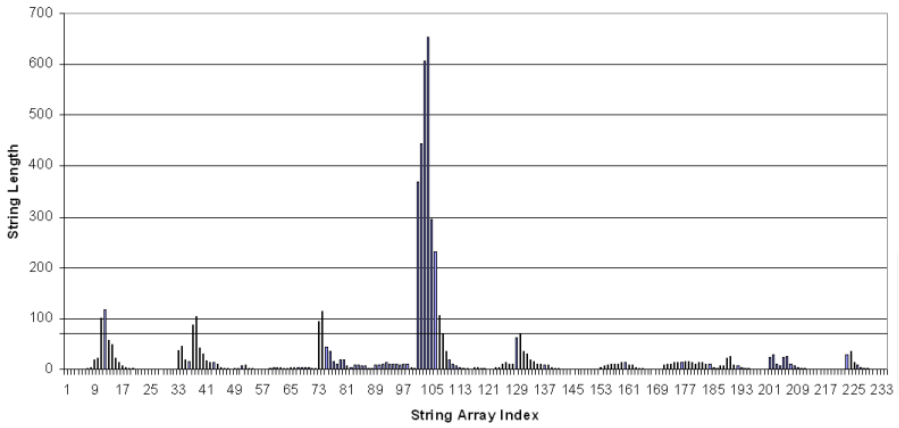

Figure 4.7: String length after smoothing

4.2.3 Table Pattern (TP)

The Table Pattern (TP) is developed mainly to extract the content from web forums. The heuristic comes from the observation that almost all web forums are presented in a table like structure. Among forum posts in the same page, they share the same pattern of the table. As we can see in Figure 4.8, there are two posts and the table structure are similar.

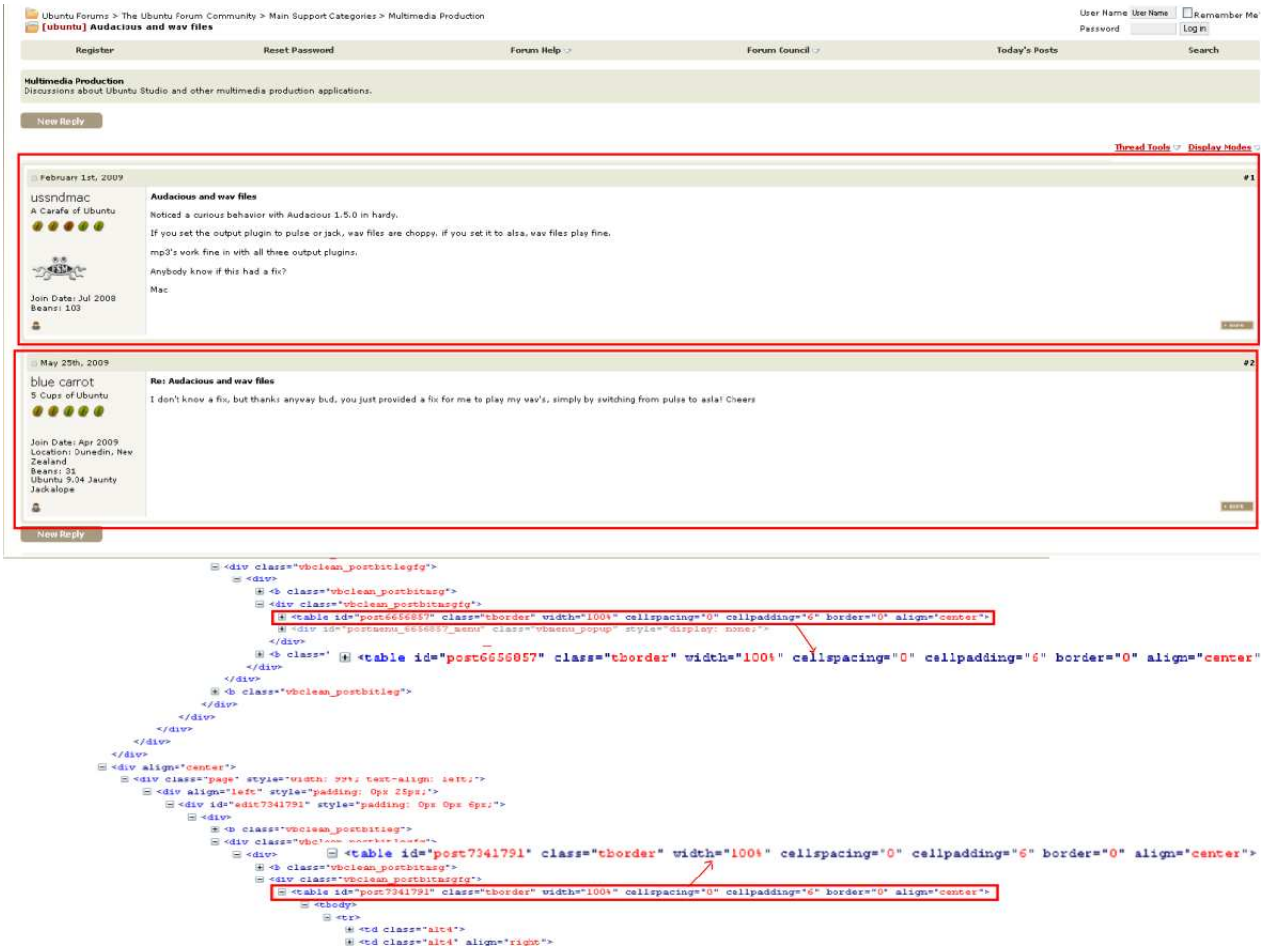

Figure 4.8: Table pattern in forum websites

Moreover, we can observe this similarity from the internal DOM structure. The tables which contain the forum posts share almost the same properties such as CSS Class, border, width, align, cell spacing, cell padding etc.

The way we use this heuristic is as follows, after the segment and content classification processes we will get a number of DOM nodes. For each of the DOM nodes we retrieved the nearest `table` parent. Next we can compare the properties of the tables. For tables which have the similar properties, we perform the text extraction. Based on our observation from the top 50 most visited web forum in [8], the tables that contains forum post indeed share the same properties that we mentioned before e.g. CSS class, width, cellspacing, cellpadding, border, align, and also the "id" follows the similar signature.

4.3 Summary

In this chapter we have described our approach for extracting the main content from web documents. Essentially, our approach consist of two processes namely classification and heuristic approach. The initial input of our approach is the HTML document and the final output is the text string of the main content. For the classification process we have two configurations namely *two-phase classifier* and *single-phase classifier*. The outputs of the classification process are DOM nodes which are classified as main content. However, there is a possibility that there is an error in the classification or embedded noisy contents inside the DOM node. Therefore, we apply heuristic rules to further filter the DOM nodes and obtain the exact text string of the main content.

Chapter 5

Evaluation Method

In order to measure the performance of our content extraction approach, an evaluation method is needed. However, in most publications, there is no common way to evaluate content extraction task. The evaluation method and dataset used between one publication and another are often different. This makes content extraction methods are difficult to compare to each other.

For example, the approach proposed by Lin and Ho [15], they basically segment the web pages into DOM blocks. After that they analyze whether a DOM block is an informative content or noisy content. They based the evaluation on the classification accuracy of the block and computing the standard IR measure such as precision, recall and F1 measures. This approach is useful for evaluating the segment and content classification of the DOM nodes. However, it does not measure the exact accuracy of the content extraction. In order to measure the accuracy of the content extraction, we follow the evaluation approach from Gottron [5] by using Longest Common Subsequence measure. In the following section we will explain the evaluation methods and the metrics used to measure the performance.

5.1 Classification Evaluation

The classification evaluation is conducted to measure the performance of classifier in segment and content classification. This evaluation is used merely to select the best machine learning technique for the classification tasks. The entities that we measure in this evaluation are the labeled DOM node instances.

The metrics that we use for this classification evaluation are precision, recall, and F1 measures. In order to compute those metrics, we need to calculate the number of true positive, false positive, false negative, and true negative as shown in Table 5.1. In our context, the positive class would be main content (for content classification) and good segment (for segment classification), and the negative class would be noisy content and bad segment.

Table 5.1: Classification confusion matrix

Actual Class	Predicted Class	
	Positive	Negative
Positive	tp (true positive)	fn (false negative)
Negative	fp (false positive)	tn (true negative)

The recall (r), precision (p), and F1 measures are defined in Equation (5.1). We also incorporate the F1 measure in the evaluation to observe the balance between precision and recall.

$$r = \frac{tp}{tp + fn},\ p = \frac{tp}{tp + fp},\ F1 = \frac{2 * p * r}{p + r} \tag{5.1}$$

5.2 Content Extraction Evaluation

This evaluation is used as the final performance measurement of the content extraction process. Basically, the content extraction evaluation compares the expected output (gold standard) and the extracted content that our module produces. In order to be able to measure the performance of content extraction on a particular document, we need to define the gold standard. This gold standard will serve as a benchmark of our content extractor.

Once the gold standard has been created the next task is to measure the similarity between gold standard and the extracted content of our content extractor. As a method to evaluate the similarity between two text strings we use the Longest Common Subsequence (LCS) [6] of both strings. LCS is the maximum subsequence of two or more strings. LCS is based on character sequence and evaluates the agreement between the gold standard and extracted content in character level. The similarity between the gold standard and the extracted content is affected by the length of the characters and also the order in which the characters appeared. In addition to that, the punctuation marks and other symbols are also taken into account.

Likewise, as an evaluation metric we use precision, recall, and F1 measure. In order to apply these measures we need to find the overlap between the gold standard and the extracted content. Looking at the LCS as the intersection of the gold standard and the extracted content, the computation of recall (r) and precision (p) are defined in Equation (5.2)

$$r = \frac{|c|}{|g|} \; p \; = \frac{|c|}{|e|} \tag{5.2}$$

$|c|$ The length of the longest common subsequence.

$|g|$ The length of the gold standard string.

$|e|$ The length of the extracted string.

The F1 measure is computed the same way as in Equation (5.1)

5.3 Statistical Significance Test

5.3.1 McNemar's Test

McNemar's approximate statistical significance test [3] is used to compare two classifiers at a particular value of bias. To determine if one classifier (*C1*) significantly outperforms the other (*C2*), the χ^2 statistic is calculated using Equation 5.3:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \tag{5.3}$$

n_{01} Number of cases misclassified by *C1* and classified correctly by *C2*.

n_{10} Number of cases misclassified by *C2* and classified correctly by *C1*.

If the *null* hypothesis, which states that one classifier performs no better than the other, is correct then the probability of χ^2 being greater than $\chi^2_{1,0.95} = \mathbf{3.841459}$ is lower than 5%. If the calculated χ^2 is greater than that, we may reject the null hypothesis, assuming that one classifier performs better than the other.

5.3.2 T-Test

The t-test [4] assesses whether the means (average) of two groups of samples are *significantly* different from each other. The two sample t-test for a difference in mean can be either *unpaired* or *paired*. The unpaired, or "independent samples" t-test is used when two separate independent and identically distributed samples are obtained, one from each of the two populations being compared. The paired t-test typically consist of a sample of matched pairs of similar units.

We use the t-test to determine whether the average precision and recall of certain web content extraction method is significantly different to the other. As we utilize the same dataset and the observations on the two populations of interest are collected in pairs, we use the paired t-test.

The null hypothesis for the is that the mean of first sample is equal to the mean of the second sample. If the calculated *p-value* is below the threshold chosen for statistical significance, usually **0.05**, then the null hypothesis is rejected.

Chapter 6

Experiments

In this section we conduct a series of experiments in order to examine the performance of our approach and compare it with the other existing heuristic approach. In general, we perform the experiments related to the classification tasks and web content extraction evaluation.

6.1 Dataset

The datasets that we used for our experiments are divided into three domains namely a news dataset, a blog dataset, and a forum dataset. We utilized Teezir's crawler to gather the web pages from the Internet. For each domain, we crawled several hosts and downloaded dozens of web pages. Table 6.1 shows the distribution of the amount of web pages for all three domains.

Table 6.1: Dataset specification

Dataset Type	Number of Source	Number of Pages
News	14	1889
Blog	80	1070
Forum	4	300

In particular, for the news dataset, we also used the dataset from [5], in which for every web page in the dataset the gold standard is available as a benchmark for the content extraction evaluation.

6.2 Experiment Design

Essentially, the experiment is conducted to select the best classifier model, combine the classification and our heuristic approach to perform web content extraction, evaluate the performance of our approach, and compare the performance to the other existing heuristic approaches.

As shown in Figure 6.1, first we divide the dataset into two parts namely the training dataset and the evaluation dataset. From the training dataset, we generate the labeled training instances by performing annotation to the web pages from the training dataset. The training instances will be used to train the learning algorithms for classification purpose. After that, we perform cross validation, feature selection, and measure the performance of the learning algorithms.

From this phase, we select one of the best learning algorithm and obtain the classifier models. Next, we apply the classifier models combined with our heuristic approaches to perform web content extraction to the evaluation dataset namely news, blog, and forum. Afterwards, we compute the Longest Common Subsequence [6] of the *extracted content* with the *gold standard* from which we may measure the precision, recall and F1 of our approach. The same web content extraction evaluation is also performed to the other existing approaches so that we can compare the performance of our approach with the existing heuristic approaches.

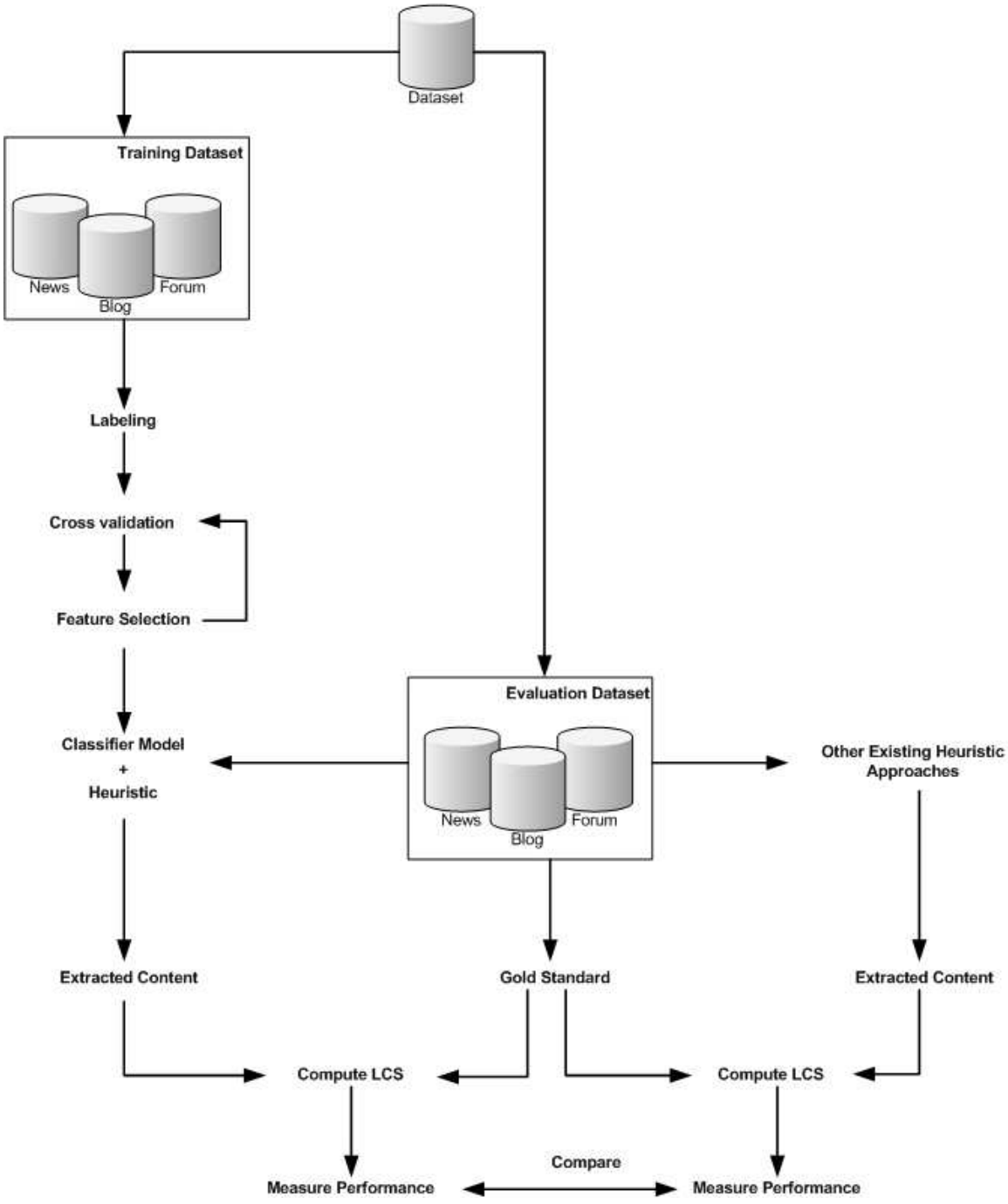

Figure 6.1: Overall experiment design

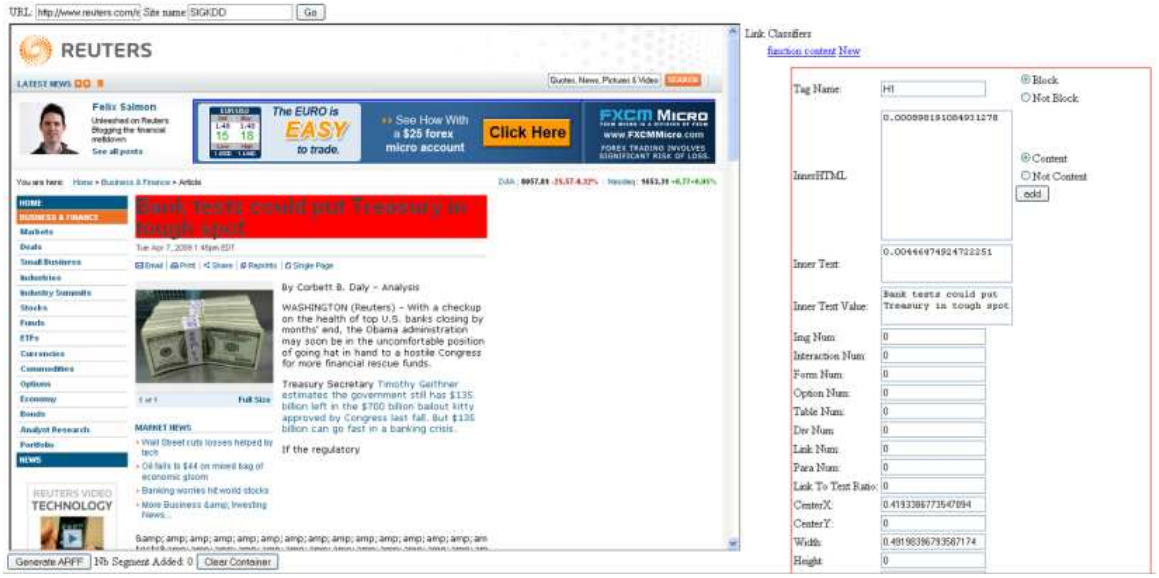

Figure 6.2: Labeling tool user interface

6.3 Generating Labeled Training Set

Our approach requires labeled training data; therefore we developed a web based tool so that users can label the area of the web page as a main content or a noisy content, a good segment or a bad segment. For each dataset type we take around 10% of the total pages as a training set and the rest as an evaluation set. The pages in the training set are taken from different sources which do not occur in the evaluation set e.g. if the evaluation set consist of Yahoo news pages, there will not be any Yahoo news pages in the training set.

Table 6.2: Dataset specification

Dataset Type	Number of Pages in Training Set	Number of Pages in Evaluation Set
News	160	1689
Blog	70	605
Forum	16	164

Figure 6.2 shows the user interface of the labeling tool. The area which is selected by the user will be highlighted. Upon selecting a certain area, the DOM properties of the selected area are retrieved and shown in the right part of the user interface. After the user finishes the labeling, he can export the labeled training data into ARFF data format which can be used by WEKA [11]. As shown in Table 6.3, the labeling activity produce two training sets namely for segment classification and content classification. These training sets will be used by the *two-phase classifier*. We can also see that there is a possibility that a training instance labeled as a *bad segment* and *main content*. This case is illustrated in Figure 6.3, from the figure we can see the area which are highlighted by rectangles. The smaller rectangle is the area which is usually enclosed by `p` tag, and we regard this as bad segment as we prefer the larger segment that is the larger rectangle which is enclosed by HTML structural elements such as `div` tag.

Table 6.3: Training instances distribution for two-phase classifier

Dataset Type	Segment Training Instance		Content Training Instance	
	Good Segment	Bad Segment	Main Content	Noisy Content
Blogs	904	111	504	511
News	702	313	127	888
Forum	592	86	253	425

As for *single-phase classifier*, we need to transform the existing training sets into a single training set which consists of binary class instances namely the *main content* and the *noisy content*. The mapping from the two-phase classifier training set into single-phase classifier training set is shown in Table 6.4. The distribution of the training instances is shown in Table 6.5.

In addition to that, we also need to label the exact text string from evaluation dataset that is expected to be extracted as main content (gold standard). In order

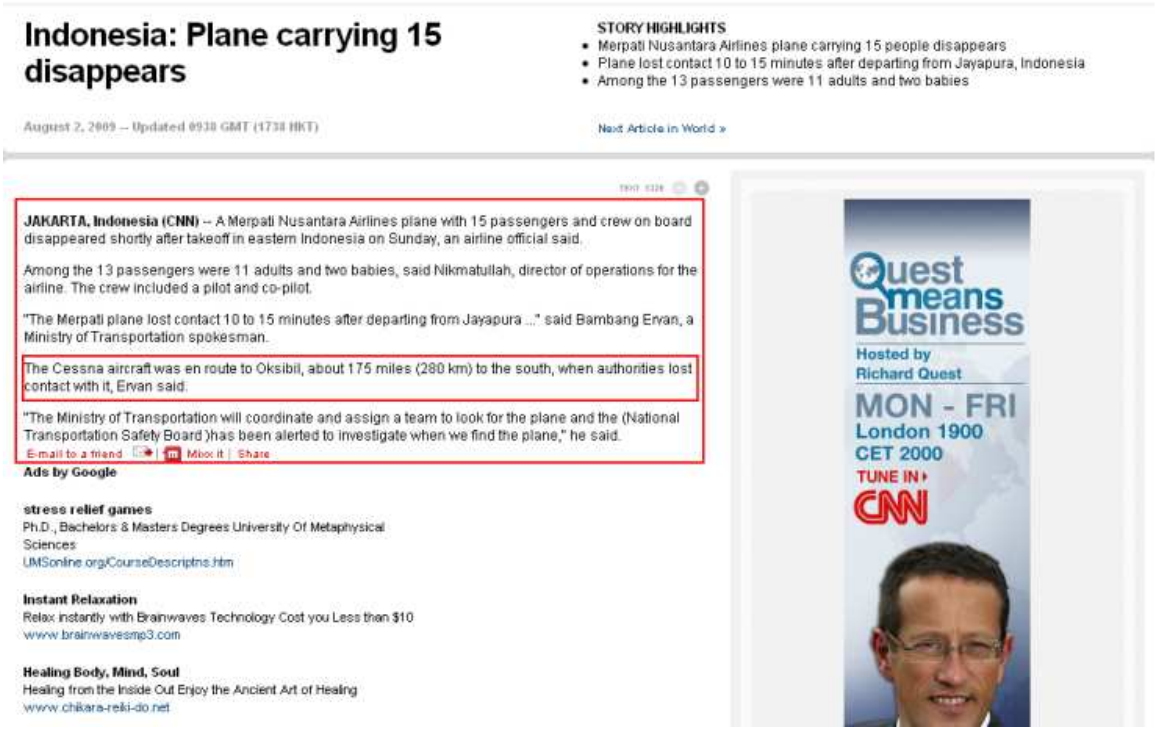

Figure 6.3: Example of a bad segment in a main content

Table 6.4: Mapping from two-phase classifier training set to single-phase classifier training set

Class labels in two-phase classifier	Class labels in single-phase classifier
Good Segment, Main Content	Main Content
Good Segment, Noisy Content	Noisy Content
Bad Segment, Main Content	Noisy Content
Bad Segment, Noisy Content	Noisy Content

to establish the gold standard, we wrote a manual extractor (written in C#) for each website source to extract the main content. Generally, by using this manual extractor, for each website type we assigned static DOM node targets which contain the main content so that the extractor is able to extract the exact text string which resembles the main content.

Table 6.5: Training instances distribution for single-phase classifier

Dataset Type	Training Instance	
	Main Content	Noisy Content
Blogs	471	544
News	124	891
Forum	252	426

6.4 Experiments

In this section we perform the experiments related to classification and feature selection. We experiment with several learning algorithms from WEKA [11] namely J48 Decision Tree, Multilayer Perceptron, Random Forest, and SMO. For the brief explanation regarding the learning algorithms see [17]. In addition to the classification experiments, we also perform the McNemar's Test to determine whether a classifier significantly outperforms the other.

6.4.1 Classification Experiments

In order to select the best learning algorithm for the classification process, we performed 10-fold cross validation to our labeled training instances. For the parameter values, we used the default parameter settings from WEKA.

Two-Phase Classifier

The results that we obtained for both segment and content classification in terms of precision, recall, and F1 measure are reasonably high.

For segment classification experiments, in general most of the learning algorithms perform reasonably well by achieving more than 90 % in terms of accuracy, precision, recall, and F1 measure as shown in Table 6.6. In order to determine if one classifier

Table 6.6: Segment classification result with two-phase classifier

Segment Classification-News								
Learning Algorithm	**tp**	**fn**	**fp**	**tn**	**accuracy**	**precision**	**recall**	**f1**
J48 Decision Tree	675	27	25	288	0.949	0.964	0.962	0.963
Random Forest	685	17	24	289	0.960	0.966	0.976	0.971
SMO	633	69	66	247	0.867	0.906	0.902	0.904
Multilayer Perceptron	680	22	42	271	0.937	0.942	0.969	0.955
Segment Classification-Blog								
J48 Decision Tree	881	23	44	67	0.934	0.952	0.975	0.963
Random Forest	893	11	34	77	0.956	0.963	0.988	0.975
SMO	895	9	92	19	0.900	0.907	0.990	0.947
Multilayer Perceptron	873	31	48	63	0.922	0.948	0.966	0.957
Segment Classification-Forum								
J48 Decision Tree	585	7	7	79	0.979	0.988	0.988	0.988
Random Forest	588	4	5	81	0.987	0.992	0.993	0.992
SMO	586	6	47	39	0.922	0.926	0.990	0.957
Multilayer Perceptron	582	10	7	79	0.975	0.988	0.983	0.986

significantly outperforms the other we performed the McNemar's Test [3].

For the segment classification in the news dataset, we can see from Table 6.7 that the learning algorithms namely J48, Random Forest, and Multilayer Perceptron are not significantly outperform each other. However, for SMO, when it compared to the other learning algorithm, the other learning algorithms significantly outperform SMO. The same thing happens for segment classification in the blog dataset and forum dataset as shown in Table 6.7, SMO is significantly outperformed by the other learning algorithms while J48, Random Forest, and Multilayer Percepton are not significantly outperform each other.

Regarding the content classification, in general, most of the learning algorithms also perform reasonably well for all measurements. However, the performance of SMO in terms of precision degrades below 90% in the blog dataset and forum dataset as

illustrated in Table 6.8. Table 6.9 provides the result of McNemar's test, for the news dataset the significant difference only occurs when Random Forest is compared to SMO. For the other algorithms in the news dataset namely Multilayer Perceptron and J48, according to McNemar's Test, there is no significant difference in performance. As for the blog dataset, SMO is outperformed by Multilayer Perceptron and Random Forest, also the decision tree is outperformed by Random Forest. For the forum dataset, the performance of Random Forest, Multilayer Perceptron,and J48 are not significantly different. As for SMO, in the forum dataset, it is outperformed by the other algorithms.

Table 6.7: Result of McNemar's test between classifiers in segment classification (+ *significantly outperforms*, = *tie*, - *significantly outperformed*)

Learning Algorithm	J48 Decision Tree	Random Forest	SMO	Multilayer Perceptron	Cumulative		
	news/blog/forum	news/blog/forum	news/blog/forum	news/blog/forum	Nb +	Nb =	Nb -
J48 Decision Tree	∅	=/=/=	+/=/+	=/=/=	2	7	0
Random Forest	=/=/=	∅	+/+/+	+/=/+	5	4	0
SMO	-/=/-	-/-/-	∅	-/=/-	0	2	7
Multilayer Perceptron	=/=/=	-/=/-	+/=/+	∅	2	5	2

Table 6.8: Content classification result with two-phase classifier

Content Classification-News							
Learning Algorithm	**tp**	**fn**	**fp**	**tn**	**accuracy**	**precision**	**recall**
decision tree(J48)	122	5	2	886	0.993	0.984	0.961
Random Forest	123	4	0	888	0.996	1.000	0.969
SMO	123	4	8	880	0.988	0.939	0.969
Multilayer Perceptron	125	2	1	887	0.997	0.992	0.984
Content Classification-Blog							
decision tree(J48)	474	30	40	471	0.931	0.922	0.940
Random Forest	489	15	32	479	0.954	0.939	0.970
SMO	479	25	99	412	0.878	0.829	0.950
Multilayer Perceptron	473	31	36	475	0.934	0.929	0.938
Content Classification-Forum							
decision tree(J48)	248	5	11	414	0.976	0.958	0.980
Random Forest	248	5	7	418	0.982	0.973	0.980
SMO	233	20	78	347	0.855	0.749	0.921
Multilayer Perceptron	248	5	13	412	0.973	0.950	0.980

Table 6.9: Result of McNemar's test between classifiers in content classification (+ *significantly outperforms*, = *tie*, - *significantly outperformed*)

Learning Algorithm	J48 Decision Tree	Random Forest	SMO	Multilayer Perceptron	Cumulative		
	news/blog/forum	news/blog/forum	news/blog/forum	news/blog/forum	Nb +	Nb =	Nb -
J48 Decision Tree	∅	=/-/=	=/=/+	=/=/=	1	7	1
Random Forest	=/+/=	∅	+/+/+	=/=/+	5	4	0
SMO	=/=/-	-/-/-	∅	=/-/-	0	3	6
Multilayer Perceptron	=/=/=	=/=/-	=/+/+	∅	2	6	1

Single-Phase Classifier

We also tried to perform the classification by using single-phase classifier. With the single-phase classifier we only define two class labels namely *main content* and *noisy content*.

As Table 6.10 shows, the algorithms namely J48, Random Forest, and Multilayer Perceptron achieve reasonably high score of precision and recall for all measurements across datasets. The difference of performance in terms of precision and recall among those algorithms is relatively small. As for SMO, its performance is deteriorating mainly in terms of recall in the forum dataset and blog dataset.

From the McNemar's Test result in Table 6.11 we can see that the algorithms namely J48, Random Forest, and Multilayer Perceptron are rarely significantly outperform each other. Multilayer Perceptron is only once outperformed by Random Forest in the forum dataset. As for SMO, in almost all comparison, it is outperformed by other algorithms mainly by Random Forest and Multilayer Perceptron.

Table 6.10: Classification result by using single-phase classifier

News							
Learning Algorithm	**tp**	**fn**	**fp**	**tn**	**accuracy**	**precision**	**recall**
J48 Decision Tree	888	3	3	121	0.994	0.997	0.997
Random Forest	890	1	5	119	0.994	0.994	0.999
SMO	885	6	5	119	0.989	0.994	0.993
Multilayer Perceptron	890	1	2	122	0.997	0.998	0.999
Blog							
J48 Decision Tree	438	33	33	393	0.926	0.930	0.930
Random Forest	453	18	26	400	0.951	0.946	0.962
SMO	378	93	25	401	0.868	0.938	0.803
Multilayer Perceptron	434	37	33	393	0.922	0.929	0.921
Forum							
J48 Decision Tree	412	14	5	247	0.972	0.988	0.967
Random Forest	416	10	4	248	0.979	0.990	0.977
SMO	350	76	20	232	0.858	0.946	0.822
Multilayer Perceptron	410	16	7	245	0.966	0.983	0.962

Table 6.11: Result of McNemar's test between classifiers for single-Phase classifier (+ *significantly outperforms*, = *tie*, - *significantly outperformed*)

Classifier	J48 Decision Tree	Random Forest	SMO	Multilayer Perceptron	Cumulative		
	news/blog/forum	news/blog/forum	news/blog/forum	news/blog/forum	Nb +	Nb =	Nb -
J48 Decision Tree	∅	=/=/=	=/+/+	=/=/=	2	7	0
Random Forest	=/=/=	∅	+/+/+	=/=/+	4	5	0
SMO	=/-/-	-/-/-	∅	-/-/-	0	1	8
Multilayer Perceptron	=/=/=	=/=/-	+/+/+	∅	3	5	1

6.4.2 Feature Selection

In the previous experiments, we used all the features to perform web page segmentation and segment classification. It is possible to reduce the number of features used in the classification task by applying feature selection. Feature selection is a common technique used in machine learning to select a subset of relevant features for building robust learning models. Feature selection has several advantages namely alleviating the effect of curse of the dimensionality, speeding up the learning process, and enhancing the generalization capability [17].

Regarding the feature selection methods, there are numerous methods available in WEKA. Hall [16] provides comparison and experiments of feature selection methods in WEKA namely *Information Gain (IG)*, *Relief (RLF)*, *Principal Component (PC)*, *Correlation-based Feature Selection (CFS)*, *Consistency-based Subset Evaluation (CNS)*, and *Wrapper Subset Evaluation (WRP)*. We followed his experiment approach by using WEKA Experimenter tool, utilizing the *Attribute Selected Classifier*, and applying the J48 learning algorithm. We compared the performance of the feature selection methods and measure the statistical significance. In addition, a pairwise comparison is made between each method and all of the others. The methods are ranked by the total number of "wins" minus "losses".

The results of the rank comparison between feature selection methods for two-phase classifier and single-phase classifier are shown in Table 6.12 and Table 6.13. According to the results, *IG* is the best feature selection method in two-phase classifier dataset and single-phase classifier dataset. Based on this comparison we decided to use *IG* as the feature selection methods.

Table 6.12: Wins versus losses for accuracy of feature selection in two-phase classifier dataset with J48

Resultset	Wins− Losses	Wins	Losses
IG	9	9	0
CNS	8	8	0
RLF	8	8	0
CFN	7	8	1
PC	-6	6	12
WRP	-36	0	36

For the details of the accuracy obtained for the feature selection method in each dataset and also its statistical significance see Table B.1 and Table B.2 in Appendix B.1.

Table 6.13: Wins versus losses for accuracy of feature selection in single-phase classifier dataset with J48

Resultset	Wins−Losses	Wins	Losses
IG	5	5	0
CFN	5	5	0
CNS	5	5	0
RLF	5	5	0
PC	-7	3	10
WRP	-18	0	18

Two-Phase Classifier

As the results from the feature selection for the segment training instance, the features such as *innerHtmlLength*, *domHeight*, and *divNum* dominate among all datasets. As for the content training instance, the dominant features are *stopWordRatio, stringMax, paraNum* and *innerTxtLength*. There is no crucial change of classification result after the feature selection. For the details of the feature selection and the classification result see Appendix B.2.

Single Phase Classifier

The results of the feature selection the features such as *stopWordRatio*, *innerTxtLength*, *domHeight*, and *linkTextRatio* dominate among the three datasets. Similar with the two-phase classifier there is no crucial change of classification result after the feature selection. For the details of the feature selection and the classification result see Appendix B.3.

6.4.3 Summary of Classification Experiment

In the previous experiment in Section 6.4.1 we have experimented with the two-phase classifier and single-phase classifier. In general, for the classification tasks performed by all the learning algorithms the results are encouraging as for almost all measurements the scores are more than 90%.

According to the statistical significance test, the performance of J48 Decision Tree, Multilayer Perceptron, and Random Forest is relatively the same. From those three algorithms, there is no single algorithm that significantly outperforms all the others. However, exception for SMO, in most of the cases it's performance is significantly surpassed by other learning algorithms.

In Section 6.4.2, we performed feature selection and we can see that J48 yields relatively compact decision tree and relatively easy to interpret. As examples, Figure 6.4 and Figure 6.5 illustrated the generated decision tree from the news dataset. As we can see from Figure 6.4 for the segment classification, one can interpret whether a segment is a good segment or bad segment by checking the parameter values of the strong features in segment classification such as *innerHtml*, *domHeight*, and *divNum*. In the Figure 6.5, similar with segment classification, we can see that identifying the main content can be done by checking only the strong features such as *stopWordRatio*, *stringMax*, *paraNum*, *innerTxtLength* and *linkTextRatio*.

As for the value of the features, it is normalized therefore the value will be between 0 and 1.

In terms of single-phase and two-phase classifier, it is difficult to claim that two-phase classifier is better than single-phase classifier in this stage. We will make the

```
InnerHTMLLength <= 0.007179
|   DivNum <= 0: 0 (265.0/25.0)
|   DivNum > 0
|   |   LinkNum <= 0.009804
|   |   |   DomHeight <= 0.1: 1 (15.0/1.0)
|   |   |   DomHeight > 0.1: 0 (2.0)
|   |   LinkNum > 0.009804: 0 (23.0/1.0)
InnerHTMLLength > 0.007179
|   InnerHTMLLength <= 0.404013
|   |   DomHeight <= 0.428571: 1 (619.0/16.0)
|   |   DomHeight > 0.428571
|   |   |   innerTextLength <= 0.013714: 0 (6.0)
|   |   |   innerTextLength > 0.013714
|   |   |   |   innerTextLength <= 0.053747
|   |   |   |   |   InnerHTMLLength <= 0.080251: 1 (6.0)
|   |   |   |   |   InnerHTMLLength > 0.080251: 0 (2.0)
|   |   |   |   innerTextLength > 0.053747: 1 (44.0)
|   InnerHTMLLength > 0.404013
|   |   DivNum <= 0.491713: 0 (15.0)
|   |   DivNum > 0.491713
|   |   |   LinkNum <= 0.509203: 1 (8.0)
|   |   |   LinkNum > 0.509203: 0 (10.0/1.0)
```

Figure 6.4: J48 Tree for segment classification

direct comparison between single-phase and two-phase classifier in the final web content extraction evaluation. Based on the experiments that we have done, we decided to incorporate J48 Decision Tree algorithm in the web content extraction evaluation instead of Multilayer Perceptron and Random Forest as the J48 Decision Tree is simpler to implement and provides satisfying classification result in all measurements. In particular, for the implementation we use a variant of decision tree algorithm that is available in Teezir Framework, implemented in C#, which is comparable to that of J48 Decision Tree. The size of the tree generated by Teezir Decision Tree is larger than J48 Decision Tree, however the performance in the classification task is relatively satisfying. For the details of the classification results of Teezir Decision Tree see Appendix C.

```
StopWordRatio <= 0.198529
|   StopWordRatio <= 0.063686: 0 (838.0)
|   StopWordRatio > 0.063686
|   |   StringMax <= 0.833616: 0 (24.0/1.0)
|   |   StringMax > 0.833616
|   |   |   ParaNum <= 0.285714: 1 (3.0)
|   |   |   ParaNum > 0.285714: 0 (2.0)
StopWordRatio > 0.198529
|   LinkTextRatio <= 0.081448: 1 (104.0)
|   LinkTextRatio > 0.081448
|   |   innerTextLength <= 0.481852: 1 (17.0)
|   |   innerTextLength > 0.481852: 0 (27.0/2.0)
```

Figure 6.5: J48 Tree for content classification

6.5 Cost Sensitive Learning

In general machine learning settings, the classifier usually try to minimize the number of errors they will make in dealing with unseen instances. This kind of setting is valid if the costs of all possible errors are the same. However, in many real world applications the costs of misclassification errors are often unequal. For example, in the medical diagnosis, the cost of erroneously diagnosing a patient to be healthy is much bigger than mistakenly diagnosing a healthy person to be sick, because the former kind may lead to loss of a life. In addition, cost sensitive learning is often used to a dataset with very imbalanced class distribution [12]. Usually, the rare class is the positive class and it is often more expensive to misclassify an actual positive class into negative rather than actual negative into positive. Therefore, the cost of *false negative (FN)* is usually larger than that of *false positive (FP)*.

In our training data set, there is also class imbalance for example in the news training instances for content classification as shown in Table 6.3. The positive class (main content) is about 10 % of the total training instance.

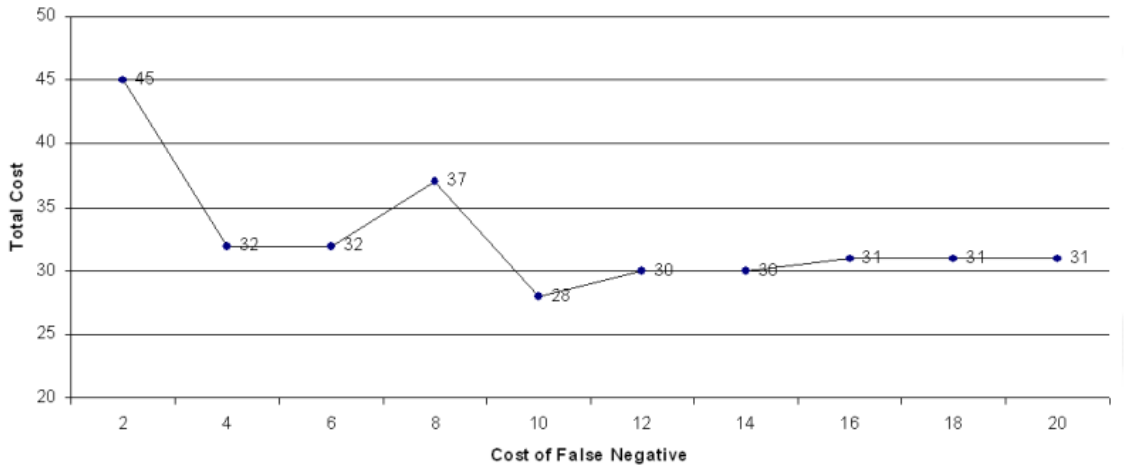

Figure 6.6: Misclassification cost for various CFN at CFP = 1

In order to apply cost sensitive learning, we have to setup the cost settings for FP (CFP) and FN (CFN). For applying the cost sensitive learning, we used the *MetaCostSensitiveClassifier* from WEKA and applied it to the news training instance. As the classifier we used the J48 algorithm with default parameter settings. We experimented with a series of cost ratios, we keep the CFP at the value 1 and try different values on CFN. As shown in Figure 6.6 the CFN interval from 2 until 20, the minimum misclassification cost is when CFN equals to 10. It also can be seen that, the ratio between CFP and CFN which produce the minimum cost is 1:10 and apparently this is also the class ratio between positive class (main content) and negative class (noisy content).

We tried to apply cost parameter settings namely $CFP = 1$, $CFN = 10$, it can be seen from Table 6.14 and Table 6.15, we can boost the recall as the false negative

decreases from 11 to 2. However, there is a slight degrade in terms of precision score as the false positive increases.

Table 6.14: Classification confusion matrix without cost sensitive learning

Actual Class	Predicted Class	
	Main Content	Noisy Content
Main Content	125	2
Noisy Content	11	877

Table 6.15: Classification confusion matrix with cost sensitive learning

Actual Class	Predicted Class	
	Main Content	Noisy Content
Main Content	122	5
Noisy Content	2	886

In our case this means that the classifier will perform better on classifying main content however it will also include more noisy content. We think that precision and recall are both important in overall web content extraction process. However, in the content classification task, we think that it is fine to get a slightly lower precision. The most important thing in the classification task is that we can get the main content segment (high recall). In order to boost the precision, in the later stage of the content extraction process we can improve it by using heuristic approaches in the extracted segments.

6.6 Web Content Extraction Evaluation

Until the previous section, we only experimented with the classification tasks in order to get the grasp of the classifier performance namely for two-phase classifier and single phase classifier. In this section we want to evaluate our approach when applied to

our goal that is web content extraction. In particular, we named our approach as *Content Seeker (CS)*.

6.6.1 Evaluation Setup

As described in Section 6.3, we divided our dataset into training dataset and evaluation dataset. We performed 10-fold cross validation in the training dataset by using Teezir Decision Tree to obtain the classifier model for each dataset namely news, blog, and forum. After that we performed the web content extraction to the evaluation dataset.

As illustrated in Figure 6.7, the evaluation process is straightforward. We stored the raw HTML and the *gold standards* in the database and then for each document in the evaluation dataset we parse the HTML code, obtain the DOM tree, perform the classification and heuristic approach and finally obtain the *extracted content*. We computed the Longest Common Subsequence (LCS) [6] of the extracted content and the gold standard which can be used to measure the precision, recall, and F1 as described in Section 5.2.

As a benchmark, we compared our approach with the existing content extraction approaches that we presented in Chapter 3 namely *Content Code Blurring (CCB)*, *K-Feature Extractor (KFE)*, *Feature Extractor (FE)*, *Document Slope Curve (DSC)*, *Crunch*, *Largest Pagelet(LP)* and *Link Quota Filter (LQF)*. As for the implementation of these existing heuristic approaches, we obtained the implementation library from Gottron [5].

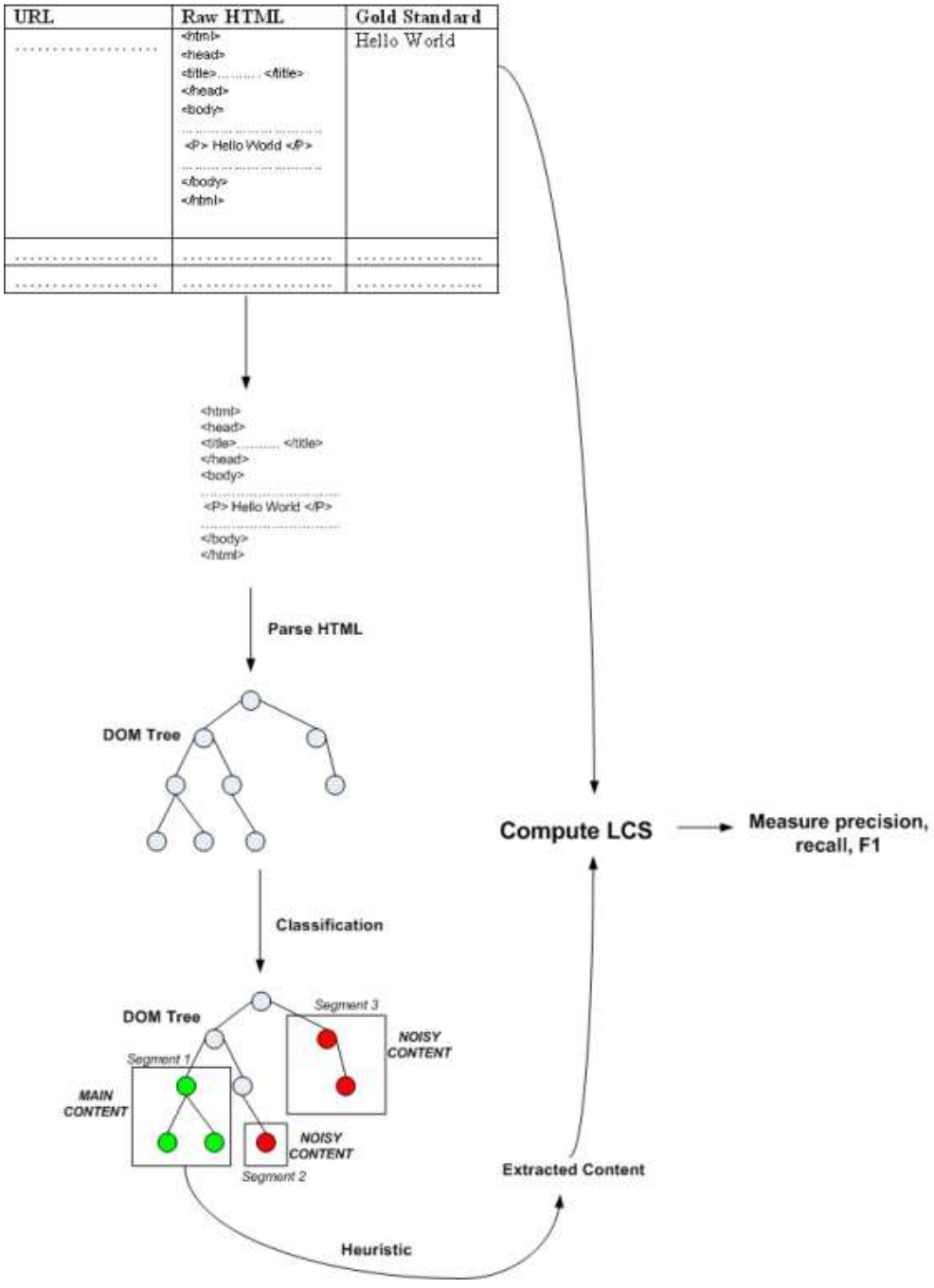

Figure 6.7: Overall web content extraction evaluation process

6.6.2 Classifier Configuration

As listed in Table 6.16 there are several configurations of *CS* that we used. We experimented with the *two-phase classifier* and *single-phase classifier* along with our own developed heuristic approaches namely *Largest Block of String (LBS)*, *String Length Smoothing (SLS)*, and *Table Pattern (TP)*.

Table 6.16: Content seeker configurations

Configuration	Description
CS+plain	Content Seeker with two-phase classifier
CS+LBS	Content Seeker with two-phase classifier combined with LBS heuristic
CS+SLS	Content Seeker with two-phase classifier combined with SLS heuristic
CS+TP	Content Seeker with two-phase classifier combined with TP heuristic
CS+1+LBS	Content Seeker with single-phase classifier combined with LBS heuristic
CS+1+SLS	Content-seeker with single-phase classifier combined with SLS heuristic
CS+1+TP	Content-seeker with single-phase classifier combined with TP heuristic

We used the strong features from feature selection that we described in Section 6.4.2. For the decision tree the parameter settings we used $confidencefactor = 0.25$ and $minobj = 2$. We decided to use these parameter values as there is no important changes of precision and recall when we tried to do the classification task for various number of features and *minobj* values (for details, see Appendix A).

In addition, to overcome the problem of overfitting, we also put a threshold Δ for the error rate in the nodes of the decision tree. As overfit occurs, the classifier will not return anything as main content. If this happens, we use this threshold so that the classification decision will take place in the upper part of the tree. In this

evaluation we used $\Delta = 0.2$.

6.6.3 News Data Set Evaluation

We applied our approach to the news dataset and compared the result to the other existing content extraction methods. As the library for Crunch is not available, it is not possible to provide the standard deviation. We obtained the average precision for Crunch from [5].

Table 6.17: Average precision, recall, F1, and standard deviation for news dataset evaluation

	Precision	Recall	F1
Our Approach			
CS+plain	0.856±0.260	0.883±0.236	0.864±0.244
CS+LBS	**0.940±0.183**	0.882±0.187	**0.905±0.182**
CS+SLS	0.918±0.208	0.741±0.188	0.785±0.220
CS+1+LBS	0.939±0.185	0.882±0.188	0.905±0.184
CS+1+SLS	0.705±0.290	0.769±0.194	0.710±0.227
Existing Heuristic Approach			
TCCB-25	0.806±0.228	0.859±0.198	0.816±0.197
CCB-r40	0.796±0.286	0.753±0.285	0.763±0.273
ACCB-r40	0.582±0.225	0.954±0.092	0.699±0.201
LP-7	0.695±0.283	0.809±0.234	0.728±0.262
LQF-75	0.571±0.192	0.957±0.109	0.698±0.177
LQF-50	0.597±0.195	0.954±0.114	0.717±0.177
LQF-25	0.606±0.197	0.606±0.197	0.606±0.197
KFE	0.488±0.297	0.687±0.422	0.502±0.351
FE	0.767±0.234	0.140±0.131	0.218±0.178
DSC	0.863±0.167	0.870±0.136	0.852±0.141
Crunch	0.65±0.260	**0.964**	0.714

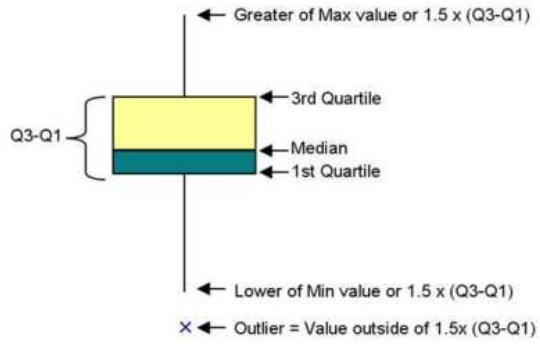

Figure 6.8: The box-whisker diagram

Precision

As shown in Table 6.17, in terms of average precision, our approach namely *CS+LBS* obtain the best score among the other existing heuristic methods. The other existing methods, that yield the highest average precision is *DSC*. In order to understand the *spread* and *skewness* of the precision scores for each content extraction method, we plot the results into *box-whisker* diagram. Figure 6.8, describes the variation and central tendency of the precision scores. The box represents the distance between the 1st (Q1) and 3rd quartiles (Q3) and the whisker show the highest and lowest data points or 1.5 times the box (Q3-Q1). Q1 is the cut off lowest 25% of data and Q3 is the cut off highest 25% of data.

The resulting box-whisker plot of the precision score for all the content extraction method is illustrated in Figure 6.9. From the box plot we can see that our approach namely *CS+LBS, CS+SLS,* and *CS+1+LBS* perform the best as the precision scores concentrates on 98% which is reasonably high and moreover, the skew is narrow. As for other existing heuristic methods, the skew of the precision score is relatively large

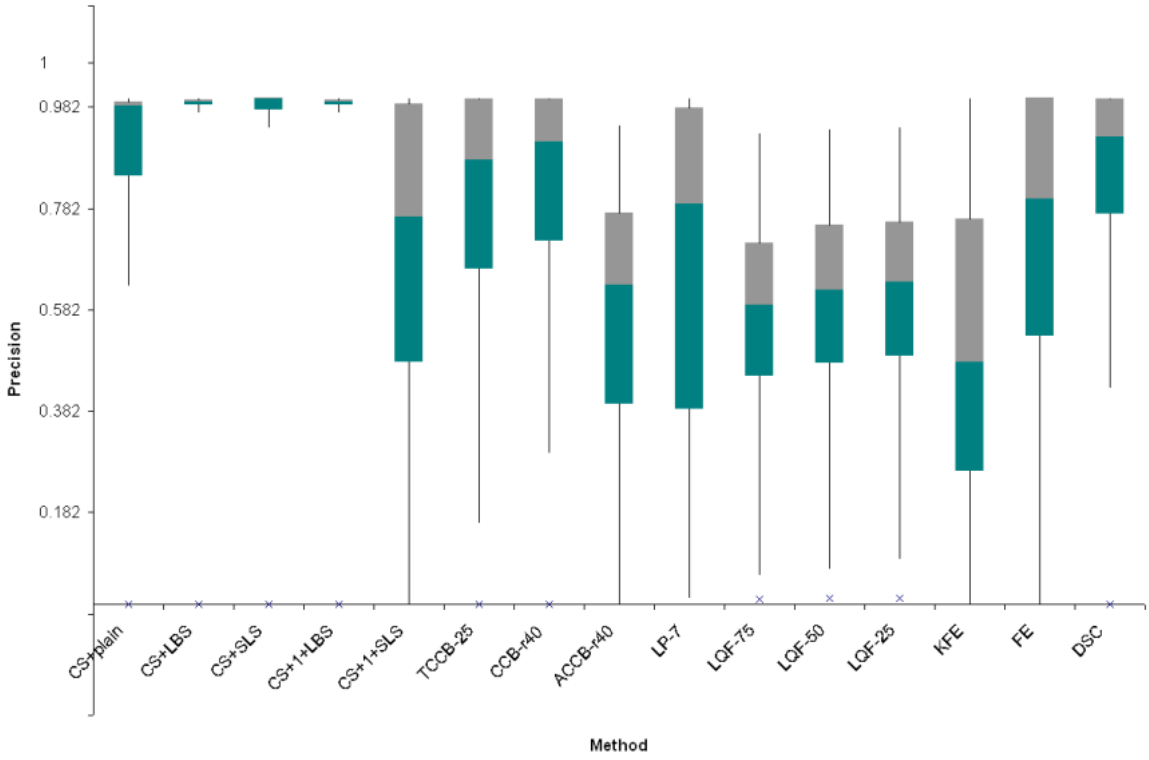

Figure 6.9: Variation and central tendency of precision for all extraction methods in the news dataset

and have wider spread range. Generally, from the existing methods we can observe that our close competitors are *CCB-r40* and *DSC*.

Essentially, the main competitor of *CS+LBS* is *CS+1+LBS*. Although the average precision of *CS+LBS* is higher and the standard deviation is smaller, it is difficult to claim that *CS+LBS* is better than *CS+1+LBS*. We performed a statistical significance test, *paired t-test*, between both approaches in order to understand whether the average precisions of both approaches are significantly different. The result of the t-test is shown in Table 6.18. The *p-value* is larger than the statistical significance threshold therefore we cannot reject the null hypothesis that the average precisions

are equal. Based on this analysis, we can say that *CS+1+LBS* is also a good choice for content extraction in terms of precision in addition to CS+LBS as they are not significantly different.

Table 6.18: Paired t-test between *CS+LBS* and *CS+1+LBS*, threshold $\alpha = 0.05$

	CS+LBS	*CS+1+LBS*
mean	0.940	0.939
p-value	**0.239**	

Recall

As for recall, as shown in Table 6.17 our approach namely *CS+LBS* is not the best in average recall. However, it still produces reasonable high score around 88% and the standard deviation is relatively small. The *LQF* methods namely *LQF-75* and *LQF-50* performs really well. The same thing happens for *Crunch* which obtains the highest average recall, however as mentioned before, we cannot obtain the standard deviation value. Although *LQF* and *Crunch* yield high score of average recall, they only produce low precision, less than 60% in average. Figure 6.10 provides the variation and the spread of recall scores for all the extraction methods. We can see from the box plot that *LQF-75* and *LQF-50* have recall scores that concentrate on the area which gives more than 90% recall. Moreover, the spread and skew of the recall scores of *LQF-75* and *LQF-50* is very narrow.

We performed the *paired t-test* for our approach namely *CS+LBS* with the *LQF-75*, the result is shown in Table 6.19. We can reject the null hypothesis that the average recall of both approaches are the same. Therefore, we may safely assume that the average recall for both methods are significantly different.

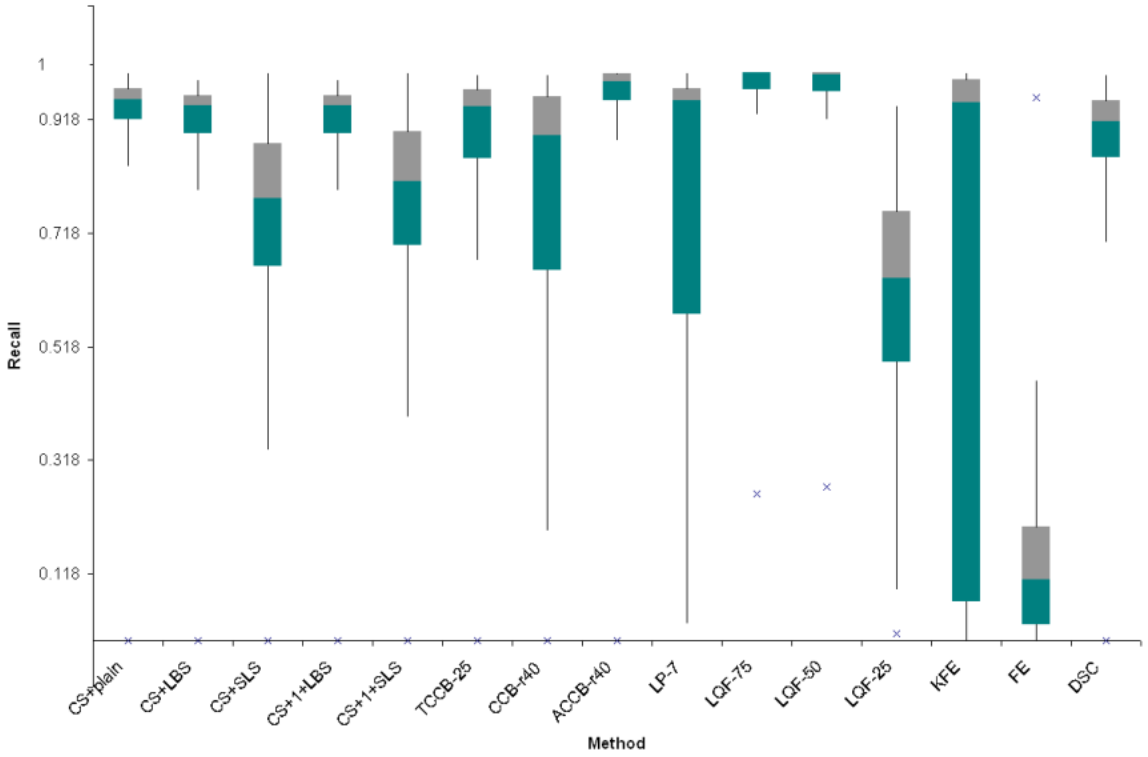

Figure 6.10: Variation and central tendency of recall for all extraction methods in the news dataset

Table 6.19: Paired t-test CS+LBS and LQF-75

	CS+LBS	*LQF-75*
mean	0.882	0.957
p-value	**4.2335E-54**	

F1

As our approach has the best average precision and relatively high recall, our approach namely *CS+LBS* managed to be the best in average F1 measure as shown in Table 6.17.

The spread of the F1 values for each methods are shown in Figure 6.11. We can see that again the difference between *CS+LBS* and *CS+1+LBS* is not significant as

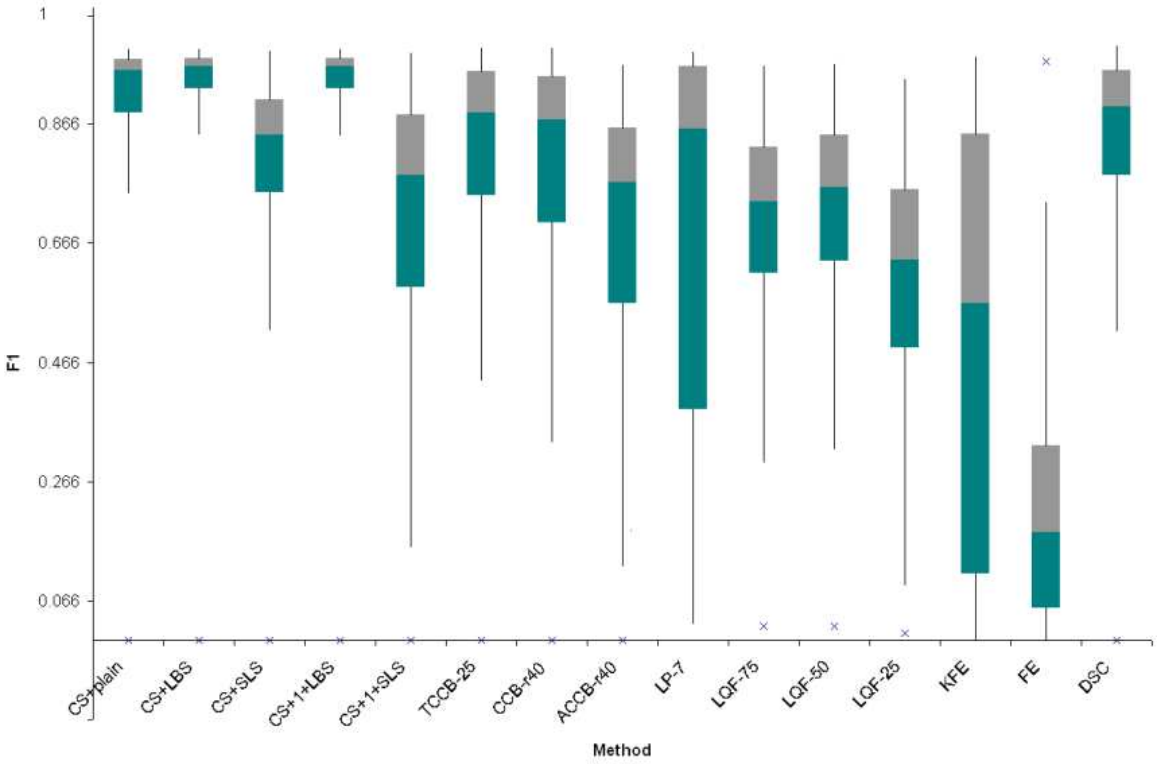

Figure 6.11: Variation and central tendency of F1 for all extraction methods in the news dataset

shown in the paired t-test result in Table 6.20.

Table 6.20: Paired t-test *CS+LBS* and *CS+1+LBS*, threshold $\alpha = 0.05$

	CS+LBS	*CS+1+LBS*
mean	0.9054	0.9050
p-value	**0.318**	

Applying Other Classifier Models to The News Dataset

We also tried to apply other classifier model namely blogs and forum classifier models to our news dataset. As for blog classifier model, it can produce quite high precision that is more than 90% as shown in Table 6.21. In fact, the precision is higher than *CS+1+LBS* that uses news classifier model. However, the recall is not as high as that of news classifier model.

For forum classifier model, as expected, it yields low precision and recall. This is understandable since the structure of forum web pages are very different with the news web pages.

Table 6.21: Results of applying other classifier models to the news dataset

Learning Algorithm	Precision	Recall	F1
CS+LBS+blog classifier model	0.949±0.146	0.806±0.275	0.841±0.261
CS+LBS+forum classifier model	0.397±0.465	0.180±0.331	0.202±0.353

6.6.4 News Dataset Evaluation Discussion

As we have seen from the web content extraction evaluation in the news dataset, our approach namely *CS+LBS* and *CS+1+LBS* are the best method since they achieve high precision and F1 measures compared to the other methods. However, if we observe the Figure 6.9, we may also see that there are outliers result for many of the extraction methods, even for *CS+LBS* and *CS+1+LBS*. In these outliers, the methods cannot extract the main content properly. We found that, for a very short main content with less stop words, it is difficult to extract the main content. The examples of this case are illustrated in Figure 6.12. The *stopWordRatio* of the area in the rectangle is lower than the threshold learned by the decision tree from the

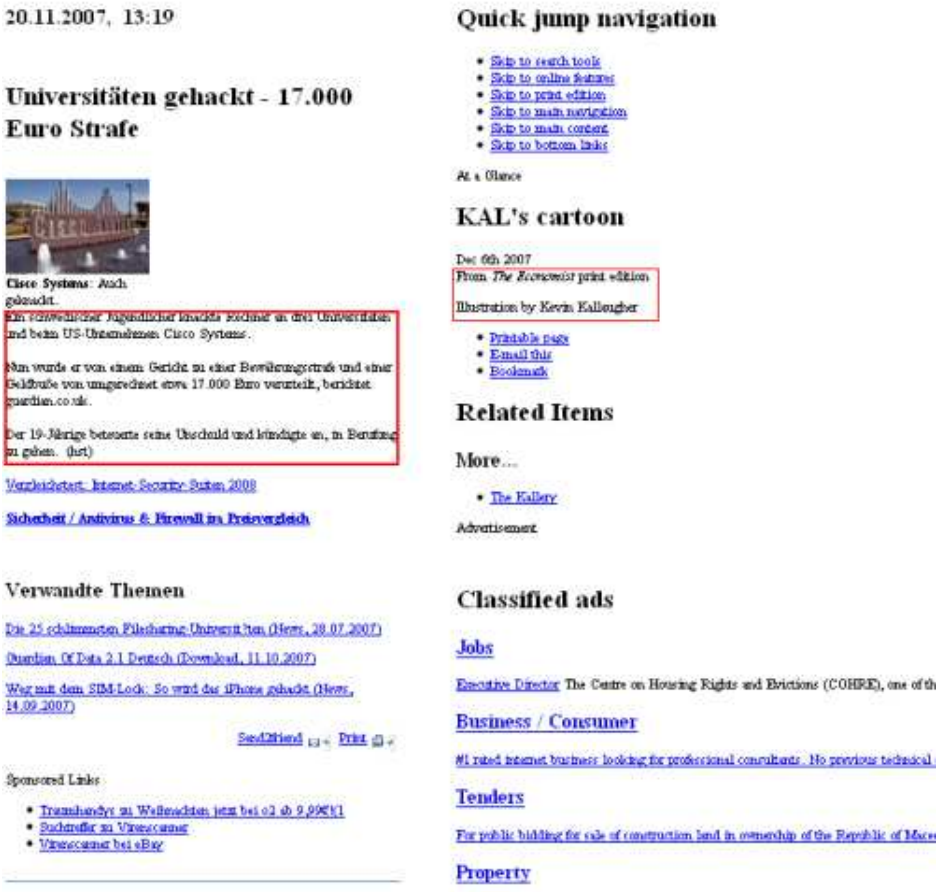

Figure 6.12: Examples of difficult cases

training dataset. Another case that makes the precision lower is the presence of embedded noisy contents such as related links inside the main content, embedded advertisements.

As for LQF variants, although it has very high scores of recall there are cases which they cannot handle. For example the structure of the page shown in Figure 6.13, the hyperlinks are included as part of the gold standard. However, LQF will remove all the hyperlinks, thus degrades the recall score significantly.

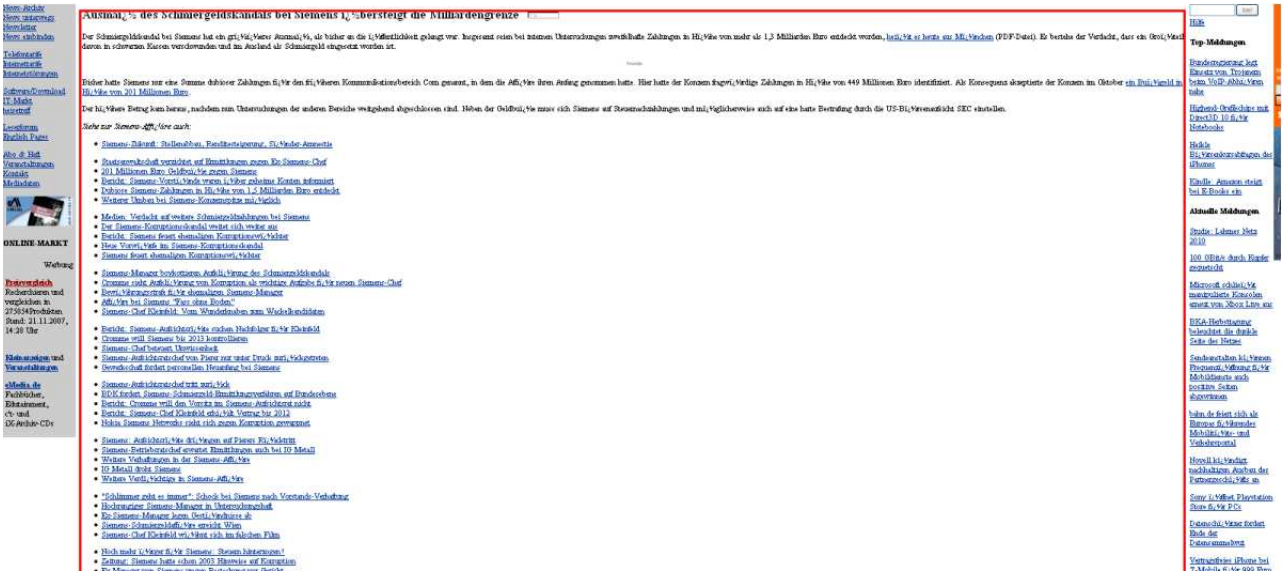

Figure 6.13: High ratio of hyperlinks inside the main content

In general, the existing heuristic methods that generate high score of recall namely *LQF* and *Crunch* only yields low precision score. This is because *LQF* and *Crunch* mainly remove the text strings that have high ratio of hyperlinked text. Therefore, in most of the cases it will obtain the main content but with fair amount of noisy contents such as advertisements that have quite high inner text.

FE in particular shows a remarkably bad performance in recall. Looking at the algorithm may explain the result. The algorithm looks for DOM node in the document which are particularly "pure" with respect to a certain desired feature. In our context, the desired feature is text. So the algorithm will look for those containing preferably only text. However, the main content in the web page is very likely consist of several paragraphs, contains hyperlinks, simple formatting to change the size and style of the writing. These structures cause the DOM node to be less pure, thus, might cause

the DOM node to be discarded. However, as shown in Figure 6.10, there are outliers where *FE* yields high score of recall. The example of these cases are the document which the DOM node of the main content only consist of one paragraph and does not contain complex formatting. Therefore, *FE* may extract this DOM node as it is "pure" with respect to the text feature.

6.6.5 Blogs Data Set Evaluation

For the blog dataset evaluation, we also compared the result of our approach to the other existing heuristic methods. However, we cannot present the result from Crunch since the implementation library is not available.

Table 6.22: Average precision, recall, F1, and standard deviation in the blog dataset

	Precision	Recall	F1
Our Approach			
CS+plain	0.015±0.057	**0.994**±0.071	0.026±0.074
CS+LBS	**0.905**±0.223	0.827±0.293	0.834±0.279
CS+SLS	0.825±0.310	0.790±0.306	0.779±0.308
CS+1+LBS	0.895±0.236	0.843±0.272	**0.844**±0.267
CS+1+SLS	0.430±0.300	0.875±0.211	0.768±0.515
Existing Heuristic Approach			
TCCB-25	0.566±0.347	0.907±0.200	0.634±0.307
CCB-r40	0.593±0.352	0.878±0.234	0.648±0.311
ACCB-r40	0.238±0.190	0.954±0.116	0.349±0.218
LP-7	0.503±0.377	0.777±0.228	0.554±0.351
LQF-75	0.268±0.195	0.983±0.109	0.386±0.220
LQF-50	0.296±0.215	0.981±0.111	0.416±0.236
LQF-25	0.305±0.222	0.969±0.130	0.423±0.240
KFE	0.482±0.236	0.717±0.355	0.486±0.254
FE	0.573±0.218	0.203±0.233	0.234±0.190
DSC	0.299±0.272	0.882±0.200	0.396±0.269

Precision

In terms of average precision, as shown in Table 6.22, our approach namely *CS+LBS* yields the best average precision. We can also see that the *CS+plain* give very low precision.

If we observe the spread of the precision score in Figure 6.14, although *CS+LBS* has the highest precision score, actually the other variants of content-seeker namely *CS+1+LBS* and *CS+SLS* have similar distribution of precision score. For the other existing methods, there is a small intersection of precision score area of *TCCB-25* with *CS+LBS*. We performed the paired t-test between *CS+LBS* with namely *CS+SLS*, *CS+1+LBS*, and *TCCB-25*.

Table 6.23 provides the *p-value* from the paired t-test. From the result we can see that the average precision of *CS+LBS* is significantly different with that of *CS+SLS* and *TCCB-25* as the p-value is less than the threshold. However, we cannot show that the average precision of *CS+1+LBS* is significantly different with *CS+LBS*. Again we can see that the performance of single-phase classifier, *CS+1+LBS*, and two-phase classifier, *CS+LBS*, has no significant difference.

Table 6.23: P-values from t-test, threshold $\alpha = 0.05$

	CS+SLS	*CS+1+LBS*	*TCCB-25*
CS+LBS	1.59758E-17	2.255	4.4206E-69

Recall

In terms of recall, *CS+plain* obtains the highest average recall score and very small standard deviation as shown in Table 6.22. Therefore, in the box plot illustrated in Figure 6.15, the plot is only a single dot as the distance between the first quartile

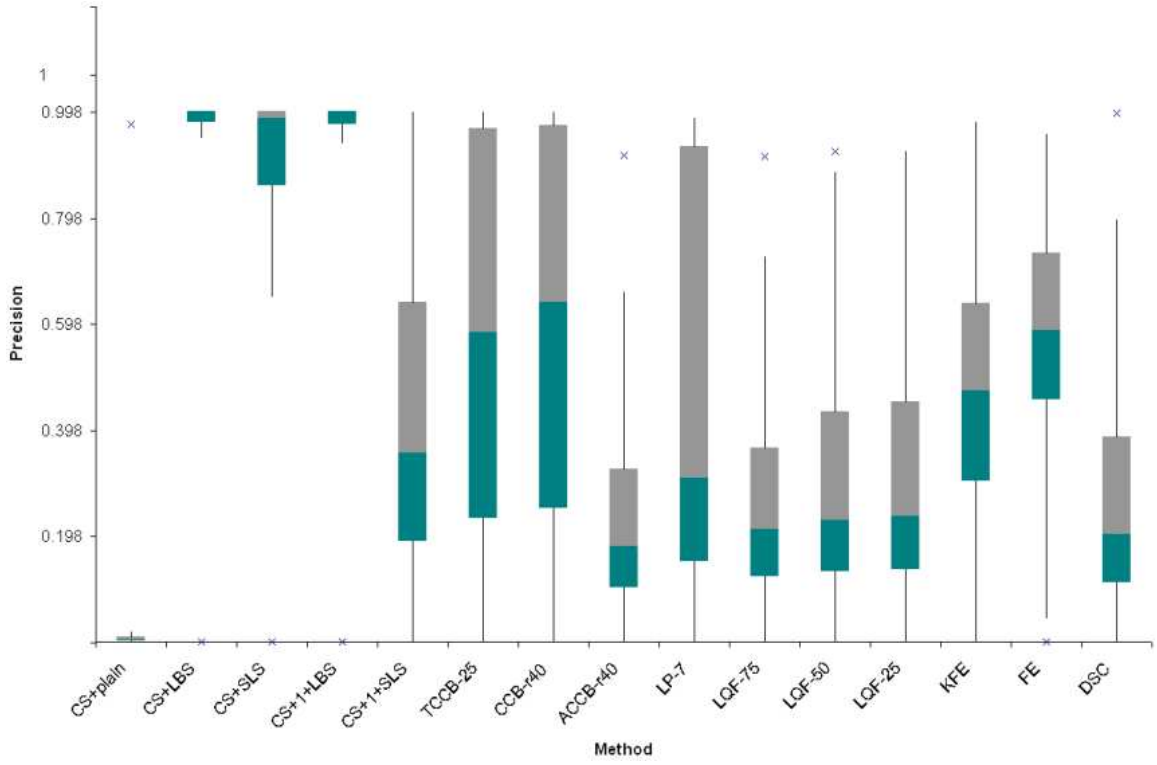

Figure 6.14: Variation and central tendency of precision for all extraction methods in the blog dataset

and third quartile is extremely small. From the box plot we can also observe that the nearest competitor of *CS+plain* are *LQF-75, LQF-50,* and *LQF-25*. We conducted the t-test between *CS+plain* and *LQF* variants, the result is shown in Table 6.24. From the result we can see that the values are less than threshold thus the average recall between *CS+plain* and *LQF* variants are significantly different.

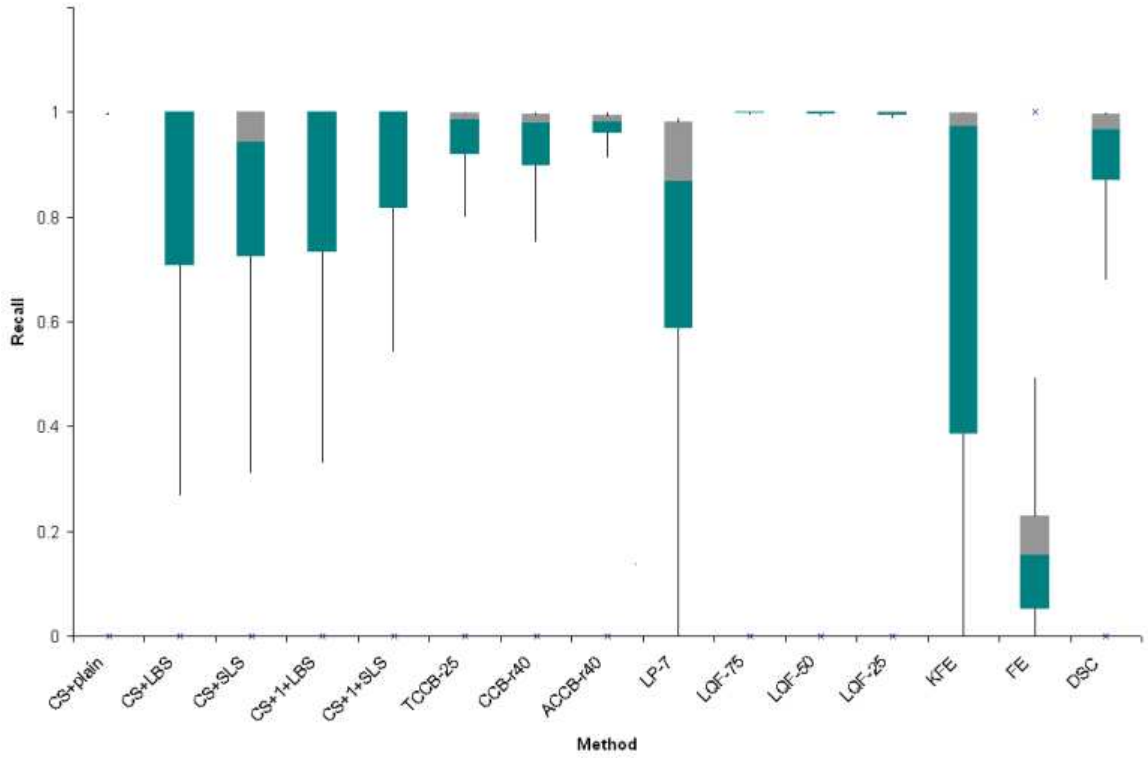

Figure 6.15: Variation and central tendency of recall for all extraction methods in the blog dataset

Table 6.24: Paired t-test result, threshold $\alpha = 0.05$

	LQF-75	*LQF-50*	*LQF-25*
CS+plain	0.045	0.017	9.087E-05

F1

For the average F1 score, *CS+1+LBS* has the best average F1 followed closely by *CS+LBS* as shown in Table 6.22. Compared to the other existing heuristic methods, *CS+1+LBS* performs better as depicted in Figure 6.16. However, the F1 difference of *CS+1+LBS* and *CS+LBS* is not that significant, the distribution of their F1 scores are also similar. The paired t-test result between *CS+1+LBS*, *CS+LBS*, and *CS+SLS* is shown in Table 6.25. From the test we can see that the *p-value* is larger than the threshold when we compare *CS+LBS* and *CS+1+LBS*, therefore we cannot reject *null* hypothesis that the average F1 between those two methods are equal. Thus, the F1 performance of *CS+LBS* and *CS+1+LBS* has no significant difference as the test shows lack of evidence to reject the null hypothesis.

Table 6.25: Paired t-test result, threshold $\alpha = 0.05$

	CS+LBS	*CS+SLS*
CS+1+LBS	0.0567	5.593E-08

Applying Other Classifier Models to The Blog Dataset

We also tried to apply the web content extraction to the blog dataset by using other classifier models namely news and forum classifier models. It is interesting to observe that the news classifier or forum classifier combined with LBS perform relatively good in terms of average precision as shown in Table 6.26. In fact, the average precision scores are very close to that of *CS+LBS* which uses blog classifier model. Moreover, for the average recall, the news classifier model yields 86.1% compared to 82.7% of *CS+LBS* which uses blog classifier model. This implies that it is *possible* to use the news classifier model to perform web content extraction on blog web pages.

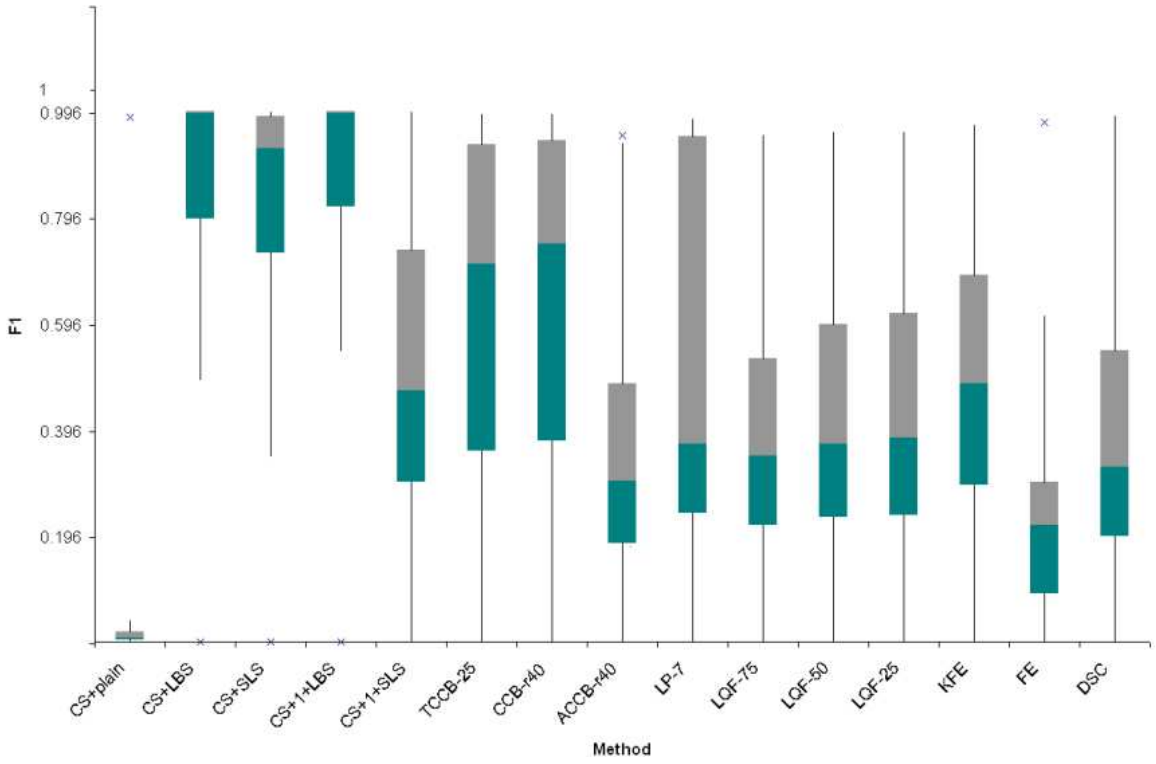

Figure 6.16: Variation and central tendency of F1 for all extraction methods in the blog dataset

Table 6.26: Results of applying other classifier models to the blog dataset

Learning Algorithm	Precision	Recall	F1
CS+LBS+ news classifier model	0.897±0.250	0.861±0.261	0.856±0.265
CS+1+LBS+ forum classifier model	0.892±0.232	0.785±0.336	0.790±0.324

6.6.6 Blog Dataset Evaluation Discussion

As we can see from the result of the blog dataset evaluation, our approach namely *CS+LBS* and *CS+1+LBS* yield the best performance with their high score of precision and F1. However, as we can see from Figure 6.14 and Figure 6.15 there are also outliers in which most of the approaches cannot extract the main content. The

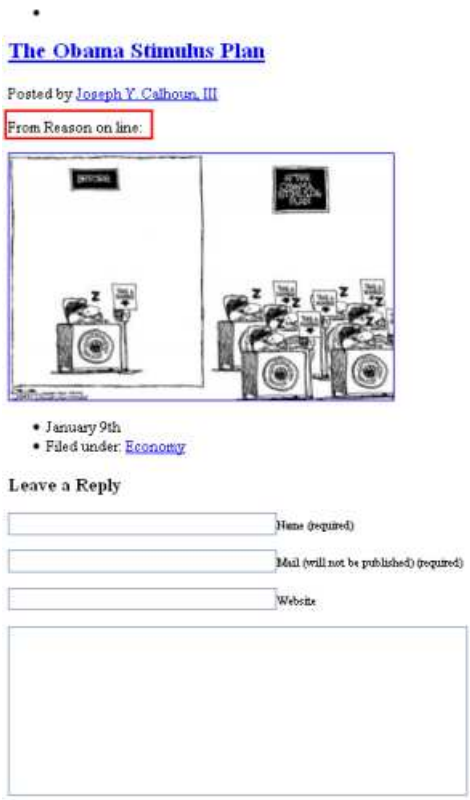

Figure 6.17: Outlier case in the blog dataset

example of this case is a blog posting which only consist of an image and very short text inside the main content, as illustrated in Figure 6.17.

As for *CS+plain*, it achieves very low precision since it only uses classification and extract all the text nodes, *CS+plain* cannot handle the noisy contents such as user comments. Based on our observation, user comments are classified as main content since they contain many stop words and fair amount of text.

As for *DSC*, although it achieves high score of recall, it only yields low precision score. The reason is that DSC also extracts the user comments. However, in our

gold standard, user comments are considered as noisy contents. Therefore, it get punished for this matter. Almost of the other methods also suffer from this problem of user comments. In addition to user comments problems, there is also element of blog web pages such as the profile of the owner of the blog. The profile of the owner of the blog usually contains long text. This problem cannot be handled by most of the methods. However, our approach with the *LBS* heuristic deals much better in this case. Therefore, our approach may yield high score of precision with the small standard deviation.

In general, in the context of recall, most of the methods perform relatively well except *FE*. Again, as in the news dataset, *FE* fail to extract all the DOM nodes which contain the main content as the DOM nodes typically contains many formatting tags. According to *FE* algorithm, it is difficult for *FE* to extract the DOM node which is not "pure" in text. However, for the case of blog posting that contains contiguous paragraph with less formatting, *FE* is able to extract the main content. Nevertheless, this case is rare to happen.

6.6.7 Forum Data Set Evaluation

For the forum dataset evaluation, we used different heuristic approach namely Table Pattern (TP).

Precision

In terms of precision as depicted in Table 6.27, our method namely *CS+TP* obtain the highest average precision. We can see from Figure 6.18, the spread of precision score of *CS+TP* is better than any other existing methods as the precision scores concentrate

Table 6.27: Forum dataset average precision, recall, F1, and standard deviation

	Precision	Recall	F1
Our Approach			
CS+TP	**0.870**±0.183	0.740±0.326	**0.754**±0.279
CS+1+TP	0.649±0.475	0.308 ±0.381	0.258 ±0.367
Existing Heuristic Approach			
TCCB-25	0.229±0.172	0.590±0.197	0.293±0.169
CCB-r40	0.241±0.169	0.528±0.199	0.295 ±0.159
ACCB-r40	0.244±0.190	0.847±0.173	0.342±0.213
LP-7	0.428±0.283	0.554±0.302	0.414±0.245
LQF-75	0.296±0.196	**0.974**±0.154	0.417±0.223
LQF-50	0.299±0.197	0.973±0.154	0.420±0.224
LQF-25	0.305±0.199	0.967±0.156	0.426±0.224
KFE	0.375±0.231	0.871±0.272	0.454±0.261
FE	0.567±0.302	0.348±0.215	0.366±0.201
DSC	0.166±0.152	0.509±0.212	0.213±0.150

around 93% score and the standard deviation is relatively small. From the Table 6.27 we may also see that the closest competitor to *CS+TP* in terms of average precision is the single phase version *CS+1+TP*. However, *CS+1+TP* has lower average precision and the standard deviation is rather large. From this observation, it is obvious that *CS+TP* is better than *CS+1+TP*.

The t-test between *CS+TP* and *CS+1+TP* also shows that the average precision of both approaches are significantly different as the *p-value* is lower than the threshold as shown in Table 6.28.

Table 6.28: Paired t-test result, threshold $\alpha = 0.05$

	CS+1+TP
CS+TP	2.19444E-08

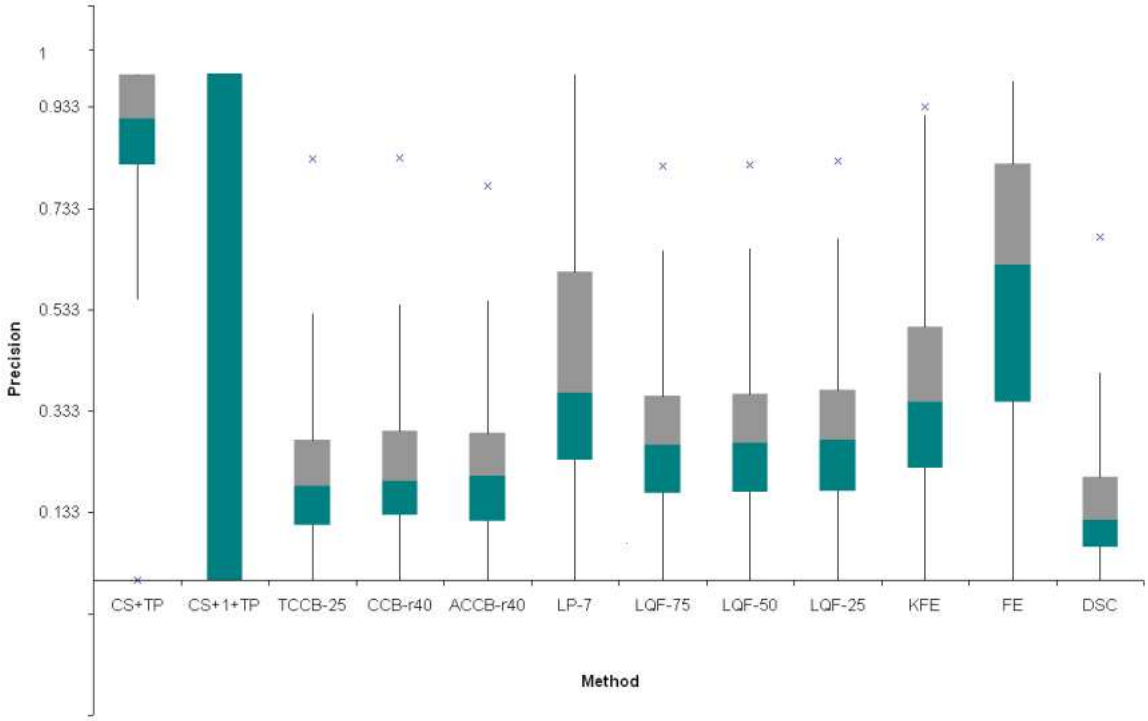

Figure 6.18: Variation and central tendency of precision for all extraction methods in the forum dataset

Recall

In terms of recall, the *LQF-75* yield the highest average recall as shown in Table 6.27. The other variants of LQF namely *LQF-50* and *LQF-25* also produce similar result. Moreover, as depicted in Figure 6.19, for *LQF-75* and *LQF-50* the spread of recall scores are very narrow as the region is only shown as a single point. However, although the recall of LQF variants are very high, the precision scores are low compared to the precision score of our method, *CS+TP* as illustrated in Figure 6.18.

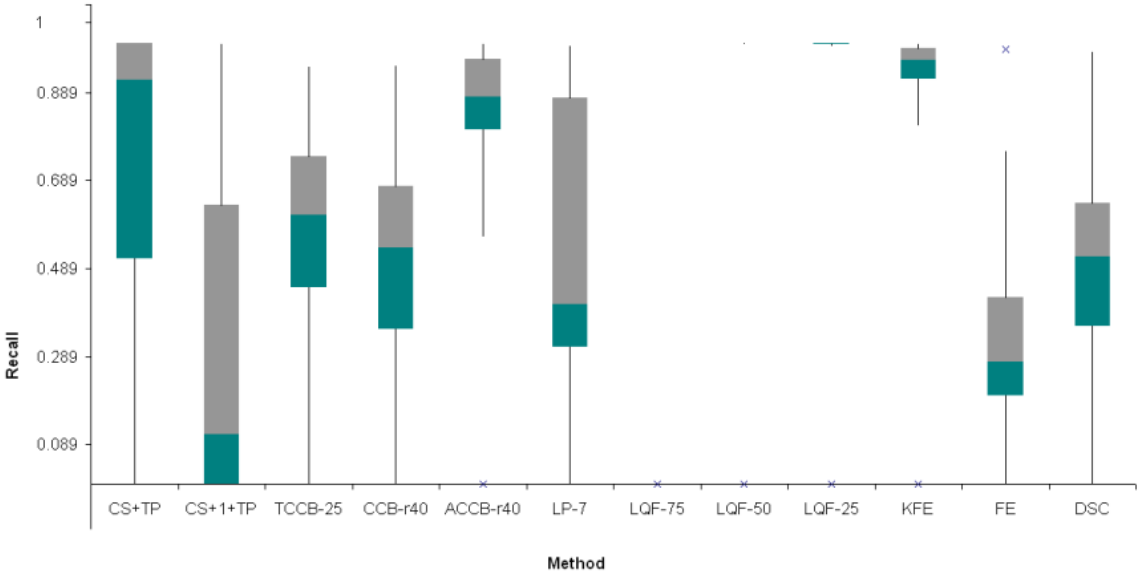

Figure 6.19: Variation and central tendency of recall for all extraction methods in the forum dataset

F1

As for F1 score, our method, *CS+TP* yields the best average score as shown in Table 6.27 . We also may see from Figure 6.20 that the distribution of the F1 scores of *CS+TP* is relatively higher than any other methods.

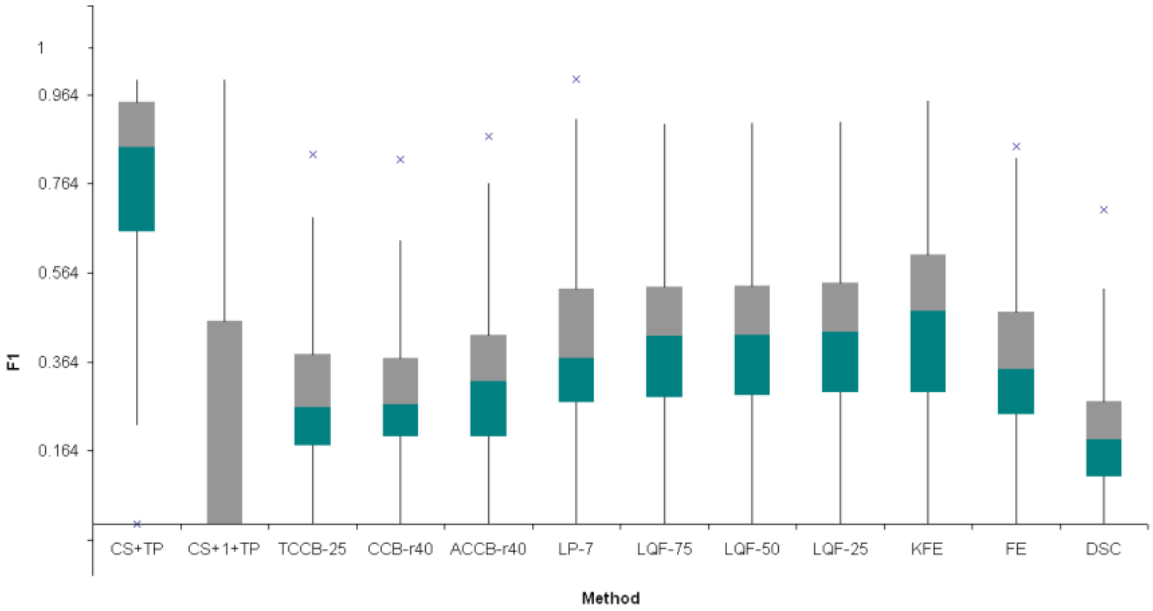

Figure 6.20: Variation and central tendency of F1 for all extraction methods in the forum dataset

Applying Other Classifier Models to Forum

Applying other classifier models namely blogs and news classifier models as shown in Table 6.29 yield relatively low precision but relatively high recall because the classifiers in many cases will return all part of the page including the user post identity, forum post description etc that contain many stop words and text.

Table 6.29: Results of applying other classifier models to the forum dataset

Method	Precision	Recall	F1
CS+news classifier model	0.489±0.218	0.980±0.106	0.624±0.207
CS+blog classifier model	0.646±0.168	0.974±0.125	0.762±0.146

6.6.8 Forum Dataset Evaluation Discussion

As we have seen in the evaluation, our approach namely *CS+TP* is the best method for forum dataset. However, there are cases which difficult for *CS+TP* and most of the methods to deal with. In many cases, in the forum postings, as illustrated in Figure 6.21 users only type several words in his posting or even a single emoticon. This is difficult because in the forum web pages also contains other elements such as the profile of the user, forum rules, description of the threads etc. These elements usually contain long text, therefore it might confuse the classifier to misclassify them as main content.

As for *CCB* variants such as *TCCB-25* and *CCB-r40*, they have problem since the variation of *content to code ratio* in forum web pages is more dynamic than news and blog web pages. For example, in a forum web page, there can be posts which vary in terms of the length of the text. In many cases, the posts which have short text and located rather far from the longer text will be missed by *CCB*.

There are also outlier cases in the dataset where a forum web page only contain single post with considerable amount of text. In this case, most of the existing heuristic methods may yield high precision and recall. However, this case is very rare to happen in the forum dataset as forum web page typically contain multiple postings.

Regarding the performance of the two-phase classifier and single-phase classifier in the forum dataset namely *CS+TP* and *CS+1+TP*, it is interesting to see that *CS+TP* significantly outperforms *CS+1+TP*. We investigated the cases in which *CS+1+TP* performs worse and found that *CS+1+TP* cannot extract the desired main content for about 30% of pages in the evaluation forum dataset. The problem is that the

Figure 6.21: Difficult case for forum dataset

DOM nodes returned by *CS+1+TP* are not suitable for the heuristic that we used. As for *CS+TP*, it returns the better DOM nodes for the heuristic that we applied. Figure 6.22 illustrated the case which single-phase classifier returns the undesirable DOM node for the heuristic. The heuristic that we applied namely TP expects that the returned DOM node is located inside the table structure of the forum post. As we can see from Figure 6.22, *CS+TP* returns the `td` element inside the table element

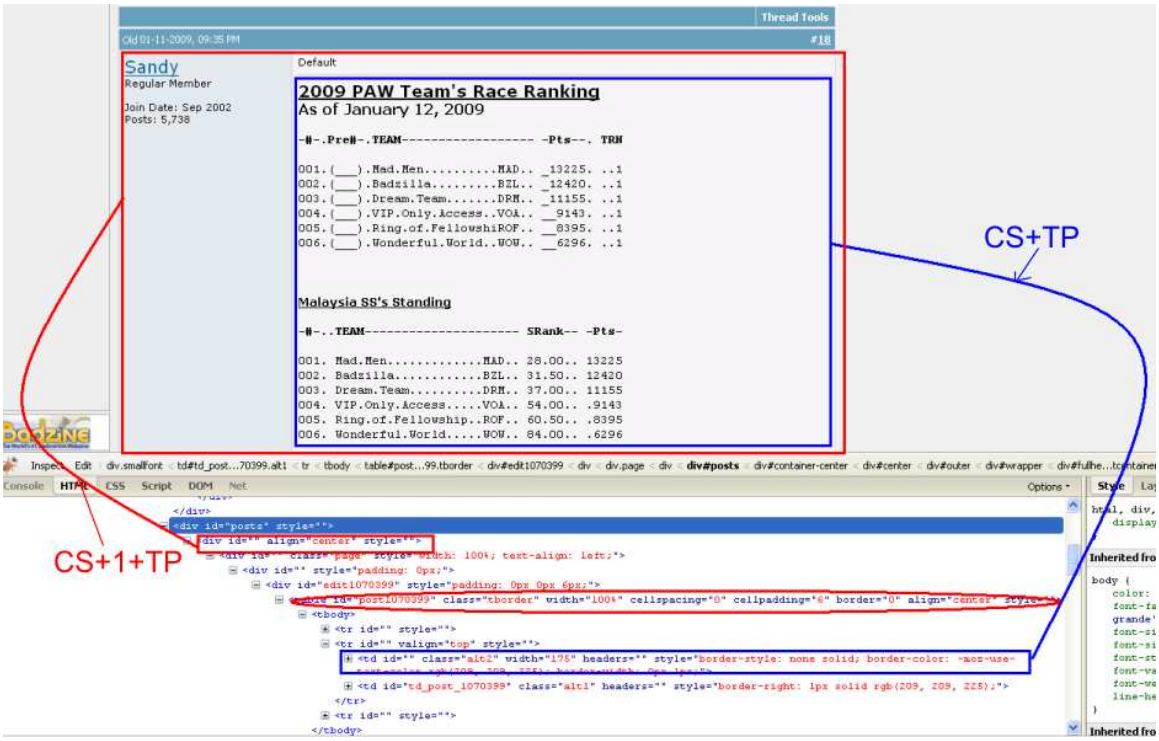

Figure 6.22: Typical case which single-phase classifier fails

while *CS+1+TP* returns the `div` element outside the table. As there is no `table` element in the upper level of the `div` element, thus the heuristic fails.

6.7 Additional Experiment

In Section 6.6, the training set is "independent" from the evaluation dataset, for example, if we use Yahoo News web pages as evaluation dataset then the Yahoo News will not be used in the training dataset. In this section, we want to try something different, performing the training by taking several pages from the evaluation dataset. We perform this experiment to observe how much training data we need to get from

a new source or type of web pages.We applied this approach to the forum dataset and performed the web content extraction by using *CS+1+TP*. We annotated five pages as training data and obtained 110 training instances.

We trained the Teezir Decision Tree and performed web content extraction evaluation. In particular, we conducted the training for various numbers of training instance which were taken randomly and observed the effect to the precision, recall, and F1 measure. The result is shown in Table 6.30, we can see that by only using 40 training instance can generate similar performance compared to that of 110 training instance. Moreover, compared to the existing heuristic approaches in Table 6.27, our approach which only uses 40 training instances obtain better result in terms of average precision, recall, and F1. This experiment shows that, in case there is a new type of web page source, we may achieve relatively good web content extraction by only using small amount of training instances from the target web page. We consider 40 training instances as a small amount, compared to the number of training instances that we generated in Section 6.3 which consist hundreds of training instances.

Table 6.30: Results of web content extraction for numerous numbers of training instances

Number of Training Instance	Precision	Recall	F1
20	0.024±0.155	0.024±0.155	0.024±0.155
40	0.975±0.155	0.682±0.303	0.762±0.278
80	0.975±0.155	0.678±0.296	0.762±0.268
110	0.975±0.155	0.678±0.296	0.762±0.268

6.8 Summary

According to our web content extraction evaluation results, for the **news dataset** and **blog dataset**, *CS+LBS* is the best approach. In terms of precision, *CS+LBS* shows that it always yield the best score in both datasets. In terms of recall, *CS+LBS* is outperformed by *LQF, Crunch,* and *CS+plain*. However, these methods yield very low precision score. Therefore, the *CS+LBS* is preferable since it is best in average precision and still gives reasonably high score of average recall thus it obtains the best F1 score. It is important to know that apparently the performance of the single-phase counterpart of *CS+LBS* namely *CS+1+LBS* is *not significantly different* with *CS+LBS* as shown by our statistical significance test. Therefore, for the news dataset and blog dataset, the classification approach either *single-phase* or *two-phase classifier* combined with LBS heuristic are the best approach. It is also interesting to see that by using news classifier model we can perform good extraction result to the blog dataset. In the other way around, by using blog classifier model we may also obtain relative good extraction result from news dataset.

For the **forum dataset**, according to our evaluation, the *CS+TP* is the best approach. The *CS+TP* yields the best performance in precision however it has tradeoff in terms of recall. The *LQF* variants are the best in recall, nevertheless its precision scores are very low. Therefore, we claim that *CS+TP* is the best approach for the forum dataset.

We also observed the effect of using only several pages from evaluation dataset as training data. In particular, we used forum dataset as an experiment and annotated five pages from the evaluation dataset. As a result, by using only small amount of training instances, we may achieve relatively good quality of web content extraction

even we can obtain higher average precision and F1 measure compared to the other existing heuristic approaches.

In terms of feature used for the classification, we found that by using several strong features may produce satisfying result. As the results from the feature selection for the segment training instance, the features such as *innerHtmlLength*, *domHeight*, and *divNum* dominate among all datasets. As for the content training instance, the dominant features are *stopWordRatio, stringMax, paraNum* and *innerTxtLength*.

Chapter 7

Conclusions and Future Work

In this thesis we have worked on several topics in the context of web content extraction. In this chapter we summarize in brief the contributions and formulate the conclusions that can be drawn from the results.

7.1 Results and Conclusions

In general, according to our experiments we have shown that our approach namely the combination of machine learning (classification) and our own developed heuristic approach is competitive compared to the existing heuristic methods.

Essentially, the main results and contributions of this thesis fall in two categories: classification approach for web content extraction and hybrid approach which consist of classification and several heuristic for content extraction.

7.1.1 Classification Approach for Web Content Extraction

The web content extraction by applying plain classification namely segment classification and content classification turned out to yield relatively good result in terms of

recall. However, we realized that by only performing classification is not sufficient, in particular if we want to get high score of precision. The reason is, often inside a DOM node which is classified as main content there can be many embedded noisy contents. This case is common for modern web documents as advertisements, link to related articles, comments etc often placed together with the main content in the same DOM node. Therefore, we think the classification approach is relatively good to *identify which DOM node contains the main content* however for *extracting the exact text string* of the main content we have to add something else namely heuristic rules.

In terms of features used for the classification task, we found that by only using several strong features, the classifier may identify the pattern to distinguish between the good segment and the bad segment, the main content and the noisy content. Features such as *stopWordRatio, stringMax, paraNum* and *innerTxtLength* dominate the content classification meanwhile the features such as *innerHtmlLength*, *domHeight* and *divNum* are strong for segment classification.

Regarding the generalization of the classifier, according to our experiment, the classifier model used in the news dataset can also be applied in the blog dataset and yield reasonably high precision, recall, and F1. The same thing happened when we applied classifier model that we trained from blog dataset and applied the content extraction to the news dataset.

However, when we used the classifier model from blog or news, it cannot generalize to the forum dataset. This is understandable since the structure of forums are completely different with blogs or news websites. Likewise, when we used the forum classifier model and then applied the model on news or blog dataset, it did not give satisfying result.

In addition, we also observed that by training the classifier from small amount of pages from the target web pages, our approach is able to yield high precision and F1 compared to the other existing heuristic methods.

7.1.2 Hybrid Approach for Content Extraction

Our next contribution after the classification approach is the development of heuristic for content extraction. As explained before, by applying only classification is often not sufficient. We introduced several heuristics that can be used in the news, blogs and forums datasets namely *Largest Block of String (LBS)*, *String Length Smoothing (SLS)* and *Table Pattern (TP)*. Essentially, our hybrid approach manages to perform better than the existing related works especially in terms of precision.

As an overall result, the *hybrid approach* as a combination of *LBS* and classification tasks namely *CS+LBS* and *CS+1+LBS*, yield the best performance for blogs and news datasets. *LBS* based on the assumption that the main content usually is placed in a large contiguous text node which is suitable for news and blogs web pages. From the result we can see that there is no significant difference in overall performance between *two-phase classifier*, *CS+LBS* and *single-phase classifier*, *CS+1+LBS*.

As for forum dataset, we developed the other heuristic namely *TP*. *TP* is based on the observation that forum web pages always presented in a table-like structure. According to our experiment, the combination of classification and *TP*, *CS+TP* gave reasonably high score of precision and F1 compared to the other existing heuristic methods. In the forum dataset, the single-phase classifier, *CS+1+TP* is outperformed by the two-phase classifier, *CS+TP* .

7.2 Future Work

In our current work, we have developed three kinds of classifier for the web content extraction module namely for news, blogs, and forums. The web content extraction module can be used in crawling process by passing the parameter namely the web site type to be crawled and the appropriate classifier will be invoked. It is possible to eliminate this parameter and determine the web page type by building separate classifier focused on classifying the web page genre prior to content extraction.

It is also interesting to complement the web content extraction module with a "tracking" module that can monitor the structure changes of the web page and the extraction result. With this track module we can observe the performance of the web content extraction module over a span of time. For instance, whether the content extractor still return the correct text string. In this way can decide whether we should re-train the classifier. Another way that we can do to update the knowledge of the classifier is by adding a kind of relevance feedback mechanism from which human can judge the relevance of the classification result.

Another challenge that may arise is the use of HTML 5. According to HTML 5 specification [7], there are new elements, for example `article` and `nav` tags. These tags reflect the typical usage on modern website, `article` defines the section where the main contents such as blog posts, forum posts, news articles should be placed. While `nav` defines a section which contains links to other pages. As for the segment classification we defined several structural elements based on HTML tags, we need to incorporate the new tags to adapt with HTML 5 environment. As for identifying content, we think that the feature such as *stopWordRatio* and *innerTxt* length are still dominant although HTML tags change over time.

Bibliography

[1] R. Hartono A.F.R. Rahman, H. Alam, *Content extraction from html documents*, In 1st Int. Workshop on Web Document Analysis (2001).

[2] Ryan Coleman-W. Bruce Croft Matthew King Wei Li David Pinto, Michael Branstein and Xing Wei, *Quasm: a system for question answering using semi-structured data*, In JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, 2002, pp. 46–55.

[3] T.G. Dietterich, *Approximate statistical tests for comparing supervised classiication learning algorithms*, Neural Computation, vol. 10, 1998, pp. 1895–1924.

[4] George C. Runger Douglas C.Montgomery, *Applied statistics and probability for engineers*, Wiley, 2007.

[5] Thomas Gottron, *Content extraction: Identifying the main content in html documents*, Ph.D. thesis, Johannes Gutenberg-Universitt Mainz, 2008.

[6] D.S. Hirschberg, *A linear space algorithm for computing maximal common subsequences*, Communications of ACM Volume 18 Number 6, 1975, p. 342.

[7] http://dev.w3.org/html5/spec/Overview.html.

[8] http://rankings.big boards.com.

[9] http://www.w3.org/TR/2000/WD DOM-Level-1-20000929/introduction.html.

[10] http://www.w3.org/TR/html401/struct/global.html.

[11] Eibe Frank Ian H. Witten, *Data mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2005.

[12] N. Japkowicz and S Stephen, *The class imbalance problem: A systematic study*, Intelligent Data Analysis, vol. 6, 2002, pp. 429–450.

[13] Marie-Francine Moens Javier Arias Moreno, Koen Deschact, *Language independent content extraction from web pages.*, Proceedings of the 9th Dutch- Belgian Information Retrieval Workshop, 2009, pp. 50–55.

[14] Mazin Gilbert Junlan Feng, Patrick Haffner, *A learning approach to discovering web page semantic structure*, Proceedings of the 2005 8th International Conference on Document Analysis and Recognition (2005).

[15] Shian-Hua Lin and Jan-Ming Ho, *Discovering informative content blocks from web documents*, ACM SIGKDD, 2002.

[16] Geoffrey Holmes Mark A. Hall, *Benchmarking attribute selection techniques for discrete class data mining*, IEEE Transcations on Knowledge and Data Engineering **15** (2003).

[17] Vipin Kumar Pang-Ning Tan, Michael Steinbach, *Introduction to data mining*, Addison Wesley, 2005.

[18] C. Lee Gilles Sandip Debnath, Prasenjit Mitra, *Automatic extraction of informative blocks from web pages*, Proceedings of the 2005 ACM Symposium on Applied Computing, 2005, pp. 1722–1726.

[19] H. Wen J.-R W.-Y Ma Song, R. Liu, *Learning block importance models for web pages*, ACM WWW, 2004.

[20] David Neistadt Suhit Gupta, Gail Kaiser and Peter Grimm, *Dom-based content extraction of html documents*, Proceedings of the 12th International Conference on World Wide Web, 2003, pp. 207–214.

[21] Gail Kaiser Suhit Gupta and Salvatore Stolfo, *Extracting context to improve accuracy for html content extraction*, Special Interest Tracks and Posters of the 14th International conference on World Wide Web, 2005, pp. 1114–1115.

[22] Peter Grimm Michael F. Chiang Suhit Gupta, Gail E. Kaiser and Justin Starren, *Automating content extraction of html documents*, World Wide Web, 2005, pp. 179–224.

[23] Tim Weninger and William H. Hsu, *Text extraction from the web via text-tag-ratio*, Proceedings of the 5th International Workshop on Text Information Retrieval, 2008, pp. 23–28.

[24] Sridhar Rajagopalan Ziv Bar-Yossef, *Template detection via data mining and its application*, ACM WWW Conference, 2002.

Appendix A

Decision Tree Parameter Tuning

A.1 Two-Phase Classifier

A.1.1 News Classifier

Segment Classification

Table A.1: News Dataset Feature Selection- Segment Classification

Number of Feature Used	Precision	Recall	F1
1	0.96	0.93	0.944
2	0.959	0.936	0.947
3	0.961	0.946	0.953
4	0.955	0.946	0.950
5	0.965	0.956	0.960
6	0.96	0.964	0.961
7	0.96	0.962	0.960
8	0.96	0.964	0.961
9	0.96	0.964	0.961
10	0.962	0.964	0.962

Table A.2: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.965	0.956	0.960
6	0.961	0.956	0.958
10	0.949	0.937	0.942
14	0.95	0.944	0.946
18	0.956	0.934	0.944
20	0.96	0.93	0.944

Content Classification

Table A.3: News Dataset Feature Selection- Content Classification

Number of Feature Used	Precision	Recall	F1
1	0.705	0.907	0.793
2	0.83	0.724	0.773
3	0.893	0.866	0.879
4	0.91	0.896	0.902
5	0.883	0.917	0.899
6	0.932	0.922	0.926
7	0.962	0.95	0.955
8	0.971	0.963	0.966
9	0.956	0.963	0.959
10	0.956	0.963	0.959

Table A.4: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.937	0.925	0.930
6	0.939	0.916	0.927
10	0.946	0.915	0.930
14	0.93	0.91	0.919
18	0.93	0.91	0.919
20	0.91	0.93	0.919

A.1.2 Blog Classifier

Segment Classification

Table A.5: Blogs Dataset Feature Selection- Segment Classification

Number of Feature Used	Precision	Recall	F1
1	0.902	1	0.943
2	0.932	0.972	0.951
3	0.951	0.957	0.953
4	0.957	0.956	0.956
5	0.958	0.951	0.954
6	0.944	0.961	0.952
7	0.942	0.971	0.956
8	0.953	0.974	0.963
9	0.949	0.974	0.961
10	0.946	0.966	0.955

Table A.6: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.958	0.951	0.954
6	0.954	0.955	0.954
10	0.957	0.95	0.953
14	0.953	0.955	0.953
18	0.956	0.95	0.952
20	0.956	0.944	0.949

Content Classification

Table A.7: News Dataset Feature Selection- Content Classification

Number of Feature Used	Precision	Recall	F1
1	0.817	0.869	0.842
2	0.862	0.931	0.773
3	0.897	0.933	0.914
4	0.893	0.931	0.911
5	0.899	0.933	0.915
6	0.901	0.923	0.911
7	0.899	0.915	0.906
8	0.896	0.909	0.902
9	0.898	0.913	0.905
10	0.889	0.919	0.903

Table A.8: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.899	0.933	0.915
6	0.897	0.921	0.908
10	0.902	0.915	0.908
14	0.904	0.913	0.908
18	0.9	0.909	0.904
20	0.898	0.913	0.905

A.1.3 Forum Classifier

Segment Classification

Table A.9: Forum Dataset Feature Selection - Segment Classification

Number of Feature Used	Precision	Recall	F1
1	0.975	0.931	0.952
2	0.973	0.98	0.976
3	0.968	0.986	0.976
4	0.98	0.976	0.977
5	0.978	0.971	0.974
6	0.978	0.976	0.976
7	0.978	0.978	0.978
8	0.98	0.978	0.978
9	0.98	0.98	0.98
10	0.976	0.981	0.978

Table A.10: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.978	0.976	0.976
6	0.978	0.973	0.975
10	0.971	0.978	0.974
14	0.954	0.973	0.963
18	0.957	0.968	0.962
20	0.96	0.966	0.962

Content Classification

Table A.11: Forum Dataset Feature Selection - Content Classification

Number of Feature Used	Precision	Recall	F1
1	0.776	0.656	0.710
2	0.822	0.767	0.793
3	0.88	0.897	0.888
4	0.925	0.933	0.928
5	0.926	0.941	0.933
6	0.923	0.945	0.933
7	0.923	0.953	0.937
8	0.940	0.937	0.938
9	0.925	0.929	0.926
10	0.932	0.925	0.928

Table A.12: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.925	0.933	0.928
6	0.932	0.921	0.926
10	0.927	0.901	0.913
14	0.934	0.901	0.917
18	0.922	0.885	0.903
20	0.912	0.862	0.886

A.2 Single Phase Classifier

A.2.1 News Classifier

Table A.13: News Dataset Feature Selection

Number of Feature Used	Precision	Recall	F1
1	0.812	0.903	0.855
2	0.95	0.919	0.934
3	0.952	0.96	0.955
4	0.946	0.984	0.964
5	0.953	0.984	0.968
6	0.953	0.984	0.968
7	0.953	0.984	0.968
8	0.953	0.984	0.968
9	0.953	0.984	0.968
10	0.953	0.984	0.968

Table A.14: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.953	0.984	0.968
6	0.946	0.984	0.964
10	0.946	0.984	0.964
14	0.946	0.984	0.964
18	0.944	0.952	0.947
20	0.951	0.935	0.942

A.2.2 Blogs Classifier

Table A.15: Blogs Dataset Feature Selection

Number of Feature Used	Precision	Recall	F1
1	0.811	0.897	0.851
2	0.853	0.939	0.893
3	0.877	0.918	0.897
4	0.888	0.897	0.892
5	0.903	0.918	0.910
6	0.907	0.92	0.913
7	0.889	0.918	0.903
8	0.914	0.918	0.915
9	0.913	0.915	0.913
10	0.913	0.913	0.913

Table A.16: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.903	0.918	0.910
6	0.907	0.918	0.912
10	0.91	0.92	0.914
14	0.899	0.923	0.910
18	0.893	0.923	0.907
20	0.892	0.927	0.909

A.2.3 Forum Classifier

Table A.17: Forums Dataset Feature Selection

Number of Feature Used	Precision	Recall	F1
1	0.819	0.575	0.675
2	0.763	0.754	0.758
3	0.845	0.885	0.864
4	0.887	0.933	0.909
5	0.896	0.921	0.908
6	0.913	0.952	0.932
7	0.916	0.952	0.933
8	0.922	0.944	0.932
9	0.922	0.944	0.932
10	0.93	0.944	0.936

Table A.18: Precision,Recall, F1 for Various Values of Min Obj at confidence factor = 0.25

Min Obj	Precision	Recall	F1
2	0.896	0.921	0.908
6	0.878	0.913	0.895
10	0.878	0.913	0.895
14	0.861	0.913	0.886
18	0.852	0.889	0.870
20	0.852	0.869	0.860

Appendix B

Feature Selection

B.1 Comparison of Feature Selection Methods

Table B.1: Comparison of feature selection methods in two-phase classifier dataset with J48

Dataset	J48	IG	RLF	PC	CNS	CFN	WRP
Segment News	94.76±1.98	94.72±1.96	94.63±1.99	93.15±2.26 •	94.55±2.07	93.59±2.36 •	69.16±0.37 •
Segment Blog	93.91±1.93	93.96±1.90	93.83±1.94	93.02±2.10	93.36±2.40	93.25±2.35	89.06±0.28 •
Segment Forum	97.51±1.88	97.48±1.89	97.46±1.88	96.25±1.99	97.26±1.96	97.74±1.93	87.32±0.68 •
Content Blog	92.70±2.44	92.81±2.41	92.82±2.40	89.89±3.29 •	93.09±2.39	92.80±2.35	50.35±0.32 •
Content Forum	97.18±1.91	97.19±1.89	97.21±1.89	91.30±3.33 •	96.99±2.39	96.69±2.22	62.68±0.64 •
Content News	99.24±0.71	99.23±0.72	99.23±0.72	99.04±0.89	99.09±1.01	99.00±0.97	87.49±0.42 •
Average	95.88	95.90	95.86	93.78	95.72	95.51	74.34

◦, • statistically significant improvement or degradation

Table B.2: Comparison of feature selection methods in single-phase classifier dataset with J48

Dataset	J48	IG	RLF	PC	CNS	CFN	WRP
News	99.24±0.71	99.23±0.72	99.23±0.72	99.04±0.89	99.09±1.01	99.00±0.97	87.49±0.42 •
Blog	92.70±2.44	92.81±2.41	92.82±2.40	89.89±3.29 •	93.09±2.39	92.80±2.35	50.35±0.32 •
Forum	97.18±1.91	97.19±1.89	97.21±1.89	91.30±3.33 •	96.99±2.39	96.69±2.22	62.68±0.64 •
Average	96.37	96.41	96.42	93.41	96.39	96.17	66.84

◦, • statistically significant improvement or degradation

B.2 Two-Phase Classifier

Table B.3: Segment Classification Result with Feature Selection

Learning Algorithm	News			Blogs			Forums		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Decision Tree (J48)	0.965	0.954	0.96	0.962	0.949	0.955	0.978	0.971	0.975
Random Forest	0.965	0.972	0.968	0.956	0.979	0.967	0.986	0.983	0.985
SMO	0.692	1	0.818	0.901	0.992	0.944	0.876	0.995	0.932
Multilayer Perceptron	0.919	0.92	0.92	0.907	0.998	0.95	0.934	0.986	0.96

Table B.4: Content Classification Result with Feature Selection

Learning Algorithm	News			Blogs			Forums		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Decision Tree (J48)	0.984	0.945	0.964	0.899	0.933	0.915	0.926	0.941	0.933
Random Forest	0.976	0.961	0.968	0.919	0.95	0.935	0.944	0.941	0.943
SMO	0.832	0.937	0.881	0.752	0.925	0.829	0.617	0.344	0.442
Multilayer Perceptron	0.984	0.961	0.972	0.883	0.887	0.885	0.783	0.798	0.791

Table B.5: Feature Selection Result for Two-Phase Classifier

Data Set	Features for Segment Classification	Features for Content Classification
News	innerHtmlLength domHeight innerTxtLength linkNum divNum tagName linkToTextRatio imgNum stopWordRatio stringMax	stopWordRatio stringMax innerTxtLength paraNum linkToTextRatio domHeight innerHtmlLength divNum tagName imgNum
Blogs	divNum innerHtmlLength domHeight linkNum innerTxtLength tagName paraNum imgNum headerAround stopWordRatio	linkToTextRatio stopWordRatio paraNum innerTxtLength divNum innerHtmlLength tagName imgNum linkNum domHeight
Forums	innerTxtLength linkToTextRatio domHeight stringMax innerHtmlLength divNum stopWordRatio linkNum imgNum tableNum	innerTxtLength stopWordRatio divNum paraNum stringMax tableNum linkNum linkTextRatio imgNum domHeight

B.3 Single-Phase Classifier

Table B.6: Classification Result with Feature Selection with Single-Phase Classifier

Learning Algorithm	News			Blogs			Forums		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Decision Tree (J48)	0.961	0.984	0.972	0.903	0.918	0.910	0.896	0.921	0.908
Random Forest	0.976	0.976	0.976	0.907	0.944	0.925	0.893	0.96	0.925
SMO	0.961	0.984	0.972	0.716	0.984	0.829	0.679	0.571	0.621
Multilayer Perceptron	0.91	0.984	0.946	0.883	0.887	0.885	0.704	0.813	0.755

Table B.7: Feature Selection Result for Single-Phase Classifier

Data Set	Features Selected
News	stopWordRatio innerTextLength stringMax paraNum linkTextRatio domHeight innerHtmlLength divNum imgNum tagName
Blogs	stopWordRatio linkTextRatio paraNum innerTextLength divNum innerHtmlLength tagName domHeight linkNum imgNum
Forums	innerTextLength stopWordRatio divNum paraNum stringMax tableNum linkNum linkTextRatio domHeight imgNum

Appendix C

Teezir Decision Tree Classification Result

Table C.1: Segment Classification Result with Two-Phase Classifier

Segment Classification-News								
Learning Algorithm	**tp**	**fn**	**fp**	**tn**	**accuracy**	**precision**	**recall**	**f1**
Teezir Decision Tree	672	30	31	282	0.940	0.956	0.957	0.957
Segment Classification-Blog								
Teezir Decision Tree	866	38	30	81	0.933	0.967	0.958	0.962
Segment Classification-Forum								
Teezir Decision Tree	552	7	11	65	0.972	0.980	0.987	0.984

Table C.2: Content Classification Result with Two-Phase Classifier

Content Classification-News							
Learning Algorithm	**tp**	**fn**	**fp**	**tn**	**accuracy**	**precision**	**recall**
Teezir J48	121	6	3	885	0.991	0.976	0.953
Content Classification-Blog							
Teezir J48	468	36	38	473	0.927	0.925	0.929
Content Classification-Forum							
Teezir J48	228	11	21	375	0.950	0.916	0.954

Table C.3: Classification Result by Using Single-Phase Classifier

News							
Learning Algorithm	**tp**	**fn**	**fp**	**tn**	**accuracy**	**precision**	**recall**
Teezir J48	890	7	1	117	0.992	0.999	0.992
Blog							
Teezir J48	433	37	38	389	0.916	0.919	0.921
Forum							
Teezir J48	408	18	9	243	0.960	0.978	0.958