Improving Retrieval Accuracy
in
Main Content Extraction
from
HTML Web Documents

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades
DOCTOR rerum naturalium
(Dr. rer. nat.)
im Fachgebiet
Informatik

vorgelegt
von Master Applied Math. Hadi Mohammadzadeh
geboren am 12.10.1969 in Mashhad, Iran

Die Annahme der Dissertation wurde empfohlen von:
1. Professor Dr. Jinan Fiaidhi (Ontario, Kanada)
2. Professor Dr. Gerhard Heyer (Leipzig)

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 27.11.2013 mit dem Gesamtprädikat *magna cum laude*

.

Amtierender Dekan:
Professor Dr. Hans-Bert Rademacher

Gutachter:
Professor Dr. Jinan Fiaidhi
Professor Dr. Gerhard Heyer

Tag der Promotion: 27. November 2013

I would like to dedicate this thesis to my loving parents

Abstract

The rapid growth of text based information on the World Wide Web and various applications making use of this data motivates the need for efficient and effective methods to identify and separate the "main content" from the additional content items, such as navigation menus, advertisements, design elements or legal disclaimers.

Firstly, in this thesis, we study, develop, and evaluate R2L, DANA, DANAg, and AdDANAg, a family of novel algorithms for extracting the main content of web documents. The main concept behind R2L, which also provided the initial idea and motivation for the other three algorithms, is to use well particularities of Right-to-Left languages for obtaining the main content of web pages. As the English character set and the Right-to-Left character set are encoded in different intervals of the Unicode character set, we can efficiently distinguish the Right-to-Left characters from the English ones in an HTML file. This enables the R2L approach to recognize areas of the HTML file with a high density of Right-to-Left characters and a low density of characters from the English character set. Having recognized these areas, R2L can successfully separate only the Right-to-Left characters. The first extension of the R2L, DANA, improves effectiveness of the baseline algorithm by employing an HTML parser in a post processing phase of R2L for extracting the main content from areas with a high density of Right-to-Left characters. DANAg is the second extension of the R2L and generalizes the idea of R2L to render it language independent. AdDANAg, the third extension of R2L, integrates a new preprocessing step to normalize the hyperlink tags. The presented approaches are analyzed under the aspects of efficiency and effectiveness. We compare them to several established main content extraction algorithms and show that we extend the state-of-the-art in terms of both, efficiency and effectiveness.

Secondly, automatically extracting the headline of web articles has many applications. We develop and evaluate a content-based and language-independent approach, *TitleFinder*, for unsupervised extraction of the headline of web articles. The proposed method achieves high performance in terms of effectiveness and efficiency and outperforms approaches operating on structural and visual features.

Zusammenfassung

Das rasante Wachstum von textbasierten Informationen im World Wide Web und die Vielfalt der Anwendungen, die diese Daten nutzen, macht es notwendig, effiziente und effektive Methoden zu entwickeln, die den Hauptinhalt identifizieren und von den zusätzlichen Inhaltsobjekten wie z.B. Navigations-Menüs, Anzeigen, Design-Elementen oder Haftungsausschlüssen trennen.

Zunächst untersuchen, entwickeln und evaluieren wir in dieser Arbeit R2L, DANA, DANAg und AdDANAg, eine Familie von neuartigen Algorithmen zum Extrahieren des Inhalts von Web-Dokumenten. Das grundlegende Konzept hinter R2L, das auch zur Entwicklung der drei weiteren Algorithmen führte, nutzt die Besonderheiten der Rechts-nach-links-Sprachen aus, um den Hauptinhalt von Webseiten zu extrahieren.

Da der lateinische Zeichensatz und die Rechts-nach-links-Zeichensätze durch verschiedene Abschnitte des Unicode-Zeichensatzes kodiert werden, lassen sich die Rechts-nach-links-Zeichen leicht von den lateinischen Zeichen in einer HTML-Datei unterscheiden. Das erlaubt dem R2L-Ansatz, Bereiche mit einer hohen Dichte von Rechts-nach-links-Zeichen und wenigen lateinischen Zeichen aus einer HTML-Datei zu erkennen. Aus diesen Bereichen kann dann R2L die Rechts-nach-links-Zeichen extrahieren. Die erste Erweiterung, DANA, verbessert die Wirksamkeit des Baseline-Algorithmus durch die Verwendung eines HTML-Parsers in der Nachbearbeitungsphase des R2L-Algorithmus, um den Inhalt aus Bereichen mit einer hohen Dichte von Rechts-nach-links-Zeichen zu extrahieren. DANAg erweitert den Ansatz des R2L-Algorithmus, so dass eine Sprachunabhängigkeit erreicht wird. Die dritte Erweiterung, AdDANAg, integriert eine neue Vorverarbeitungsschritte, um u.a. die Weblinks zu normalisieren. Die vorgestellten Ansätze werden in Bezug auf Effizienz und Effektivität analysiert. Im Vergleich mit mehreren etablierten Hauptinhalt-Extraktions-Algorithmen zeigen wir, dass sie in diesen Punkten überlegen sind.

Darüber hinaus findet die Extraktion der Überschriften aus Web-Artikeln vielfältige Anwendungen. Hierzu entwickeln wir mit *TitleFinder* einen sich nur auf den Textinhalt beziehenden und sprachabhängigen Ansatz. Das vorgestellte Verfahren ist in Bezug auf Effektivität und Effizienz besser als bekannte Ansätze, die auf strukturellen und visuellen Eigenschaften der HTML-Datei beruhen.

Acknowledgements

Like all research, the work described in this dissertation was not conducted in isolation. It is impossible to thank everyone who has in some way contributed to it and equally impossible to thank individuals for all of their contributions but I would like to thank those people that gave me the opportunity to work and enjoy during my PhD journey. I am indebted for technical insights, encouragement and friendship to many members of the Institute of Applied Information Processing at the University of Ulm, as well as those I had the privilege of working with in the Department of Natural Language Processing at the University of Leibzig.

This thesis would not have been possible without the support and the insights of Prof. Dr. Franz Schweiggert. I'm pleased for the given chance to reach this point of my career. I am most grateful to Prof. Dr. Gholamreza Nakhaeizadeh. It has been a big pleasure working with him. I have to thank him for all the support, the suggestions, the patience, and every kind of help that he gave to me, and for the person I became at this point. And of course, for the time spent out of work and his continues suggestions. Grateful acknowledgment is made to my supervisor Prof. Dr. Gerhard Heyer for all his help, guidance, and support.

I sincerely would like to thank Prof. Dr Jinan Fiaidhi for being the reviewer of this thesis. She gave me useful comments.

Thanks also go to my friends and colleagues at the University of Ulm, in particular Dr. Andreas Borchert, Armin Rutzen, Wolfgang Kaifler, Rene Just, Steffen Kram and Michaela Weiss. I would like to show my gratitude to Dr. Thomas Gottron for his advice, collaboration, and the opportunity to work and publish some papers together. I want to thank one of my best friends Dr. Norbert Heidenbluth for helping me to prepare figures and diagrams.

A special thank is dedicated to Mrs Roswitha Jelinek and Mrs Gisela Richter for them support and efforts. At the end, I appreciate Dr. Ljubow Rutzen-Loesevitz for proof-reading this thesis.

Publications

The algorithms, for main content extraction form HTML web documnets, and experimental results introduced in this thesis have been published in six conference and workshop proceedings in addition to one book chapter, as follows:

- Hadi Mohammadzadeh, Franz Schweiggert, and Gholamreza Nakhaeizadeh. *Using UTF-8 to Extract Main Content of Right to Left Language Web Pages.* In: ICSOFT 2011 - Proceedings of the 6th International Conference on Software and Data Technologies, Seville, Spain, 2011, volume 1, pp. 243 – 249, SciTePress.

- Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gholamreza Nakhaeizadeh. *A Fast and Accurate Approach for Main Content Extraction based on Character Encoding.* In: TIR'11: Proceedings of the 8th International Workshop on Text-based Information Retrieval(DEXA'11), Toulouse, France, 2011, pp. 167 – 171, IEEE Computer Society.

- Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gholamreza Nakhaeizadeh. *Extracting the Main Content of Web Documents Based on a Naive Smoothing Method.* In: KDIR'11: International Conference on Knowledge Discovery and Information Retrieval, Paris, France, 2011, pp. 470 – 475, SciTePress.

- Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gholamreza Nakhaeizadeh. *Extracting the Main Content of Web Documents based on Character Encoding and a Naive Smoothing Method.* In: Software and Data Technologies, CCIS Series, Springer, 2013, pp. 217 – 236, Springer-Verlag Berlin Heidelberg.

- Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gholamreza Nakhaeizadeh. *The impact of source code normalization on main content extraction.* In: WEBIST'12: 8th International Conference on Web Information Systems and Technologies, Porto, Portugal, 2011, pp. 677 – 682, SciTePress.

- Hadi Mohammadzadeh, Omid Paknia, Franz Schweiggert, and Thomas Gottron. *Revealing trends based on defined queries in biological publications using cosine similarity.* In: BIOKDD'12: 23th International Workshop on Database and Expert Systems Applications (DEXA'12), Vienna, Austria, 2012, pp. 218 – 222, IEEE Computer Society.

- Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and Gerhard Heyer. *TitleFinder: Extracting the Headline of News Web Pages based on Cosine Similarity and Overlap Scoring Similarity.* In: WIDM'12: Proceedings of the 12th International ACM Workshop on Web Information and Data Management, Hawaii, USA, 2012, pp. 65 – 71, ACM Press.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Problem Description

This chapter will briefly discuss the history of the World Wide Web (WWW) in Section 1.1. It will also explain how much useful information has been published on the web by users so far and what proportion of the information on the web is in text-based format. A glance at this deal of text information will demonstrate that the processing of such a huge volume of articles is completely useful and necessary to acquire knowledge. It is also interesting to note that most of the text-based information written in HTML format is stored in web sites (and of course in web pages) and some encyclopedia sites such as Wikipedia. Section 1.2 will visually demonstrate the various elements of a news web site (BBC in this case) and a web page, respectively. Details about the "Statement of Problem" are provided in Section 1.3, where the definition for content extraction is suggested and different parts of a typical HTML file is explained. Furthermore, it will be concluded that implementation of content extraction algorithms is not very simple due to the reasons which will be explained completely in this section. Finally in Section 1.4 we will briefly demonstrate the Thesis Outline of this contribution.

1.1 Motivation

The first idea about WWW was initiated in 1980 when Tim Berners-Lee designed a network called ENQUIRE (116) at CERN, Switzerland. Although the modern web today is rather different, the main idea has been inspired from his invention. Tim and his co-worker Robert Caillian established the first successful connection between a host and an http user on December 25, 1990 through the internet. Specifications extracted from HTTP, URL and HTML were then published as web technology (25). Shortly after this, the first web site was designed at CERN with the first online site being established on August 6, 1991. This web site used to describe WWW, how anyone can have a web explorer and how to customize a web service provider. Although a long time has not elapsed from the appearance of the web, a great deal of information has been published on web with a considerable amount of it being in text-based format (11)(see Figure 1.1). To the best of our knowledge, it is not simple to determine the entire volume of this information, but the following statistics - even though

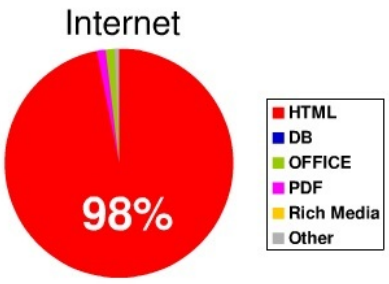


Figure 1.1: Distributions of file formats in Internet (11)

they are not exact figures - clearly reveal its daily accelerated growth (11).

- 3.164 billion - Number of email accounts worldwide
- 112 - Number of emails sent and received per day by the average corporate user
- 2.267 billion - Internet users worldwide (December 2011)
- 555 million - Number of web sites (December 2011)
- 300 million - Added web sites in 2011
- 800+ million - Number of users on Facebook by the end of 2011
- 225 million - Number of Twitters account
- 250 million - Number of tweets per day (October 2011)
- 70 million - Total number of Word press blogs by the end of 2011

To emphasize and illustrate the rapid growth of information on the web, Figure 1.2 shows the increase of hostnames in the last years. As it can be seen, the curve approximates an exponential form and then indicates that the growth tendency is going to increase even further (117).

It is worth mentioning that the design and creation of web pages for natural languages other than English, German, Spanish, French, and Portuguese, for example the Persian and Arabic languages, have been accelerated by the spread and pervasion of the internet into all underdeveloped countries. This may encourage one to focus on the natural language used in web pages when extracting useful information, e.g. main content and headline, from the web pages. The BBC news web site, for example, provides users and visitors with information in various languages.

In summary, it seems necessary to process this volume of information, in which a significant amount is in text format, as well as in different languages. In order to process text information saved in news web pages automatically, one would first need to extract this useful information from HTML web documents. The most important and significant components of information which exist on various web pages are believed to be the following items:

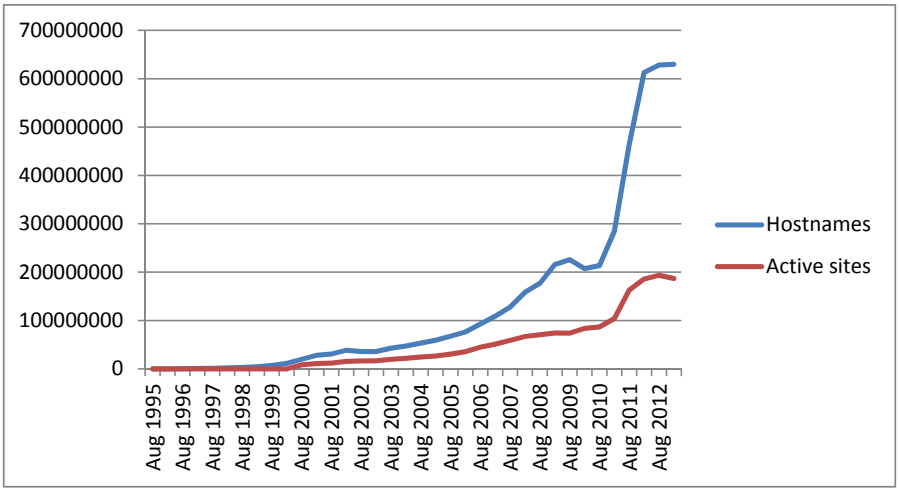


Figure 1.2: Growth of the number of hostname (117)

- Main content
- Headline, subtitles
- Author(s)
- Date

Of course some other components can be placed in one HTML file which are called *additional items* or *external items* , such as menus, advertisements, footer and header, logos, counters, search boxes, category information, navigational links, related links, copyright information, which constitute up to 40-50% of the total contents of web pages (42).

1.2 Elements of Web Sites and Web Pages

Thousands of news broadcasts appear on web sites and web pages daily. Although each web site and web page specialize on a special subject, i.e. *politics, sports, science, culture*, etc., one common factor among all these web pages is that they all have a *main content, a headline, a specified author and a date of issuance* (10). The remaining discussion will be assigned to introduce elements in most web sites and web pages. Figure 1.3 depicts a screenshot of the BBC news web site. To illustrate elements which contain important information at the bottom of the web site in Figure 1.3, some middle parts of this web site, including various news items, have been omitted. As can be clearly seen, this web site involves the following elements:

- Logo + Menu + Search
- Top News Story
- News
- Business
- Sport
- More From BBC News
- Future
- Spotlight

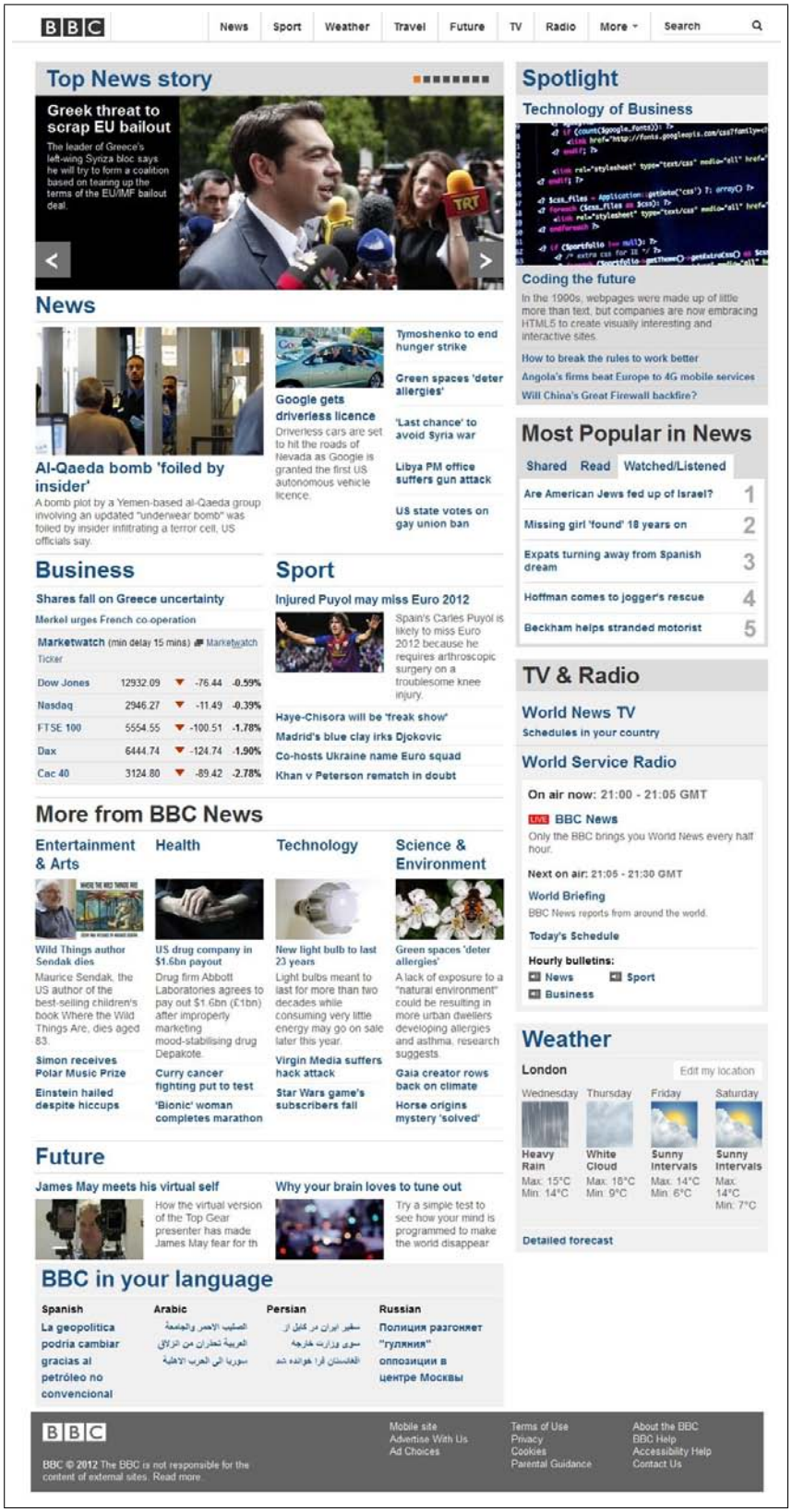


Figure 1.3: Sample of a BBC Web Site

- Most Popular in News
- BBC in your language
- TV and Radio
- Weather
- Footer

All elements in this web site, with the exception of logo, menu and footer, will lead us to web pages which contain one news component each. For example, if one chooses the news entitled *"New light bulb to last 23 years"* from technology tab, one will be directed to the web page depicted in Figure 1.4. The following elements are located in the web page shown in Figure 1.4:

- The first rectangle in red color includes the logo of the BBC web site, main menu and search choice. Located in the same section is a banner labeled *"News Technology"* which defines the category of the news on this web page. It can be argued that this web page belongs to the technology group. Beneath this banner is another menu which directs one to other sections of the site.

- The next part seen on this web page is the image of an advertisement (Lights, CAMERA, LONDON), which is somehow associated with the main subject of this web page. To the best of our knowledge, several ongoing research is being conducted currently about contextual advertisement to show an advertisement to the user which is related to the subject of text on a web page. This can possibly persuade the user to have a look at it (44) (21).

- The next part of this web page provides the user with an option to print this page, send it to a friend through email, or share it with others on social networks such as Facebook and Twitter.

- The rest of web page is divided into two columns. The left column shows the news body in addition to its upload time (8 May 2012 in this case) and the headline of the news (here, LED light bulb to last more than 20 years). The main body of news is written beneath which contains some subsections and paragraphs. For example, two subsections of "Saving Energy" and "LED Challenges" are seen with some paragraphs. Some figures are also demonstrated on the left column in addition to their captions. Furthermore, there is another part entitled *"Related Stories"*, which is not regarded as a part of news. It is not accounted for the main content of this web page but includes the news related to the subject of this web page for further reference of the interested users. *"More on this story"* is shown again after the main text body which is not part of the main content but gives similar information to the users. Features to print, send via email, and share this web page is devised in the following.

- Immediately after this part is an advertisement whose subject is not similar to that of the web page. This is followed by the section services, including news feeds, mobile, podcasts, alerts and email news. Finally, the web page footer can be seen on the bottom of the page containing some information such the copyright, "Terms of Use", "Contact Us", "Advertise With Us", and "BBC Help".

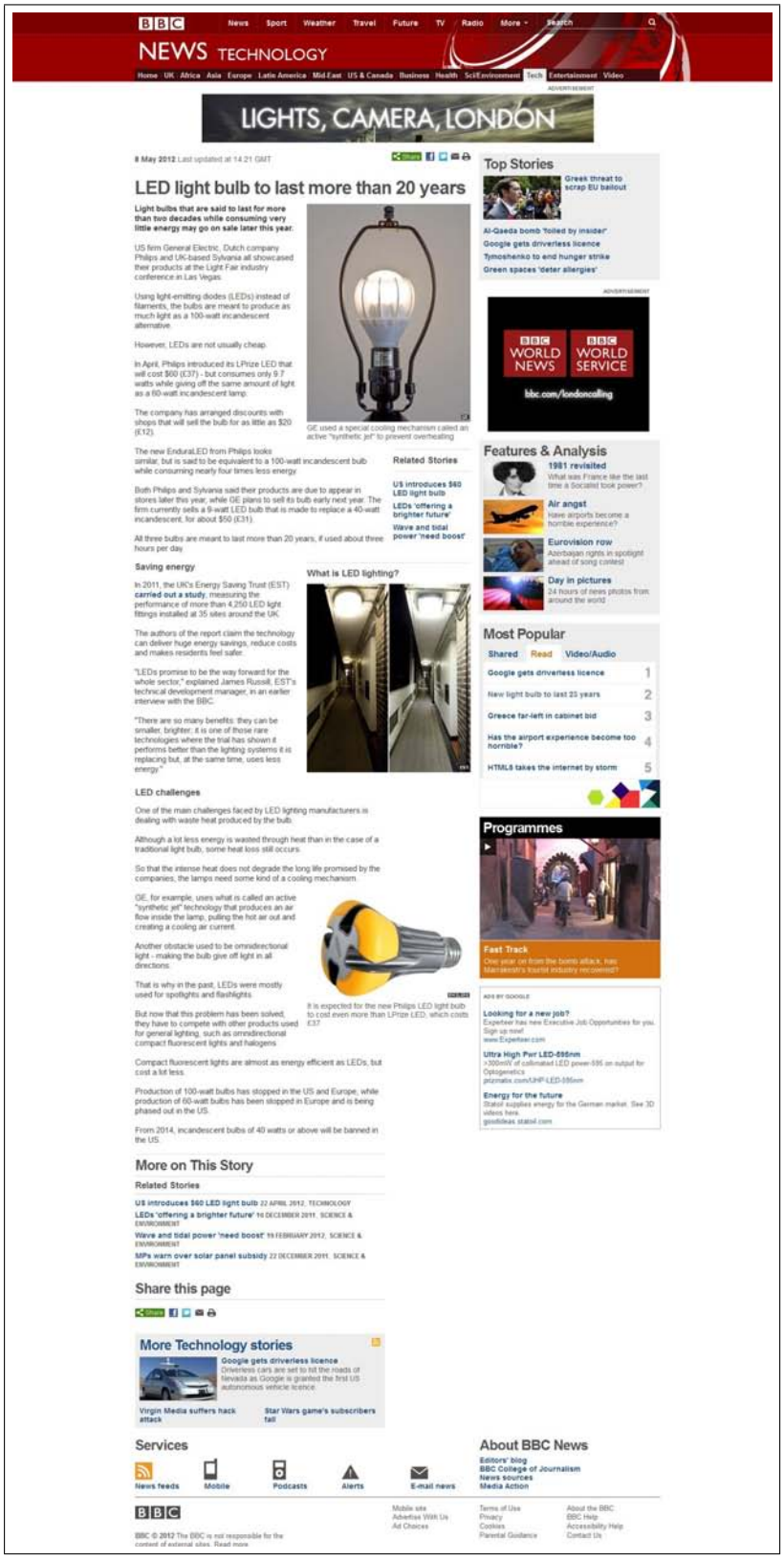


Figure 1.4: Sample of a BBC Web Page

1.3 Statement of Problem

The rapid growth of text-based information on the web and various applications making use of this data motivates the need for efficient and effective methods to identify and separate the *Main Content (MC)* from the additional content items. In particular, the identification of the MC is beneficial for web search engines: when crawling and indexing the web, knowing the actual main content of each web page can be utilized for the purpose of determining more precise and descriptive index terms for the document. Furthermore, *Main Content Extraction (MCE)* is also applied in scenarios in which a reduction of a document to its main content is of advantage, e.g. on devices that have limited storage or bandwidth capacity or underlie restrictions regarding the presentation of web documents (55), such as mobile phones, screen readers, etc. Furthermore, MCE can be considered as a preprocessing step for general text mining applications operating on the web. All these application scenarios have led to the development of several approaches and algorithms for main content extraction from HTML documents. In addition, taking into account the great diversity of web sites on WWW, the most important examples of which are:

- News web pages: e.g. CNN, BBC
- Blog sites: e.g. blogger, Xanga
- Social web sites: e.g. Facebook, Twitter, google+
- Information sites: e.g. Encyclopedia and Wikipedia

Considering the varying structure and layout of HTML files for each web site, it seems necessary to propose outstanding algorithms which are able to extract the main content from various web sites automatically. Nevertheless, this is not simple and requires accurate algorithms because any web page has an anarchic nature and some components which are called "additional items" such as *menu*, *advertisement*, *most viewed news*, etc. cause extraction of some irrelevant information once we use main content extraction algorithms. Another feature is that some web pages do not have any text as a main content and they contain only images or videos instead. In this case, algorithms of extracting main content often mistake another text located in the HTML file of that web page and report it to the user as the main content! Providing images and videos from these kinds of web pages is definitely irrelevant to this thesis and in the last chapter we will give helpful comments on how to handle these types of web pages in order to extract these multimedia items. Consequently, the objectives of this thesis are:

"Investigating the state of the art algorithms in fields of main content extraction from HTML web documents. Moreover, the main part of this thesis deals with developing novel and accurate algorithms for extracting the main content and headline of HTML news web pages".

It can be claimed that the term *"Content Extraction" (CE)* was first introduced by Rahman (100) in 2001 in a conversation entitled *"Content extraction for HTML documents"*. Subsequently, after almost a decade, numerous studies have been launched on main content extraction; several algorithms have been implemented and various datasets have been formed to test these algorithms. In spite of all these studies, it is still impossible to extract the main content by a special algorithm which is capable of being applied on all kinds of web pages with 100% accuracy. Some algorithms are able to extract the main content of some web pages with maximum 98% accuracy but they will show rather low accuracy once employed for other web pages. Hence, they can not be considered to be general algorithms for main content extraction.

1.3.1 Definition of Main Content Extraction

Thomas Gottron, in his article labeled *"Content Code Blurring"* (48)(2008), defined content extraction as follows:

Definition 1. *Content extraction is the process of identifying the Main Content and/or removing the additional items, such as advertisements, navigation bars, design elements or legal disclaimers*

It is interesting to note in his definition that once the additional items such as advertisement, navigation bars and design elements are removed, the main content in a web page would be accessible (110). The only difference is that considering the different structures of web pages, identification and deletion of these additional items would not seem simple. In an HTML web document, the main content is usually but not always placed next to initial information of the web page, namely JavaScript codes, CSS codes and HTML codes related to construction of menus. Meanwhile, footers are located after the main content. Listing 1.1 completes the explanations of this paragraph.

1.3.2 Strategy of Proposed Approaches

The most common traditional approach to MCE has been to hand-code rules, often implemented by regular expressions. These hand tailored rules achieve a high or even perfect accuracy on the web documents for which they have been designed. However, since different web sites have different layouts, perhaps even in a variety of configurations and layouts frequently changing over time, hand coded rules are highly labour intensive and easily broken by changes to the structure of a web page (95). This has led to an interest in finding solutions which are generic (i.e. applicable to various types of web pages from different web sites), accurate (i.e. able to extract all important content at a high precision) and efficient (i.e. capable of processing a large number of web pages at a high throughput rate) (74). However, generic MCE approaches do not reach a perfect accuracy. Even state-of-the-art methods have been shown to still leave room for improvement (46).

Listing 1.1: Sample of HTML file in which the MC is placed in the middle of HTML file

```
<!DOCTYPE html>
<html>
  <head>
    --------------  CSS Code ---------------
    <style type="text/css">
                body { background-color:#d0e4fe; }
                h1  { color:orange; }
                p   { font-size:10px; }
    </style>
  </head>
  <body>
    --------------  Menu Bar ---------------
    <div id="blq-local-nav">
      <ul id="nav" class="nav">
       <li class="first-child␣"><a href="/news/">Home</a></li>
       <li><a href="/news/uk/">UK</a></li>
       <li><a href="/news/world/africa/">Africa</a></li>
       <li><a href="/news/world/asia/">Asia</a></li>
       <li><a href="/news/world/europe/">Europe</a></li>
       <li><a href="/news/world/latin_america/">Latin America</a></li>
       <li><a href="/news/world/middle_east/">Mid-East</a></li>
       <li><a href="/news/world/us_and_canada/">US &amp;amp; Canada</a></li>
       <li><a href="/news/business/">Business</a></li>
       <li><a href="/news/health/">Health</a></li>
       <li><a href="/news/science_and_environment/">Sci/Environment</a></li>
       <li class="selected"><a href="/news/technology/">Tech</a></li>
       <li><a href="/news/entertainment_and_arts/">Entertainment</a></li>
       <li><a href="/news/10462520">Video</a></li>
      </ul>
    </div>

    ---------------   Main Title  -------
    <h1 class="story-header">LED light bulb to last more than 20 years</h1>

    ---------------   First Photo in MC with its Caption --
    <div class="caption␣body-narrow-width">
       <img src="http://news.bbcimg.co.uk/media/images/60090000/jpg/_60090417.jpg"
        width="304" height="405" alt="General␣Electric␣LED␣light␣bulb" />
       <span style="width:304px;">GE used a special cooling mechanism called an
         active &amp;quot;synthetic jet&amp;quot; to prevent overheating</span>
     </div>

    ---------------   Main Content Area -----
     <p class="introduction" id="story_continues_1">Light bulbs that are said to
      last for more than two decades while consuming very little energy may go
      on sale later this year.</p>

     <p>US firm General Electric, Dutch company Philips and Sylvania all showcased
      their products at the Light Fair industry conference in Las Vegas.</p>
     <p>Using light-emitting diodes (LEDs) instead of filaments, the bulbs are
      meant to produce as much light as a 100-watt incandescent alternative.</p>
     <p>However, LEDs are not usually cheap.</p>

     <p>In April, Philips introduced its LPrize LED that will cost $60 ( 37) - but
      consumes only 9.7 watts while giving off the same amount of light as a
      60-watt incandescent lamp.</p>
     <p>The company has arranged discounts with shops that will sell the
      bulb priced at around $20 ( 12).</p>
     <p>The new EnduraLED from Philips looks similar, but is said to be equivalent
      to a 100-watt incandescent bulb while consuming only a quarter of the energy.
     </p>
  </body>
</html>
```

Chapter 4 of this thesis will utilize the idea we are going to explain here briefly for implementation of proposed algorithms. In each HTML web document, characters (or tokens) can be considered as *Content* or *Code*. The characters (or tokens) which are part of HTML tags, CSS codes, and JavaScript codes are categorized as *Code* while the rest of characters (or tokens) will be grouped as *Content*.

The second categorization that can be performed on HTML web documents is that characters (or tokens) which have been grouped into *Content* will be separated into two subcategories: *Main Content*, shaping the main part of news web page, and *Non-main Content*, for example content characters (or token) used in menu bar, header and footer of an HTML web document. As can be seen in Listing 1.1, the number of main content characters (or token) in the main content area is interestingly greater than the number of code characters (or token) used in HTML tags (90) while in other areas the density of code characters (or tokens) is much higher than the density of non-main content characters (or tokens), for instance HTML tags composing menu bar. In some regions, for instance CSS and JavaScript codes, the number of content characters is exactly zero. Now the problem of finding main content in the HTML web page becomes the problem of finding regions which have a high density of *Main Content* characters (or tokens) and low density of *Code* characters (or tokens), comprising the main content.

1.4 Thesis Outline

Taking into account the above-mentioned issues which specify necessity of extracting useful information including main content and headline from web pages, the following chapters will explain proper algorithms which can get this information.

- Some explanations will be provided about features of texts and web page documents in Chapter 2 of this thesis. Then, the concepts in information retrieval will be briefly discussed. The basic concepts of content extraction will be emphasized later, and finally Unicode and UTF-8 encoding form will be reviewed in concisely at the end of this chapter.

- Chapter 3 will comprehensively mention state-of-the-art algorithms about extraction of main content from web pages. This chapter will attempt to divide the algorithms of main content extraction into several categories and then explain the algorithms of main content extraction which are located under each of these categories. Content extraction systems will be studied at the end of this chapter.

- Chapter 4 which is the most important part of this thesis has adopted to explain and describe algorithms of main content extraction from web pages. Algorithms which will be investigated in this chapter are: R2L (89), DANA (85), DANAg (84, 87), and finally AdDANAg (86), in order of appearance. We analyse our approaches under the aspects of efficiency and effectiveness. We compare them to eleven established MCE algorithms (41, 48, 55, 80, 91, 96) and show that we extend the state-of-the-art in terms of both efficiency and effectiveness. Meanwhile, this thesis will introduce a new set of data sets which have been collected and prepared by the author of this thesis.

- Chapter 5 aims to explain an algorithm which is able to extract the headline of web pages (83). At the beginning of this chapter, previous contributions in this field will be introduced and subsequently, the algorithm implemented in current work, TitleFinder, will be explained in detail. This chapter also argues that the accuracy of the main content extraction from web pages can be improved when it is possible to extract the headline from them.
- Chapter 6 will point out one of the most interesting applications in the fields of text mining and main content extraction (88). Having extracted relevant information of some published papers about biology, this chapter has tried to demonstrate whether the number of published papers on a given topic by a specific publication has decreased or increased during a definite period of time (approximately 40 years).
- Finally, Chapter 7 will discuss the conclusion and future works. However, some ideas addressed in the future works section of this chapter are being investigated and examined at the present time.

Chapter 2

Background

This chapter will discuss some basic issues. First, features of text and web page document in addition to components in an HTML document will be discussed in Section 2.1. Then, Section 2.2 aims to explain the concepts of Information Retrieval (IR) briefly and describe two well known models of IR in detail. Section 2.3, which is of great importance, will emphasize on the concepts of content extraction and explain the concepts of gold standard, recall, precision and F1-measure. Finally, Section 2.4 will point to Unicode and UTF-8 encoding form, since Chapter 4 needs such concepts.

2.1 Understanding Text and Web Page Documents

Before being able to run any process on texts and web pages, the components of a text file and web page document must be identified. This section explains different components of a text and a web page in brief. It will then illustrate elements and structure of an HTML file. Afterwards, DOM tree as one hierarchical view of a web document will be introduced and finally, the pre-processing operations needed in this thesis to be applied on web documents will be presented.

2.1.1 Text file as a sequence of characters, tokens, lines, paragraphs

Text files, including HTML documents, are made up of a series of characters. Meanwhile, tokens are created by placing some characters together. In a text file which contains English characters, all tokens are a subset of words in that language. However, the situation is somewhat different in an HTML file. As mentioned in Section 1.3.2, all tokens can be divided into two general categories in HTML files, namely content and code tokens. The purpose of content tokens are those which can be accounted for a member from the set of tokens of a natural language, i.e. English. On the other hand, code tokens are those which form HTML tags. In Listing 2.1, the content tokens highlighted in bold can be distinguished from the code tokens. This code segment is a part of HTML code provided in Chapter 1 (Listing 1.1).

Listing 2.1: A Portion of HTML file comprising of menu items

```
-------------- Menu Bar ---------------
1     <div id="blq-local-nav">
2     <ul id="nav" class="nav">
3     <li class="first-child "><a href="/news/">Home</a></li>
4     <li><a href="/news/uk/">UK</a></li>
5     <li><a href="/news/world/africa/">Africa</a></li>
6     <li><a href="/news/world/asia/">Asia</a></li>
7     <li><a href="/news/world/europe/">Europe</a></li>
8     </ul>
9     </div>
```

Listing 2.2: A Portion of HTML file comprising of some paragraphs.

```
   --------------- Main Content Area -----
1     <p class="introduction" id="story_continues_1">Light bulbs that are said to
2     last for more than two decades while consuming very little energy may go
3     on sale later this year.</p>
4     <p>US firm General Electric, Dutch company Philips and Sylvania all showcased
5     their products at the Light Fair industry conference in Las Vegas.</p>
6     <p>Using light-emitting diodes (LEDs) instead of filaments, the bulbs are
7     meant to produce as much light as a 100-watt incandescent alternative.</p>
```

The words **Home, UK, Africa, Asia and Europe** are content tokens, while other tokens in this code segment such as `</ul>` or `<div>` are code tokens. Lines of an HTML file are produced by putting together content tokens and code tokens. However, sometimes only the code tokens can be seen in a line of HTML file. The lines 1, 2, 8 and 9 in Listing 2.1 only contain code tokens, whereas other lines include both content tokens and code tokens.

The paragraphs in an HTML file can be mentioned after the lines. A paragraph is a set of lines which is usually placed in an HTML tag. The most common and prevalent HTML tag to build a paragraph is the `<p>` tag. Three paragraphs are seen in Listing 2.2 which is a part of Listing 1.1. Here, the content tokens are shown in bold to be easily distinguished from code tokens.

To conclude, the content tokens can be divided into two general categories as below:

- *Main Content Tokens*: These are content tokens which make the MC of a web page. All content tokens in Listing 2.2 are classified under these kinds of tokens.
- *Non-main Content Tokens*: These are tokens which are not included in the MC of a web page. For example, all content tokens summarized in Listing 2.1 constitute items of the web page menu and are known as non-main content tokens.

2.1.2 Elements and Structure of an HTML document

The elements in web sites and web pages were investigated in terms of their appearance in Chapter 1 (Section 1.2), meaning what is displayed by web browsers. This section adopts to

study web pages in terms of their structural point of view (60). In other words, it wants to see which components typically exist in an HTML file.

In an HTML file, in addition to the content tokens which can be observed by an end user through a web browser, the following elements and components can be used:

- HTML tags
- CSS codes
- JavaScripts codes
- Comments
- Meta tags[1]

It should be noted that the content tokens are usually placed in an HTML tag. In Listing 2.1 and Listing 2.2, the content tokens have been located between opening tags and closing tags. Concerning CSS, JavaScript, comments and Meta tags, the only point which must be mentioned here is that since these components have nothing to do with extraction of MC and headline from web pages, they are often removed from the HTML file prior to running the extraction algorithm of MC and headline. Noteworthy here is that HTML files sometimes have syntactic errors, the most common of which is forgetting to put a closing tag at the end of an HTML tag. Thus, whenever the extraction program of MC is sensitive to syntactic errors, and especially to the closing tags, it is recommended to use some programs in order to fix these errors. One of the best well-known programs for this purpose is Tidy (99), which is able to correct syntactic errors. Tidy is also able to receive an HTML file and transform it into a valid HTML code or even a XHTML code. Programmers who use the Java language for implementation of their projects can utilize a Java version of this project called JTidy (45).

2.1.3 Dom Tree, an Hierarchical view of HTML documents

The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. In other words, DOM is a programming interface for XML and HTML documents. Using this interface, we can have a full access to the XML and HTML documents and process these types of documents. Programmers can create a document by DOM, add some elements to it, remove some elements from it, and particularly modify it. DOM has been designed by World Wide web Consortium (W3C) for being used via programming languages (2, 7, 8, 9). DOM makes it possible to have a tree presentation from an XML or HTML document, because the elements inside XML and HTML documents have nested structures. Texts which are located between opening tags and closing tags are usually demonstrated as leaves of this tree. In order to display the DOM of an HTML document by a tree, one would need some applications to receive an HTML document as input and then demonstrate the relevant DOM tree.

[1] Meta tag is data (information) about data

The document element is usually placed at the highest level of such a tree with one or more children. However, manipulation of the elements in a DOM tree needs to utilize a programming language such as JavaScript. Numerous MC extracting algorithms are introduced in Chapter 3 of this thesis which have utilized DOM tree to achieve their goals.

2.1.4 Text and Web Page Pre-processing

It was stated before in Section 2.1.1 that tokens are the smallest components of a document in a text file and also in an HTML file. However, it will be discussed further in Section 2.2.2.2 that all of these tokens, which are extracted from an HTML file, do not show the same importance and value. Generally speaking, some tokens, including prepositions, do not have considerable value for further processes, while others such as keywords, are much more valuable once identified. Therefore, some pre-processes are implemented on the text before main processing in text mining and information retrieval. The following has addressed some of these pre-processing operations.

2.1.4.1 Tokenization

Assuming a given stream of text, tokenization can be defined as the breakdown of this stream into some parts called tokens (14) and probably removing some special characters such as punctuation simultaneously. White space, line break or punctuation characters are used in order to tokenize a text. Furthermore, punctuation characters are also deleted. Nevertheless, the tokenization is not always simply possible due to the following reasons:

- Existence of special characters such as an apostrophe which is used to show possession and contraction;
- Existence of hyphenated words in some languages including English;
- Existence of compound nouns which are used in some languages including German;
- Existence of URLs and emoticons in many of computerized texts;
- Existence of some languages where there is no space between different words in a text, such as Chinese.

Despite all of the aforementioned problems, there are some algorithms which can solve them.

2.1.4.2 Stopword Removal

Stopwords are words we specify to exclude from our text because they occur too frequently or because of their unimportance to the text (15). In other words, stopwords are words which are filtered out prior to, or after, processing of natural language data (text) (16). Most of the time, stopwords are repeated several times in texts and do not have any meaning alone. Some examples of these words are prepositions and adjectives. One of the methods to determine a stopword list is to extract tokens of a text first and sort the tokens based on their number of repetitions in the text. Tokens with the most frequency can be considered as a stopword

list. However, some stopword lists are made for each language including English, German, Farsi (114), etc. Thus, by comparing the tokens extracted from a text with a stopword list, it is easy to specify which extracted tokens are parts of the stopword list. Sometimes removing the stopwords will cause some problems. For example, when the stopwords are removed from a phrase, that phrase might lose its main meaning and sense.

In Chapter 5, an algorithm will be explained which aims to extract headline from web pages.

2.1.4.3 Stemming

The term stemming in information retrieval and linguistic morphology is used to address the process of reducing inflected words to their stem, base or root form. The selected stem is not necessarily the same as morphological root of the word and the related words should just be mapped to the selected stem, even though this selected stem is not the main root itself. Stemmer is usually a software which tries to perform the stemming process on a selected text. Generally speaking, the right stemmers must be utilized for different languages due to their dissimilar linguistic structures. Two sample stemmers for German and Farsi have been proposed by Caumanns (26) and Taghva (115) in 1998 and 2003, respectively. An example can be provided here to clarify this issue. By application of the special stemmer of English language on "cats", "catlike" and "catty" strings, they will be mapped into the "cat" token, because "cat" is the root for all of these strings. Porter-stemmer (98) is one of the most famous stemming algorithms for the English language, which was invented by Martin Porter in 1980.

2.1.4.4 Web Page Pre-processing

A pre-processing step is considered in all algorithms provided in this thesis, i.e. R2L, DANA, DANAg, AdDANAg, TitleFinder and finally TrendFinder, in order to prepare the input data for the main processing. Although this will be discussed further in Chapters 4, 5 and 6, all steps of pre-processing can be summarized as below:

- Pre-processing in R2L, DANA, DANAg and AdDANAg algorithms:
  - normalizing the lines of HTML files
  - removing the JavaScript codes from HTML codes
  - removing the CSS codes from HTML files
  - removing the comment lines from HTML files
- Pre-processing in TitleFinder algorithm:
  - removing all the stopwords before running the main program
- Pre-processing in TrendFinder algorithm:
  - using stemmer algorithm
  - removing all the stopwords before running the algorithm

2.2 Information Retrieval

First, basic concepts of information retrieval will be briefly discussed in this section and then information retrieval models, i.e. Boolean and VSM, will be emphasized and examined. At the end, some explanations will be provided about evaluation measures.

2.2.1 Basic Concept of IR

The term IR has found extensive applications in recent years through academic societies and even people. It can be claimed that the invention of WWW has caused the popularity and prevalence of this term. After the appearance of WWW, all individuals have become able to search and retrieve whatever is desired from this developing ocean of information hosted by web pages, weblogs, and forums. The retrieved information could be in the form of a text, an image, a video or an audio. "In a simple definition, Information Retrieval means finding and retrieving a set of records - documents in this case - which are relevant to user query". Therefore, for retrieving information, it merely needs to form a query and request it via one of the browsers - such as Firefox - from WWW to find those records more relevant to the query and then deliver them to the user. Retrieval systems rank documents located in a document collection in terms of their relevance score to the query. Afterwards, the ranked documents are displayed from the document with the highest rank to the one having the lowest rank (in trolley form).

One important point is efficiency of the methods used for ranking the documents, since when the ranking operation is done properly, the documents having more relevance with the query will be demonstrated. In this case, efficiency would be more important than effectiveness. It should be noted that web pages with conventional text documents which are used in traditional IR systems are completely different. There are numerous hyperlinks and anchor texts in web pages which are not seen in the traditional documents. Furthermore, web pages are semi-structured and there are tag tokens in a web page, in addition to content tokens, which have usually surrounded the content tokens. Moreover, JavaScript codes and CSS codes are observed in a typical web page in addition to HMTL tags. However, the most important components which are considered in this thesis are the headline and main content of each web page which are surrounded by HTML tags. Content tokens in a web page are located in a number of structural blocks. Some of these blocks are important in this contribution because they include the main content, while some others are unimportant such as involve content tokens which exist in menus, advertisements, privacy policy and copyright notice.

A very simple architecture of an IR system is depicted in Figure 2.1 (**?**). Regarding Figure 2.1, it seems necessary to answer the following questions:

- How are documents - in this case web pages - indexed?
- How does the retrieval system find the correct records out of the indexed documents and give them to the user?

(Noteworthy is that the same system will be utilized to find the headline of web pages, but the document collections here are in fact blocks of a web page which include content

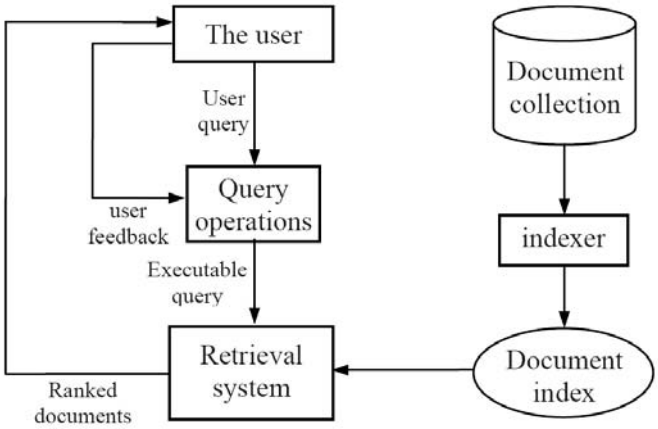


Figure 2.1: A general IR system architecture (73)

tokens). In Figure 2.1 the user commences to build a query considering the information desired, and then this query is issued for the retrieval system by query operation module. The retrieval system uses document index to determine those documents which contain several query items. Afterwards, it calculates retrieval scores for these documents and finally rank the retrieved documents based on the scores obtained. At this time, the ranked documents must be displayed so that the user could find his desired information there. One unit which is of great importance in Figure 2.1 is the indexer. Indexer is a module that receives original new documents, transfers them into a data structure, and indexes them to provide an efficient retrieval.

2.2.2 Information Retrieval Models

IR models answer how a document and a query are represented in computer systems and also how the relevance of a document is defined to a user query. Generally speaking, there are four main IR models including (61):

- Boolean Model
- Vector Space Model
- Language Model
- Probablistic Model

The most used models in IR systems and web are the first three models. However, this thesis will only explain and describe Boolean and VSM models. Both Boolean and VSM models deem any document or user query as a "bag" of words or terms. Nevertheless, sequence of terms and their positions are disregarded in the sentences of documents and user queries, which means that a document or user query is described by a set of distinctive terms. In other words, it does not mind here whether a term is noun, subject, verb, object and etc. These terms can be placed in a document or query in any arrangement.

In an IR model, a weight is associated to any term. Assume a set of documents D, and take $V = \{t_1, t_2, ..., t_{|V|}\}$ as a set of distinctive terms in D. The set V is usually called Vocabulary, and $|V|$ is its size. A $w_{i,j} > 0$ is associated to any term t_i which exist in a $\mathbf{d}_j \in D$. Meanwhile, the term that does not appear in a document $\mathbf{d}_j$ would have $w_{i,j} = 0$. Each document $\mathbf{d}_j$ is presented by a term vector as shown below (see Equation 2.1):

$$\mathbf{d}_j = (w_{1,j}, w_{2,j}, ..., w_{|V|,j}) \tag{2.1}$$

The weight $w_{i,j}$ determines the degree of importance of t_i in document $\mathbf{d}_j$. The order of components (terms) is not important in a vector. A collection of documents can be simply displayed by a matrix at this kind of demonstration. Every term is an attribute, while each weight corresponds to an attribute value in this matrix. The weight $w_{i,j}$ is calculated and determined in various forms for different retrieval models.

2.2.2.1 Boolean Model

The Boolean Model (BM) is one of the most straightforward and basic information retrieval models. In BM, only documents which have been matched with the user query are retrieved and offered to the user. Gudivada et al. have suggested calling this model "set theoretic" (51). In this model, documents and queries are represented as a set of representative keywords which are known as index terms. Each term can exist or not exist in a document:

$$w_{i,j} = \begin{cases} 1 & \text{if } t_i \text{ appears in } \mathbf{d}_j \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

In this technique of weighting, all terms in a document will be weighted equally which is in fact one of the major disadvantages of BM.

In BM, query terms can be combined with each other using Boolean operators such as AND, OR, and NOT to form a Boolean query. Assume that query q is given, then the retrieval system retrieves those documents which cause query q to obtain the true value, and provides selected documents to the user. It can be declared that the operation of retrieval is based on binary decision making. In other words, a document is either relevant or irrelevant, thus this form of retrieval is also known as "exact match". This way of retrieving is one of the most important drawbacks of BM, since it is unable to rank the retrieved documents and the users are not completely satisfied with the obtained and retrieved results.

Here, it seems reasonable to consider a definition of BM which has been introduced by Baeza-Yates in his book "Modern Information Retrieval" (18).

Definition 2. *Given a collection of documents D, let $V = \{t_1, t_2, ..., t_{|V|}\}$ be the set of distinctive terms in the collection, where t_i is a term. Further, let g_i be a function that returns the weight associated with the index term t_i in any t-dimensional vector (i.e., $g_i(\vec{\mathbf{d}}_j) = w_{i,j}$). For the Boolean model, the index term weight variables are all binary i.e. $w_{i,j} \in \{0, 1\}$. A query q is a conventional Boolean expression. Let $\vec{q}_{dnf}$ be the disjunctive normal form for the query q. Further, let $\vec{q}_{cc}$ be any of the conjunctive components of $\vec{q}_{dnf}$. The similarity of a document $\mathbf{d}_j$ to the query q is defined as*

$$sim(\mathbf{d}_j, q) = \begin{cases} 1 & if \quad \exists \quad \vec{q}_{cc} \quad | \quad (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall t_i, g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ 0 & otherwise \end{cases} \tag{2.3}$$

If $sim(\mathbf{d}_j, q) = 1$ then the Boolean model predicts that the document $\mathbf{d}_j$ is relevant to the query q (it might not be). Otherwise, the prediction is that the document is not relevant.

About storing documents, the simplest data structure often utilized in BM to store the documents is the binary term-document incidence matrix. Indexed units are indeed terms (words here) within this $2D$ matrix. Every row of this matrix belongs to a term t_i of vocabulary while each column of it belongs to a document $\mathbf{d}_j$. If the term t_i exists in the document $\mathbf{d}_j$, then value "1" will be stored at row i and column j of this matrix; otherwise, value "0" will be inserted there. Figure 2.2 illustrates this data structure.

		Documents						
		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Terms	Antony	1	1	0	0	0	1	
	Brutus	1	1	0	1	0	0	
	Caesar	1	1	0	1	1	1	
	Calpurnia	0	1	0	0	0	0	
	Cleopatra	1	0	0	0	0	0	
	mercy	1	0	1	1	1	1	
	worser	1	0	1	1	1	0	
	...							

Figure 2.2: A term-document incidence matrix (78)

Another data structure which is used for BM is inverted index, which is more advantageous than the term-document incidence matrix since it needs a much smaller space for data storage. This data structure is comprised of two parts: vocabulary which contains terms, and posting list. There is one list for each term in which the number of documents containing this term has been specified. Figure 2.3 shows this data structure.

2.2.2.2 Vector Space Model

Vector Space Model (VSM) (105) (107) is the most common and applicable model of IR. Each document $\mathbf{d}_j$ and each user query q is demonstrated by a t-dimensional vector in this model (Equations 2.4 and 2.5).

$$\vec{\mathbf{d}}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j}), \quad w_{i,j} \geq 0 \tag{2.4}$$

$$\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q}) \tag{2.5}$$

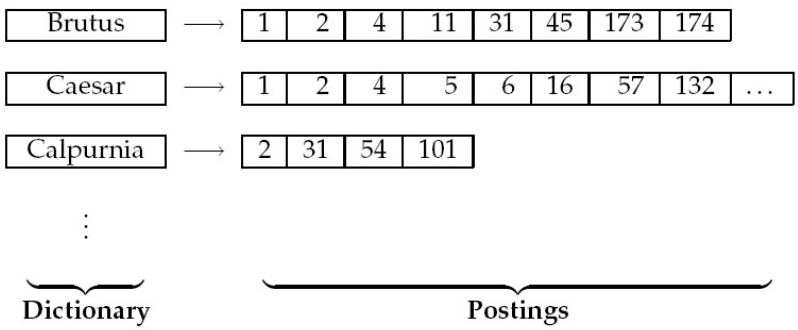


Figure 2.3: An inverted index data structure((78))

Appropriate weights of non-binary ≥ 0 are attributed to the index terms in query vectors and document vectors in this model. Therefore, it becomes possible to calculate the degree of similarity (DOS) between each document vector and query vector. Afterwards, a number of documents with the maximum DOS are chosen by sorting the documents in a descending order based on DOS for a specific query and then delivered to the users according to their needs and considering a threshold. It is appreciable that how the documents are chosen in VSM is more accurate and efficient than how those are chosen in BM.

In order to calculate DOS among document $\mathbf{d}_j$ and user query q, one just needs to benefit from the correlation between $\vec{\mathbf{d}}_i$ and $\vec{q}$ vectors. This correlation can be assessed by measurement on the cosine of the angle between these two vectors. Figure 2.4 depicts this more clearly. To obtain and measure this correlation, Equation 2.6 can be used, where $|\vec{\mathbf{d}}_i|$ and $|\vec{q}|$ represent norms of document and query vectors. One noticeable point which seems interesting is that the value of $|\vec{q}|$ is ineffective on the ranking of the documents, since it has the same value for all documents.

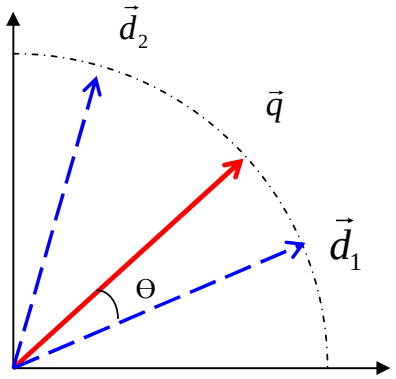


Figure 2.4: Illustration of Cosine Similarity. $score(\vec{q}, \vec{\mathbf{d}}_1) = \mathbf{cos}\theta$

$$\cos(\vec{q}, \vec{\mathbf{d}}_j) = \frac{\vec{q} \cdot \vec{\mathbf{d}}_j}{|\vec{q}| \times |\vec{\mathbf{d}}_j|} = \frac{\sum_{i=1}^{t} w_{i,q} \cdot w_{i,j}}{\sqrt{\sum_{i=1}^{t} w_{i,q}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{i,j}^2}} \tag{2.6}$$

However, it should be discussed how index term weights are calculated. One may argue that the most important disadvantage of the VSM is that the values of index terms must first be defined. Assignment of appropriate values to the index terms are known as term weighting. Salton et al. (109) (105) have demonstrated in their studies that the term weighting is by no means a simple and trivial problem. This section introduces the terms frequency scheme and TF_ IDF scheme which are usually used to calculate index term weights. As mentioned earlier, $w_{i,j}$ of term t_i in document d_j is no longer limited to 0 or 1 and can have any positive value as well.

- Term Frequency (TF) Scheme: In this method, weight of term t_i in document d_j is equal to the number of times the term t_i appears in the document d_j and is demonstrated by $f_{i,j}$. In order to normalize $f_{i,j}$, Equation 2.7 can be used. If the term t_i does not appear in the document d_j, then $tf_{i,j}$ will be zero. Here, $|V|$ gives the size of vocabulary. One disadvantage of the TF scheme is that it ignores the situation where a term appears in many documents. The term t_i might fail to be discriminative in this case.

$$tf_{i,j} = \frac{f_{i,j}}{\max\{f_{1,j}, f_{2,j}, ..., f_{|V|,j}\}} \tag{2.7}$$

- TF_ IDF Scheme: This scheme is one of the most popular methods of weighting scheme (see (73, 78)). TF and IDF stand for Term Frequency and Inverse Document Frequency, respectively. Different varieties of this scheme exist, though the simplest of them is investigated in this contribution. Assume N as the total number of documents in the document collection and df_i being indicative of the number of documents in which t_i appears at least once there. Thereby, idf_i of term t_i would be calculated through Equation 2.8. It can be seen and understood that if a term is repeated several times in a number of documents, that term will be unimportant and also not discriminative. Based on the information obtained so far, TF_ IDF can be calculated by Equation 2.9.

$$idf_i = \log \frac{N}{df_i} \tag{2.8}$$

$$tf\_idf_{i,j} = tf_{i,j} \times idf_i \tag{2.9}$$

The $tf\_idf_{i,j}$ weighting scheme assigns a value to the term t_i in the document $\mathbf{d}_j$ which can involve one of the following conditions:

- If the term t_i appears several times in a small number of documents, then the term t_i will get the greatest weight among all other terms. As a result, this term will give high discriminating power to the documents in which it appeared.
- If the term t_i appears fewer times in a document or if this term is observed in numerous documents, then the term t_i will get the lowest weight among all other terms. As a result, this term will behave as a ordinary term.
- If the term t_i exists almost in all documents, then the term t_i will have the lowest weight. Stopwords are usually classified under this group of terms.

It can be noted here that calculating the weight of terms in a query q is exactly similar to that of terms in documents. For example, Salton et al. (106) have proposed Equation 2.10 for calculating weight of terms in a query q.

$$w_{i,q} = \left(0.5 + \frac{0.5 f_{i,q}}{\max\{f_{1,q}, f_{2,q}, ..., f_{|V|,q}\}}\right) \times \log \frac{N}{df_i} \tag{2.10}$$

2.3 Content Extraction

This section discuss the concept of CE more precisely and also to provide a formal definition for it. Then, it will be investigated how to evaluate algorithms of MCE and how accurately they extract MC from web pages (71).

2.3.1 Formal definition of content extraction

If one needs to explain characteristics of CE algorithms in addition to evaluate their outputs, one must have a formal definition of CE. Here, we consider an HTML file as a sequence of tokens [1] and, consequently, MC would be a subsequence of this HTML file but not necessarily a continuous subsequence. The CE algorithm can be taken as a function whose domain is an HTML file, i.e. source code, and its range is a set of ordered pairs. Each pair contains start and end points of a part of MC in the source code. In Equation 2.11, D represents the HTML file, so $|D|$ would give the number of tokens in that file. s_i and e_i are the positions of the first and last character of a fragment of MC and they must definitely have relevance in the Equation $1 \leq s_i \leq e_i \leq |D|$. The number of fragments in the source code is N.

$$f_{CE}: D \rightarrow \{(s_i, e_i): 1 \leq s_i \leq e_i \leq |D|, \quad 1 \leq i \leq N \ , \quad s_{j+1} > e_j, \quad 1 \leq j < N\} \tag{2.11}$$

[1] According to the MC extraction approaches, an HTML file is considered as a sequence of characters or tokens.

2.3.2 Evaluation of main content extraction approaches

In order to evaluate the performance of the MCE algorithms, one might bear in mind to collect a great number of web pages from different domains in order to use them in the process of algorithm evaluation. The term "data set" is utilized for the collected web pages. Two data sets are being used in this thesis to examine and test the proposed algorithms. The first data set was prepared and collected by Thomas Gottron (46) in 2007, while the second one was collected by the author of this thesis in 2011. Each of these two data sets contains numerous web pages collected from various domains.

The second task to do in order to calculate the MCE algorithm accurately is to determine manually the MC of each web page in the data sets using an independent expert and then save each of these manually crafted MCs in separate text files. Such files are called *gold standard file* or *ground truth file*. It is obvious that this would be a time consuming and expensive activity which is associated with human errors. Therefore, if the gold standard files are not prepared with great care and precision, it will be possible to make some mistakes in evaluating the performance of the MCE algorithms. In other words, the algorithm which is able to extract MC at great accuracy might be misinterpreted as an inappropriate and weak algorithm; vice versa, an algorithm with a poor ability to extract MC from web pages might be incorrectly rated as appropriate and good.

By preparing data sets and gold standard files in steps 1 and 2, it would then be time for step 3. At this step, the algorithms of MC extraction input each of the web documents presented in data sets and then process each of them to extract the MC of collected web pages. The output of the MCE algorithms is usually a text file and it is expected to contain the MC of a processed HTML file. Van Rijsbergen (103) has labeled this output "retrieved item", though it is named as "extracted content" in this thesis with the file containing it being called "cleaned file".

However, the last part of this section will investigate how to evaluate the accuracy of the MCE algorithms. This thesis has employed *classical information retrieval performance measure* including **Recall**, **Precision** and **F1-measure**. Then some variables are introduced here: g and m show the number of tokens in gold standard file and cleaned file, respectively. k represents the number of tokens which are retrieved by *LCS (Longest Common Subsequence)* function. LCS is used to find the overlap between two files of gold standard and cleaned file.

Considering the defined variables of g, m and k, Recall, Precision and F1-measure can be introduced as in Formula 2.12:

- Recall r: Includes attribution of extracted relevant items to all relevant items;
- Precision p: Includes attribution of extracted relevant items to all extracted items ;
- F1-measure F1: Provides us with a suitable similarity measure for comparing extracted items and relevant items based on two concepts of Recall and Precision.

$$\mathrm{r} = \frac{\mathrm{length(k)}}{\mathrm{length(g)}}, \quad \mathrm{p} = \frac{\mathrm{length(k)}}{\mathrm{length(m)}}, \quad \mathrm{F1} = 2 * \frac{\mathrm{p} * \mathrm{r}}{\mathrm{p} + \mathrm{r}} \tag{2.12}$$

Metrics of Recall, Precision and F1-measure can have values between 0 and 1. It is obvious that as much as the values of F1 metric are closer to one, the method applied for MC extraction will show a higher accuracy.

2.4 R2L Languages, Unicode, and UTF-8 Encoding Form

Various languages use different directions in writing. Languages such as English, Spanish, and German are written from left to right, while languages such as Arabic, Farsi, Pashto, and Urdu are written from right to left. Our approaches R2L and DANA are able to extract the main content of right to left language web pages. Therefore, in this section we will explain some characteristics of these languages as well as their representation and encoding.

2.4.1 Languages on the Web

Figure 2.5 provides statistics concerning the top 10 languages of Internet users (12). Following English, Chinese, Spanish, Japanese, Portuguese and German, users speaking Arabic – one of the four languages discussed in this chapter – rank at position 7 of this list. Considering the statistics from the Internet World Statistics (IWS), in 2011 more than 33.5% of people in Arabian countries have access to the Internet. By exploring these statistics further, we can see around 3.4% of all users in the world come from Arabian countries. By comparing this value with older statistics from 2000 where only 0.8% of all users in the world were from Arabian countries, we see a high growing rate of Internet use in Arabian countries. It is important to know that the population of Arabian countries is more than 350 million people and in the future, even more people in these countries will have access to the Internet. Therefore, the numbers of visitors of Arabian web pages are going to increase. These considerations motivate doing MCE research on Arabian web sites.

2.4.2 Unicode Character Set

Before the Unicode Character Set (13) was introduced, ASCII (developed to ISO 8859*) and EBCDIC were used on computers. Thereby, only one byte was allocated for storing a single character; consequently only 256 characters could be coded. By considering this limitation, rows in the interval [128, 255] in the encoding table were used by different characters of different languages. Since the introduction of UCS, where only one special number was mapped to each character, we are able to use all characters of different languages on computers. At first, from 1991-1995, only 16 bits were reserved for each character, but when the new version of UCS was introduced (July 1996), it was possible to save a character in 21 bits. The newly defined UCS encoded all characters in the interval [U+0000, U+10FFFF] (32). There are several encoding forms in UCS, such as UTF-8, UTF-16, and UTF-32. In each of these encoding forms, respectively, one character can be saved in one to maximally four bytes, one or two words, or 32 bits.

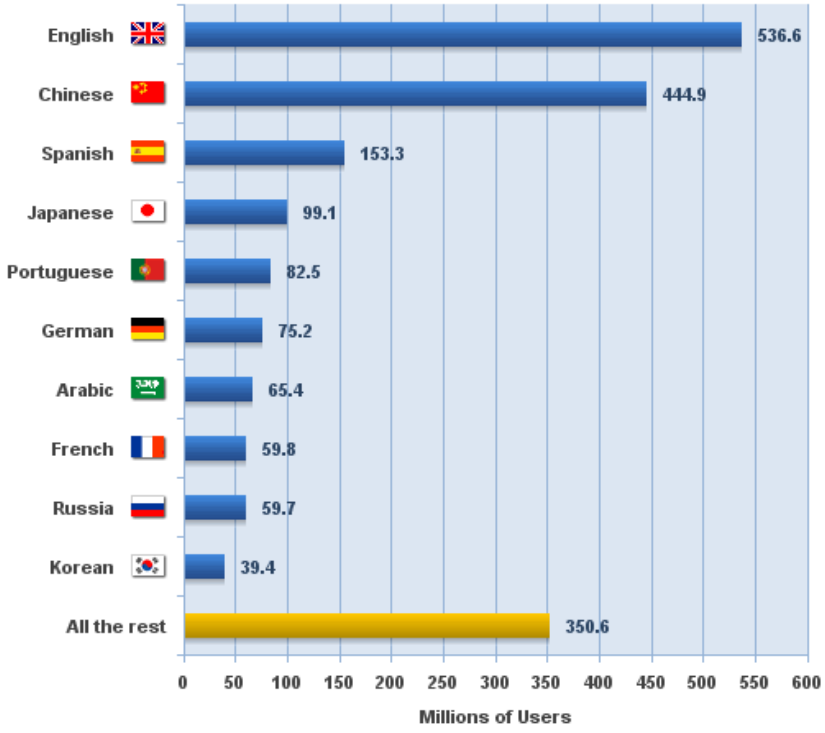


Figure 2.5: Top 10 languages on the Internet in millions of users in 2010 2.5

2.4.3 UTF-8 Encoding Form

As we mentioned in 2.4.2, UTF-8 is a variable-length encoding form in UCS. This encoding form can code all characters in UCS and represents each character in one to four bytes and has two of the following special characteristics (Keep in mind that characters which are used in non-English languages need two, three, or four bytes):

- It reserves the same character codes from ASCII that makes UTF-8 backward-compatible with ASCII. Hence, every valid ASCII character (a 7-bit character set) is a valid UTF-8 character sequence and is mapped onto the following scheme. Each of these characters has a value of less than 128.

Bits	Last code point	Byte1
7	U+007F	0xxxxxxx

- It is capable of encoding up to 2^{31} characters and uses the scheme in Table 2.1 to handle code points with up to 31 bits. Some features of this scheme are: 1) For every UTF-8 byte sequence, the first byte determines the length of the sequence in bytes. 2) The rest bytes have 10 as their two most significant bits (bits 7 and 6), so it can be recognized whether or not a byte is the first byte in a longer byte sequence.

Table 2.1: Scheme of byte sequence in UTF-8 (121)

Rows	Bits	Last code point	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
1	7	U+007F	0xxxxxxx					
2	11	U+07FF	110xxxxx	10xxxxxx				
3	16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx			
4	21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
5	26	U+3FFFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
6	31	U+7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

All letters of non-English-languages which we will discuss in this chapter take exactly 2 bytes, for example the Arabic character set has been represented in the interval [U+0600, U+06FF], and follow the byte sequence in Row 2 of Table 2.1. In UTF-8, each character which needs more than one byte will be coded in such a manner that each byte of this character is greater than 127 and so it can be distinguished from one-byte characters with values less than 128 (see Table 2.1).

To illustrate this, consider the following example. The letter ب in the Arabic language (corresponding to the letter *b* of the Latin alphabet) has been defined with the value 0x0628 (58) (with the equivalent binary value of 0B|00001100|00101000). According to the second row of Table 2.1, this value should be divided into three parts:

000	011000	0101000

Now, two right parts will be added to the bytes corresponding to the bytes in row 2 in Table 2.1, respectively :

11000000	10000000

The result is:

11011000	10101000

Note that the value of each of these two bytes is greater than 127. Therefore, we can easily separate one-byte characters with value less than 128 from double-byte characters by considering the first bit.

Chapter 3

Related Works

This chapter aims to categorize the developed algorithms and the invented frameworks for the extraction of main content from web pages of various aspects. Then it would be possible to put the algorithm proposed by the author of this thesis into the correct category. Main content extraction algorithms will be classified and studied from four different aspects in Section 3.1. Relevant algorithms of main content extraction with high performance will be introduced in Sections 3.2 and 3.3. Then, content extraction systems will be addressed in Section 3.4, and finally comparable platform for boilerplate removal is discussed in Section 3.5.

3.1 4-Dimensional Classifications of Main Content Extraction Algorithms

This section mainly focuses on the classification of main content extraction algorithms which is comprised of total four categories, i.e. 4-dimensions.

3.1.1 Single Document based Approaches vs. Multi Document Template Detection Approaches

Algorithms developed for identification and extraction of main content from web pages are categorized to the following general categories in terms of using single document or document collection, generated by WCMS (web content management system):

- Single Document Based Approaches (SDBA)
- Multi Document Template Detection Approaches (MDTDA)

Gibson et al. (42) has called these two groups *"Local Techniques Approach"* and *"Global Technique Approach"*, respectively. It is well known that web pages which are made by WCMS have a similar template structure. In other words, template portion exists in all web pages generated by typical WCMS. Therefore, it seems possible to identify and remove the following segments to attain the main objective, i.e. extraction of the main content in these document collections, by investigation and analysis of the template structure in a document collection:

- Navigation slide bars
- Logos
- Header and dropdown menus
- Banner advertisements and footer

Earlier works of Kolcz et al. (70), Chen et al. (28), Yi et al. (127), Ma et al. (76), Wang et al. (119), and Bar-Yossef et al. (19) give useful information about MDTDA. Furthermore, algorithms of FastContentExtractor (93), ExAlg (17), RTDM-TD (118) and NoiseEliminator (63) employ MDTD-based techniques to identify main content in web pages.

Main content is extracted in SDBA just through processing an HTML web document. It can be declared that one of the most powerful hypotheses in this method is that the main content is a continuous text in a web page. In other words, main content is a zone of single document whose content density is much greater than that of HTML tags as mentioned in Section 1.3 previously.

One advantage of MDTDA is that it does not use the assumption that main content is a continuous text, thus it will still be extractable even if main content is a non-continuous text or includes image, video and more complex structures including tables. It means that MDTDAs are more reliable in their extraction performance but their major drawback is that they can only be executed on template-based documents which are generated by WCMS. One other point is that in order to extract and process main content, namely in search engines, SDBAs are widely used. This is because many web pages which are to be processed are not generated by WCMSs, and thus it is not possible to apply MDTDAs for the extraction of main content. In other words, SDBAs can be used on any HTML web document to extract main content. Algorithms proposed in this thesis, by the author, are categorized as SDBAs. Therefore, the following text will study the algorithms classified under the SDBA group which have been presented previously. Chapter four will explain the algorithms developed by the author of this thesis including R2L, DANA, DANAg and AdDANAg.

Jan Pomikalek (97) has classified the algorithms used to extract main content into two groups of *"page-level"* and *"site-level"* associated with SDBA and MDTDA, respectively, in his PhD thesis entitled "Removing Boilerplate and Duplicate Content from web Corpora".

3.1.2 Stand-alone vs. Integrated Approaches

Since current requirements need to display the main content of web pages on different platforms, it seems thus necessary to automatically identify, extract, and display the main content in web pages on various platforms. In such cases, the main content extraction algorithm (MCEA) performs singularly and it is no longer accounted for a segment of another software. These algorithms are called *stand-alone* main content extraction algorithms. On the other hand, since the main content of web pages are valuable information sources, they can be extensively used in the following projects:

- Text classification, clustering and summarization systems
- Question and Answering systems

- Trend analysis systems
- Indexing systems and search engines

In such cases, MCEA is not considered as a stand-alone algorithm itself, but it is considered as a pre-processor for the main program instead. These algorithms are called *integrated* main content extraction algorithms. As the output of MCEA is used by the main program, a high accuracy of MCEA in extracting the main content of web pages can affect the overall performance of the system significantly. That is why researchers are still looking to find algorithms for main content extraction which show both a relatively high performance as well as some characteristics such as being domain-independent and language-independent.

3.1.3 Heuristic Techniques vs. Machine Learning Approaches

Most algorithms used for main content extraction usually benefit from a series of *Heuristic Rules* to reach their final goal, which is main content extraction with high accuracy. A main feature, for example text-to-tag ratio, is utilized to identify and recognize the main content as well as its extraction from noisy information. However, based on the information obtained from heuristic algorithms, application of a main feature will not necessarily lead to successful results since various web pages contain different characteristics.

Another disadvantage of these heuristic methods is that one should define a set of parameters and consider a threshold for each of these parameters to raise the accuracy of MCEA. The existence of such parameters will definitely render them as domain-specific and thus they could not be used on all datasets. For example, most MCEAs experience a noticeable decrease in the accuracy during main content extraction from Wikipedia pages and some other web pages like Slashdot. For solving this problem, a combination of several features can sometimes lead to better results in the main content extraction process (86).

On the other hand, some MCEAs are based on *Supervised Machine Learning Methods (SMLM)*. These methods involve two steps as for any other SMLMs. At the first step, a function, a classifier, is generated by the analysis of training data. The training data have a set of training examples with each example being comprised of an input object (typically a vector) and a desired output value. In the second step, the generated classifier receives some valid input objects and predicts some correct output values. If the training process is implemented successfully at the first step, the developed classifier can make proper outputs on new and unseen data. Otherwise, the classifier will experience a failure.

3.1.4 Methods Based on DOM Tree Structure vs. Methods Based on HTML Source Code Elements

Algorithms and tools which are implemented for main content extraction use an "HTML DOM tree structure" or "HTML source code elements" (20) in order to identify and extract main content of the web pages. Looking on Sections 3.2 and 3.3 reveals that DOM tree structure and HTML source code elements are used almost equally in main content extraction algorithms.

3.2 Methods Based on DOM Tree Structure

This section adopts to introduce 8 different algorithms which use HTML DOM tree structure for extracting MC from web pages. These algorithms usually employ an "HTML parser" to produce the DOM tree.

3.2.1 Crunch [2002, Heuristic]

The impulse for the creation of crunch framework as one of the more prominent solutions for MCE was the necessity of transmitting and displaying the contents of web pages, especially their main contents, on devices with a small memory capacity such as screen devices, currently famous as tablets, mobile phones and PDAs. Therefore, developers of this framework in 2002 decided to enable transmission and displaying of main contents related to web pages on the devices mentioned before by implementing this framework (53, 54, 55, 56, 57). This framework initially uses an HTML parser (5) to build a DOM tree for an HTML document. Then, it navigates this DOM tree recursively rather than using a raw HTML markup and applies a number of heuristic filtering techniques to extract the main content of the HTML web page.

This framework in its first heuristic filtering technique deletes tags of CSS style, images and links from the HTML code. The second heuristic filtering technique has tried to delete advertisements, link lists and empty tables. For removing advertisements in this framework, a list of advertisement server addresses is kept to be used for identification of the advertisement elements in DOM nodes. However, the Link Quota Filter technique, which will be introduced in this chapter, has been utilized to decide whether the link lists should be deleted or remained in the HTML code. The removed links will be saved at the end of the HTML file in order to keep their navigation in web pages possible. However, the last part, which is deleting empty tables, could be done more simply. It only needs to delete a table without information or containing insignificant information. It is also useful to note that the crunch is equipped with a plug-in mechanism to enable adding heuristic filtering techniques. Crunch is written in Java and it was implemented in a publicly available web proxy (67). This proxy can be used both centrally, administrated for groups of users, as well as by individuals for a personal browser. The Crunch filtering algorithm (75) is shown in Algorithm 1(Page 35).

3.2.2 Mantratzis et al.[2005, Heuristic]

Mantratzis et al. (80) suggested a new algorithm in 2005 which has attempted to distinguish the main content in a web document from the hyperlink-clutters like text advertisements and long links of syndicated references to other resources. The suggested algorithm benefits from a DOM tree structure to implement this task. This algorithm determines the areas with a high hyperlink density within a web document, so it can separate these areas from the main content in web pages. In doing this, they examined the DOM tree and assigned specific scores to each section based on the amount and relative location of hyperlink nodes in the DOM tree. The main advantage of this method is that it is able to identify "list-link" structure of hyperlinks or "table-links" by processing various levels of DOM tree which have been loosely or tightly defined, and also to find the main content present in the web document

Algorithm 1: Crunch Filtering Algorithm (49)

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

by choosing these hyperlinks and removing them from the web document. Mantratzis et al. implemented their algorithm in Java (J2SE 5.0). Moreover, they employed two open-source softwares called JTidy (4) and JDom (3). The former gets an (X)HTML file as its input and finally offers a new error free file by applying several steps of filtering on the called file which can contain some errors. However, the latter is used to create a Java-accessible object representation.

3.2.3 FeatureExtracter and K-FeatureExtracter [2005, Heuristic]

FeatureExtracter (FE) and K-FeatureExtracter (K-FE) are two algorithms proposed by Debnath et al. (34) (33) in 2005 to extract the main content from web pages based on analysis of blocks in a DOM tree. Noteworthy here is how these two algorithms identify blocks of a web page. It is well known that every block corresponds to one sub-tree from a DOM tree. These blocks (sub-trees) are selected on the basis of specific elements such as `<table>`, `<tr>`, `<p>`, `<hr>`, and `<ul>`, which can be placed in the root node of a block. In other words, each sub-tree of DOM structure, whose root node is one of the above mentioned elements, can be considered as a block in these two algorithms. To identify all blocks, these two algorithms have used a recursive algorithm shown in Algorithm 2 (49).

Algorithm 2: Decomposing a document into blocks (49)

```
Input: D: DOM node (first called with root node of a document), T: Set of block defining
       HTML elements.
Output: B: Set of blocks.
begin
    B ← D ;
    foreach t ∈ T do
        foreach b ∈ B do
            if b hasChildNode(t) then
                B ← (B \ b) ∪ getBlocks(b,t);
    return B
function getBlocks(b,t);
begin
    NewBlocks ← ∅;
    C ← descendants(b);
    foreach m ∈ C do
        if elementType(m) = t then
            NewBlocks ← NewBlocks ∪ {m};
    return NewBlocks
```

Having characterized all blocks in a DOM tree for identification and isolation of the main content blocks from non-content blocks in these two algorithms, one must attribute special features to the extracted blocks. These features are corresponding to elements which are located in the same block. For example, if FE is called with features related to text, image or links, then FE will identify text blocks, image blocks or navigational blocks. Since FE decides to identify the main content blocks, it has just adopted to introduce features corresponding to the text. Obviously, changing these features would make it possible to identify other blocks with different features instead of the main content blocks.

The features which have been addressed for identification of the main content blocks in works of Debnath et al. include:

- Text: the text content inside the block
- Text-Tag: the text tags, i.e. `<h1>` and `<h2>` inside the block
- List: the list available inside the block
- Style-sheet: This is also to make the list complete and compliant to W3C guidelines. Styles are usually important for browser rendering, and usually included inside other tags, such as links, tables, etc.

At this step, the FE algorithm attributes a value to each block based on the features considered. Afterwards, if the sum of feature values related to the desired features is greater than or equal to the sum of feature values corresponding to the remaining features, then the desired block will be transformed into a winner-basket. In the second step, feature values

will be calculated once again for a new set of blocks wherein the block with the highest sum of desired values being selected as the winner block.

To explain K-FE, it can be pointed out that this algorithm is in fact the extended version of FE which adopts to select K blocks rather than a block. K-FE uses an adaptive k-means clustering to address K blocks from the winner baskets discussed earlier instead of introducing just one winner.

3.2.4 Link Quota Filter (LQF) [2005, Heuristic]

LQF is a simple heuristic method which has been utilized by Mantratzis (79, 80) and Gupta (54, 55, 56) to delete both link lists and navigational elements from the body of a web document and thus reach a much higher accuracy in extraction of the main content. Although various versions of LQF have been implemented, the main idea in all these processes is to measure the ratio of hyperlinked contents to non-hyperlinked contents inside a DOM node. If the value of the calculated ratio is greater than the threshold defined by the user, the DOM node will be deleted. Otherwise, the DOM node will remain in the structure of DOM tree. It must be noticed that LQF readily removes additional contents with a high frequency of the hyperlinked contents from DOM tree structure. However, it fails to achieve the desired results against some parts of the web document including header and footer since not many hyperlinks could be seen in this section. The LQF algorithm is shown in Algorithm 3 (49).

Algorithm 3: Link Quota Filter (49)

```
Input: n: DOM node
Output: q: quota of links to overall text
begin
    C ← descendants(n) ;
    t_tot ← 0 ;
    t_link ← 0 ;
    foreach m ∈ C do
        if ¬isBlockNode(m) then
            if isTextNode(m) then
                t_tot ← t_tot + length(getText(m)) ;
            else if isLinkNode(m) then
                t_tot ← t_tot + length(getText(m)) ;
                t_link ← t_link + length(getText(m)) ;
            else
                t_tot ← t_tot + length(getText(m)) ;
                t_link ← t_link + Linkquota(m) · length(getText(m)) ;
        else
            C ← C \ descendants(m);
    q ← t_link/t_tot;
    return q
```

3.2.5 VIsion-based Page Segmentation (VIPS) [2003, Heuristic]

Cai et al. (22, 23) introduced the VIPS algorithm for the first time in 2003. This algorithm uses DOM structure and analyzes visual page layout features (cues) such as position, background color, font size, font weight and etc. to build a *vision-based content structure* (or simply a vision-based tree) for the web document under study. For this purpose, the algorithm first extracts the blocks present in an HTML DOM tree heuristically. It then locates the identified blocks based on the visual features of each block as well as the visual separators between blocks such as horizontal and vertical lines. Figure 3.1 shows the layout structure and the *vision-based tree* for an arbitrary web document (22).

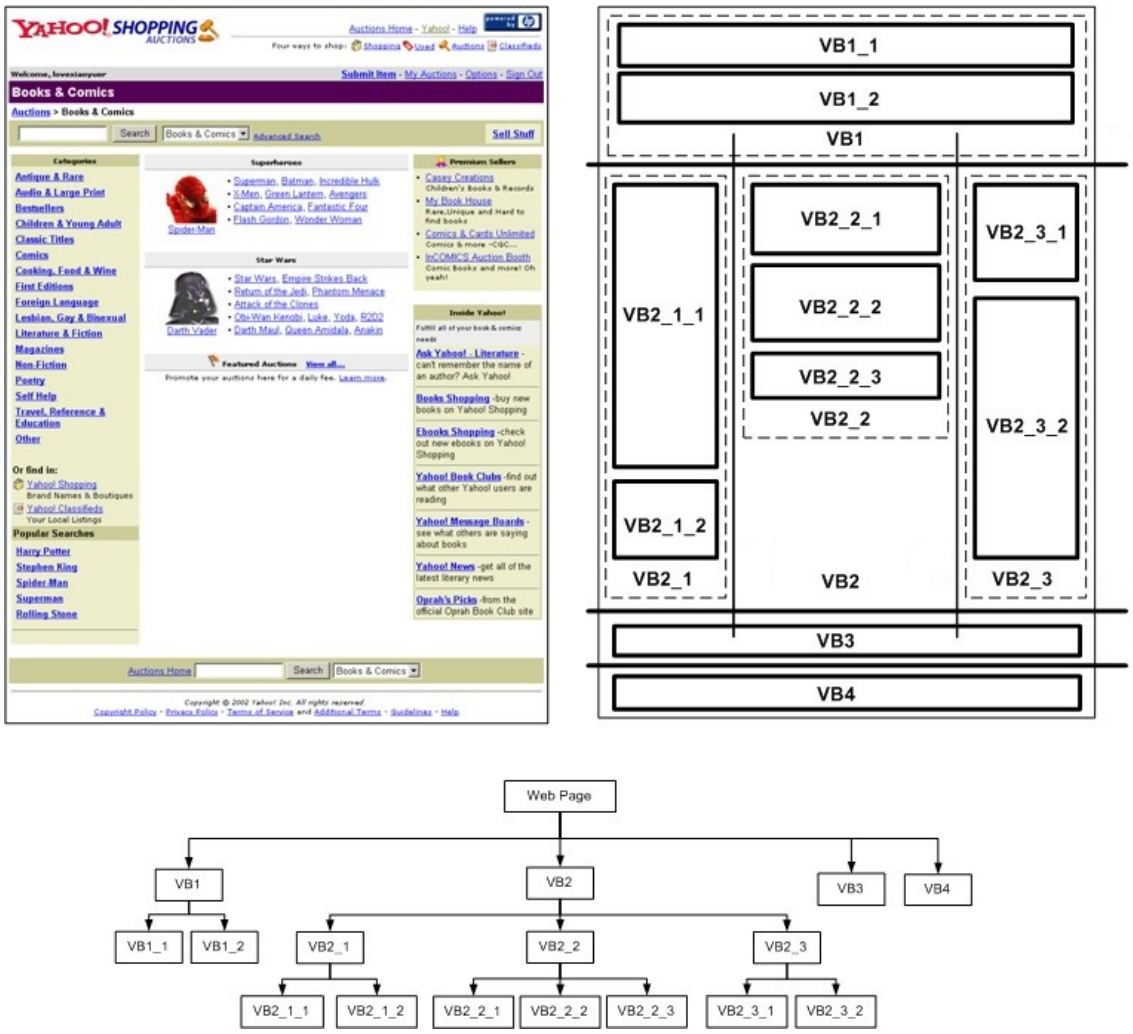


Figure 3.1: The layout structure and vision-based tree of an example page

As can be seen in Figure 3.1, this draft has four nodes at its first level in the order of $VB1$, $VB2$, $VB3$ and $VB4$ from left to right. It is evident that a visual-based tree is different from a DOM tree since the existing blocks in a web document are grouped visually in the visual-based tree. Since the blocks in one group are almost similar in the vision-based draft, the vision-based tree is described as "semantic structure" in some literature (24).

On the other hand, it is required to identify the nodes including the main content in vision-based tree for the extraction of the main content. Unfortunately, the VIPS algorithm is unable to characterize the nodes as content or non-content and thus cannot label them in terms of content or non-content nature. If a mechanism is discovered for labeling the nodes of vision-based tree to content and non-content, the VIPS will be able to extract the main content with a high degree of accuracy. Nevertheless, such a labeling mechanism has not yet been found. Finally, the VIPS is a resource intensive algorithm, since it sometimes needs to refer to external style sheet files in order to locate the block extracted from a DOM tree in the proper place in the vision-based tree. This may reduce the accuracy of the algorithm.

3.2.6 Content-seeker [2009, Hybrid]

Content-Seeker is an approach proposed by Samuel Louvan (75) which is a combination of heuristic rules and machine learning techniques and provides the capability for main content extraction from web pages, weblogs and forum pages. The first part of this algorithm, segment and content classification, aims to break down the structure of a web document into smaller segments with certain granularity. For this purpose, a DOM tree corresponding to the HTML file and supervised machine learning methods (SMLM) have been utilized. The term "segment with good granularity" is used to address a segment which represents a *uniform semantic unit*. It has been initially tried in the labeled training data to categorize and label the segments into two groups, namely good granularity and bad granularity. Afterwards, the training data are given to each of the following four classifiers used in this contribution in order to train the classifier:

- Decision Tree(J48)
- Random Forest
- Sequential Minimal Optimization
- Multilayer Perceptron

Now the classifiers can be used on the evaluation data to categorize the segments of each file into good granularity and bad granularity segments. DOM nodes corresponding to good granularity segments can also be specified as the output and be delivered as the input for the next step, i.e. content classification. The content classification algorithm classifies the segment as either main content or noisy content considering the features of a segment including number of images, number of frames, number of links, position in DOM tree, its length and width, number of stopwords, etc. The purpose of the second part of this algorithm called "additional heuristic approaches" is to identify and remove the noisy data (such as embedded advertisement, link to related articles, comments from the web page visitors) if it is embedded in the body of the DOM nodes which are delivered to this step from the previous step.

Content-Seeker employs two heuristic techniques to identify the main content and delete the noisy content. The first method is more appropriate for web pages and weblogs launches to identify and specify the Largest Block of Text String in DOM tree for reaching its main objective, i.e. main content extraction. On the other hand, the second method, which is usually applied for web forums, performs this task by identification of Table Pattern used in forum posts there.

3.2.7 Content Extraction via Text Density (CETD) [2011, Heuristic]

Sun et al. (112) (2011) developed a highly effective algorithm for main content extraction from web pages which was also able to preserve its original structure information. They declared that in a typical web page, the parts regarded as noise are usually highly formatted with significantly small text size. On the other hand, the parts including main content have more text, are simply formatted, and the number of hyperlinks is very small in comparison with the noise part. The following steps are followed in CETD after making the DOM tree corresponding to HTML web document in order to distinguish main content from noisy information and to express it as output:

- Calculation of text density based on Formula 3.1 for each tag in the DOM tree: C_i denotes the number of characters in subtree associated with tag i, and T_i represents the total number of tags in subtree associated with tag i. TD will get a greater value for the node containing a long and simply formatted text, rather than a highly formatted one with little text. Algorithm 4 suggests an approach to calculate TD for all nodes in a DOM tree. Figure 3.2 depicts a histogram where the value of text density is provided for every node in the DOM tree by vertical lines. Nodes with relatively high text density can be simply observed in this histogram.

$$TD_i = \frac{C_i}{T_i} \tag{3.1}$$

Algorithm 4: Pseudocode of ComputeDensity(N) (49)

```
Input: N: DOM node
Output: TD: the value of Text Density
begin
    foreach child node C in N do
        ComputeDensity(C) ;
    N.CharNumber ← CountChar(N);
    N.TagNumber ← CountTag(N);
    if N.TagNumber == 0 then
        N.TagNumber ← 1 ;
    N.Density ← N.CharNumber/N.TagNumber;
    return N.Density
```

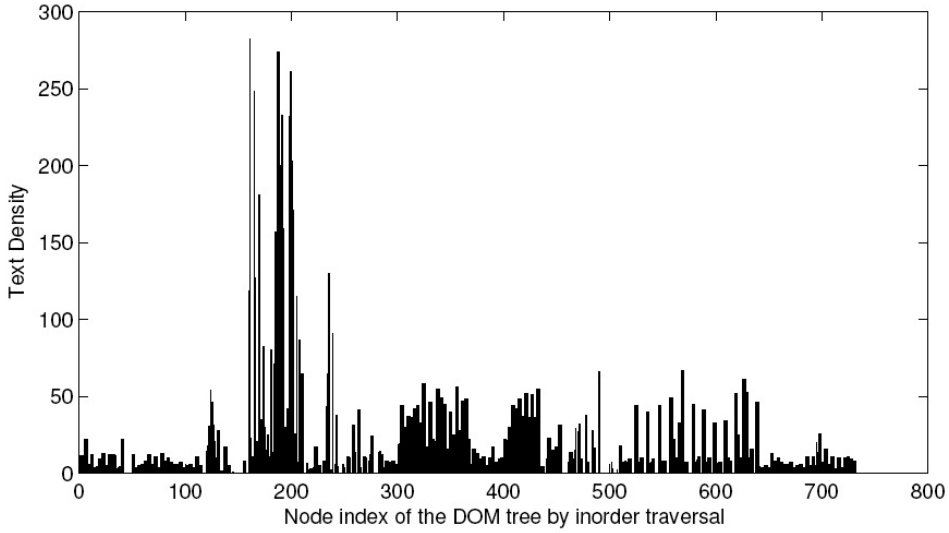


Figure 3.2: Text Density for each node from N.Y. Times web Page (112)

- Calculation of composite text density: It is probable that the node which contains a great number of hyperlinks and a little text cannot be accounted for an important node, thus it can be regarded as noise. On the other hand, the node with a small number of hyperlinks but too many texts will definitely be accounted as an important node and can be taken as content. CETD uses Formula 3.2 (Composite Text Density) instead of 3.1 to remove zones with great density of hyperlinks, where LC_i, NLC_i, LT_i, LC_b, and C_b are defined as below:
  - LC_i : number of all hyperlink characters under i
  - NLC_i : number of all non-hyperlink characters under i
  - LT_i : number of all hyperlink tags under i
  - LC_b : number of all hyperlink characters under the `<body>` tag
  - C_b : number of all characters under the `<body>` tag

$$CTD_i = \frac{C_i}{T_i} \log_{\ln(\frac{C_i}{\neg LC_i} LC_i + \frac{LC_b}{C_b} C_i + e)} (\frac{C_i}{LC_i} \frac{T_i}{LT_i}) \tag{3.2}$$

  In comparison to Figure 3.2, Figure 3.3 illustrates a histogram which has calculated the length of vertical lines for each node in the DOM tree through Formula 3.2. Furthermore, it can be observed that most of the vertical line associated with the noisy information is not seen in this histogram anymore. Thus, it would be rather simple to identify and extract nodes which include content.

- At the last step of the CETD algorithm, a threshold t is defined and considered to categorize the nodes in DOM tree to content and noise sections. Basically, any node

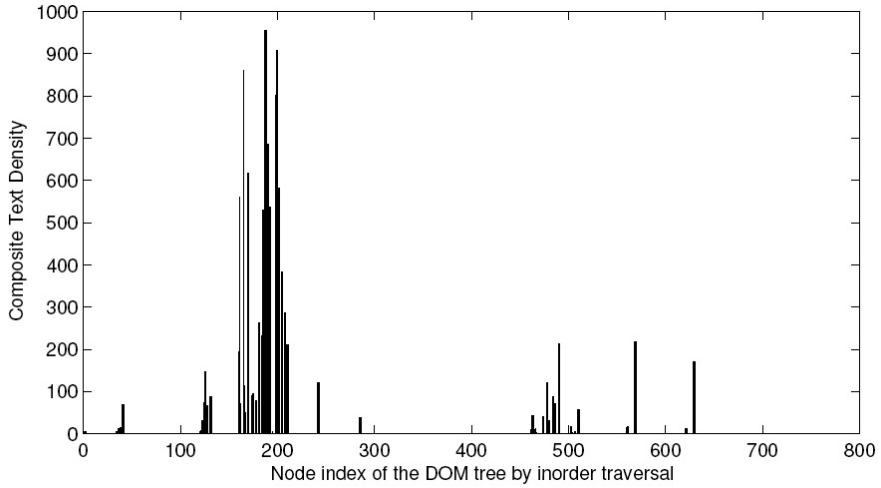


Figure 3.3: Composite Text Density for each node from N.Y. Times Web Page (112)

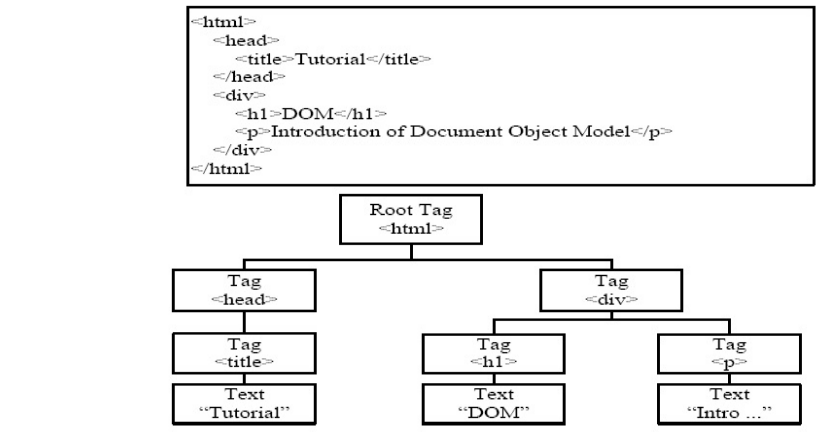
with a text density greater or equal to the threshold will be selected as content, while other nodes whose text density are smaller than the threshold will be labeled as noise. One common problem is finding the best value for the threshold t, since an inappropriate value for it would produce inaccurate results. Usually, `<body>` tag's text labeling is practically considered as the threshold.

3.2.8 Gaussian Smoothing-based Web Content Extraction (GSWCE) [2011, Heuristic]

The algorithm proposed by Liu et al. (74) called GSWCE has been inspired to a large extent from Content Code Blurring (CCB), which was developed by Gottron (48). In other words, GSWCE has utilized a Gaussian Blurring Filter exactly like CCB in order to identify the main content in web pages. However, this algorithm has also provided the ability to identify title and published date. Different steps of GSWCE algorithm can be briefly summarized as below:

- DOM Nodes List: every node in the DOM tree indicates an HTML element in the source web page such that the nodes in the DOM tree can be divided into two of the following categories:
  - structure, layout and format node
  - text nodes which make up all the content

  Through traversing DOM tree in preorder, all existing nodes in the DOM tree can be placed in one DOM nodes list (DNL), which in fact makes up the first row of the table labeled DNL in Figure 3.4.



DNL	<html>	<head>	<title>	Tutorial		<h1>	DOM	<p>	Introduction of Document Object Model				
TL	html	head	title	Tutorial	div	h1	DOM	p	Introduction	of	Document	Object	Model
RS	0.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
SRS	0.16	0.18	0.22	0.28	0.33	0.40	0.49	0.60	0.70	0.78	0.84	0.88	0.90
PCT	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
CBN									Content Base Node				

Figure 3.4: An HTML web Page with its corresponding DOM tree (74)

- Tokens List (TL): Using the first row of the table, tokens in tag nodes and text nodes were identified and then inserted in second row of the table labeled as TL.

- Gaussian Smoothing: Having observed the tokens in TL, it can be inferred that the tag tokens are significantly important once surrounded by the text tokens. On the other hand, the text tokens surrounded by the tag tokens are of little importance. Therefore, the main goal here was to calculate the importance of each token according to the importance of its surroundings. In order to calculate the effect of neighbors on each node, Gaussian Smoothing Algorithm (GSA) was applied on text-tag Ratio Sequence (RS) to find Smoothed Ratio Sequence for each node. Figure 3.4 lists the effect of applying GSA, after a number of iterations, on text-tag Ratio Sequence. The third row containes initial values of the RS. It can be seen that the tag tokens incorporate zero values, while text tokens show unit values. Meanwhile, the fourth row provides Smoothed Ratio Sequence for the values in Row 3.

- Content Base Node (CBN): at this step of GSWCE algorithm, a ratio threshold is considered to categorize tokens into content and noise groups. Tokens with values greater than ratio threshold will be considered as Potential Content Token (PCT) at the fifth row of Figure 3.4. The following will adopt to find the longest continuous sub-sequence of PCTs from token list. Finally, the node which contains higher number of PCT will be chosen as CBN.

- Primary (Main) Content Extraction (PCE): The least common ancestor of the PC node on DOM tree is sought to address the PC. As declared by the authors of this paper, PC for most web pages is included in `<p>`, `<tr>`, `<td>` or `<div>` tags. These kinds of tags have been defined as Tags Containing Primary Content (TCPC). Therefore, the nearest TCPC parent of CBN is the same common ancestor being searched with the content inside this TCPC node being in fact the same PC.

3.3 Methods Based on HTML Source Code Elements

As mentioned previously in Section 3.1.4, some main content extraction algorithms use HTML source code elements or in simple words HTML tags in order to extract the main content from web pages. Algorithms which will be discussed in Sections 3.3.1, 3.3.2 and 3.3.3 are of this type of algorithms. Most of these algorithms need to know whether the characters (tokens) in an HTML file are components of content characters (tokens) or non-content characters (tokens). For this purpose, a parser is usually used to recognize which type of component they are.

3.3.1 Character and Token-Based

Algorithms discussed in this section are categorized under character and token-based class. This is because the above mentioned algorithms take an HTML file as a sequence of characters (tokens) which certainly contain the main content in a part of this sequence. Having executed the algorithms of this section, a sequence of characters (tokens) is labeled as the main content and is provided to the user.

3.3.1.1 Body Text Extraction (BTE) [2001, Heuristic]

The work of Finn et al. (41) described the process of extracting and classifying information from HTML documents for the purpose of integrating it into digital libraries. They proposed the "Body Text Extraction" (BTE) approach, which identifies a single continuous fragment of the HTML document containing the MC. The BTE algorithm is based on the assumption that the main content in a web document is a single continuous block of text which also contains a few number of HTML tags (even perhaps no HTML tag may be seen in this text). In order to find the corresponding block, first all tokens present in the web document are identified. It is clear that the obtained tokens belong to HTML tags or they are words placed in a text segment. Therefore, the web document can be considered as a sequence of N token numbers represented by $B_0, B_1, B_2, \ldots, B_{N-1}$. Now, taking into account the policy of the BTE algorithm, the main objective would be to find a zone in this sequence of tokens, i.e. between indexes i to j, where there is a maximum value for the number of text tokens. Meanwhile, the number of tag tokens before index i and after index j must be greater than the number of tag tokens in that range. Finding such a zone just needs the function 3.3 to reach its maximum value.

$$T_{i,j} = \sum_{n=0}^{i-1} B_n + \sum_{n=i}^{j} (1 - B_n) + \sum_{n=j+1}^{N-1} (B_n) \tag{3.3}$$

Tag tokens and word tokens have been given the values one and zero in this function, respectively. An interesting interpretation of cumulative distribution of tag tokens in a web document has been depicted in Figure 3.5. It can be observed that a continuous plateau is indicative of the main content in a web document.

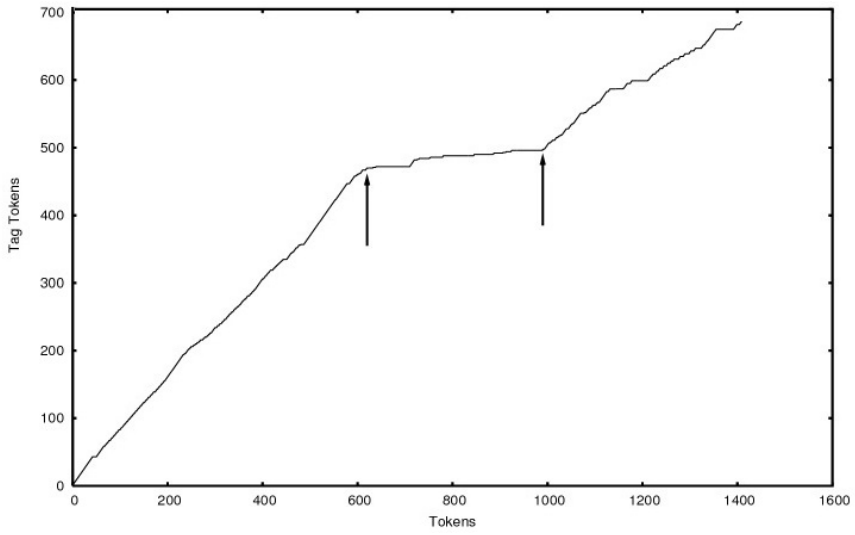


Figure 3.5: Cumulative distribution of tag tokens in a web document. The continuous plateau is indicative of the main content in a web document (41)

At this point, it would be important to note the main drawbacks of the BTE algorithm. The BTE algorithm just looks for a continuous block of tokens. Therefore, if the main content is composed of several parts, the BTE will provide us with that part of the main content since the Formula 3.3 has reported a much greater value in this range. The last thing to remember is that the BTE technique has been implemented on the basis of Algorithm 5 in the Python programming language and its module is accessible to everyone (1).

3.3.1.2 Document Slope Curves (DSC) [2002, Heuristic]

Pinto et al. (96) invented the Document Slope Curve (DSC), which is a heuristic method, in 2002 which can be claimed as a developed method for BTE. As mentioned previously concerning the BTE algorithm, it has a limitation, which is the extraction of only one continuous block from a web document. This drawback has been changed to an advantage in the DSC algorithm, since the latter is able to identify and extract more than one part from the main content once it is composed of more than one part.

The DSC method starts to characterize tokens of the web document while some of these tokens are tag tokens and the others are text tokens. Therefore, instead of using BTE formula and maximizing it, the function appeared in Formula 3.4 should just be calculated. This function explains the cumulative tag token distribution (Figure 3.6).

Algorithm 5: Finn's BTE algorithm (49)

Data: B: Token vector of an HTML document of length N, where B_i corresponds to the i-th token and has a value of 1 for tag tokens and 0 for word tokens.
Result: Index tuple $t_{index} = (i, j)$ with $i \leq j$ denoting the token range with the main content
begin

```
t_index ← (0, N − 1);
T_max ← 0;
// count tag tokens till index i
lookupForward[0] ← B_0;
for i = 1 ... N − 1 do
    lookupForward[i] ← lookupForward[i-1] +B_i;
// count tag tokens after index j
lookupBackward[N-1] ← B_{N−1};
for j = N − 2 ... 0 do
    lookupBackward[j] ← lookupBackward[j+1] +B_j;
// Search optimum
for i = 0 ... N − 1 do
    for j = i ... N − 1 do
        T_{i,j} ← lookupForward[i] + lookupBackward[j];
        for k = i ... j do
            // Add word tokens
            T_{i,j} ← T_{i,j} + (1 − B_k);
        if T_{i,j} > T_max then
            t_index ← (i, j);
            T_max ← T_{i,j};
return t_index
```

$$d(i) = \sum_{n=0}^{i} B_n, \quad for \quad 0 \leq i \leq N-1 \tag{3.4}$$

Now, because the main content contains many text tokens and just a few tag tokens, the parts having a low slope have been deemed as the main content in Figure 3.6. It can be observed that the main content is composed of five different parts. Noteworthy here is that if the web document does not include the main content, then Figure 3.6 will be transformed to Figure 3.7. Figure 3.7 shows that the number of text tokens in the web document has not been significant with the most tokens being tag tokens in nature.

The following question may arise: What is the exact definition of the low slope? How can the five parts represented in Figure 3.6 be extracted as the main content? How can all constituents of a main content be identified? Pinto has utilized a windowing technique to access his goals. Thus, it is necessary to determine the size of a window based on the web document. In other words, the size of a window is dependent on the number of tokens of the web document. For documents with up to 200 tokens, the size of a window is assumed to be 8 tokens, while those documents having 5000 tokens or even more, the size of the window is

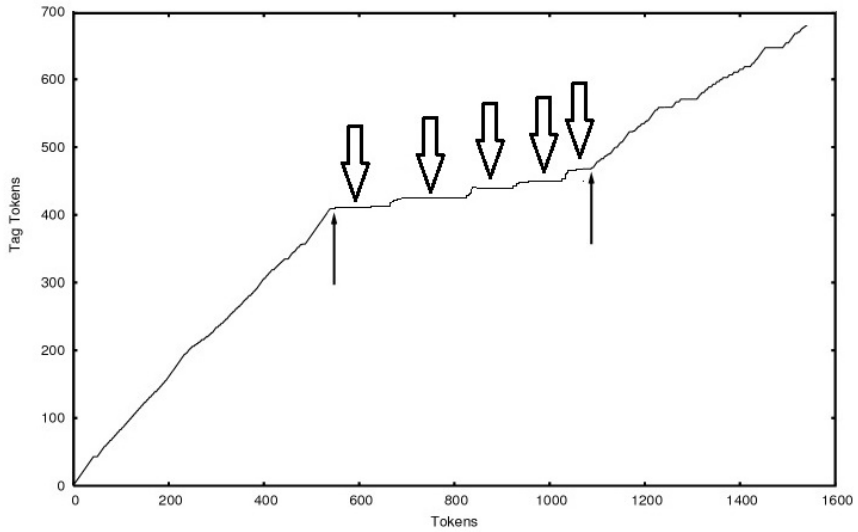


Figure 3.6: Cumulative distribution of tag tokens in a web document. The five continuous plateaus depict the main content in a web document

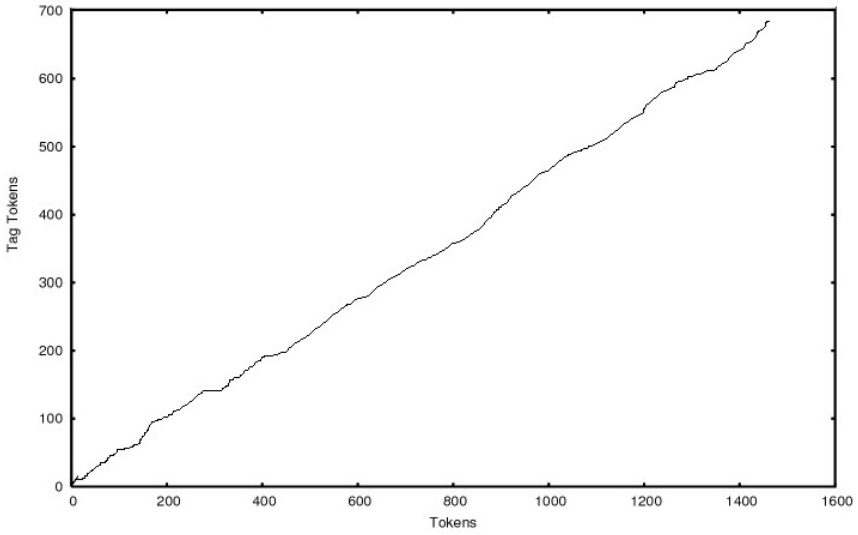


Figure 3.7: An HTML web document without any main content area

considered much larger. However, the maximum size can equal 50 tokens. At this point, take the total average slope of the document as Av. Now, the window defined with a length of L - based on the number of tokens in the document - is moved along on the sloped curve related to the web document but with steps equal to half the length of the window, that is $L/2$. Now, if the average slope of part of the document located inside the window is smaller than half of Av, then this part will be taken as a low slope part. Three continuous parts with low slope characteristics can act as the beginning of a low slope region. The region ends when it

Algorithm 6: Document Slope Curve (49)

Data: B: Token vector of an HTML document of length N, where B_i corresponds to the i-th token and has a value of 1 for tag tokens and 0 for word tokens.
Result: L: Vector of length N denoting whether token i is in a low slope region. L_i is 1 for token in low slope regions, 0 otherwise

begin
- `// Create document slope curve`
- $d[0] \leftarrow B_0$;
- **for** $i = 1 \dots N-1$ **do**
  - $d[i] \leftarrow d[i-1] + B_i$;
- `// Determine window size`
- $w \leftarrow= 8$;
- **if** $N > 5000$ **then**
  - $w \leftarrow= 50$;
- **else if** $N \geq 200$ **then**
  - $w \leftarrow= \lceil 0.00875 \cdot N + 6.25 \rceil$;
- `// Determine low slope regions`
- $s_{total} \leftarrow d[N-1]/N$;
- `// History for last two windows`
- $h \leftarrow [0, 0]$;
- `// Flag if currently in low slope region`
- $lr \leftarrow 0$;
- **for** $i = 0 \dots N-1-w$, *stepwidth* $w/2$ **do**
  - $s_i \leftarrow (d[i+w-1] - d[i])/w$;
  - `// Determine if low slope section`
  - $ls \leftarrow 0$;
  - **if** $s_i < 0.5 \cdot s_{total}$ **then**
    - $ls \leftarrow 1$;
  - `// Check history and update low slope region status`
  - **if** $lr = 0$ **then**
    - **if** $(ls = 1) \wedge (h[0] = 1) \wedge (h[1] = 1)$ **then**
      - $lr \leftarrow 1$;
  - **else**
    - **if** $(ls = 0) \wedge (h[0] = 0) \wedge (h[1] = 0)$ **then**
      - $lr \leftarrow 0$;
  - **for** $j = i \dots i+w-1$ **do**
    - $L[j] \leftarrow lr$;
  - `// Update history`
  - $h \leftarrow [ls, h[0]]$;
- **return** L

reaches three continuous parts having an average slope greater than Av. This region is the same main content under study. Algorithm 6 provides the working method of DSC (49).

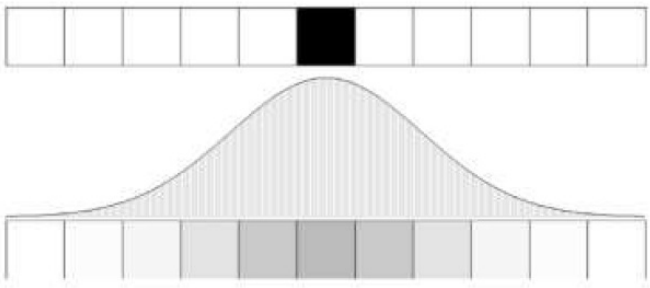


Figure 3.8: The result of repetitious application of the Gaussian distribution function on CCV elements (48)

3.3.1.3 Content Code Blurring (CCB) [2008, Heuristic]

Gottron introduced two algorithms called Content Code Blurring (CCB) and Adapted Content Code Blurring (ACCB) in 2008 (48). The main idea and objective of these two algorithms is to identify and address the regions with high density of content and low density of code in a web document. All HTML tags in the web document are taken as codes by CCB and ACCB, whereas all other things except HTML tags are regarded as content.

These two algorithms tokenize an HTML file and save any character of this file in a one dimensional vector called Content Code Vector (CCV). However, another version of them employs the tokens themselves instead of the characters. Now, if the character saved in an element (of CCV) is taken from an HTML tag, the element under study will be assigned zero; if not, then one. Next, the weighted-average is calculated, taking into account the adjacent elements - r elements on the right and r elements on the left - for each element existing in CCV, i.e. the *i-th* element. The new value is named Content Code Ratio (CCR). It is evident that the amount of CCR for the *i-th* element will be one if this element is equal to one and r elements to the right and left of it are both assigned one. Moreover, if the *i-th* element is assigned zero and r elements on both the right and left sides of it are also zero, the amount of CCR for the *i-th* element will be zero. The third case, which is even more important, would be that the *i-th* element and r elements on its right and left sides take a combination of 0 and 1. In this case, the amount of CCR for the $i-th$ element is in the range of 0-1.

Two algorithms of CCB and ACCB have utilized Gaussian distribution function for calculating the CCR. Therefore, the amount of weighted-average for CCV elements after initialization of the CCV elements would be obtained by the Gaussian distribution function. This process is known as blurring. It is obvious that blurring must be repeated on CCV elements until no change is seen in CCV elements. In other words, the CCV elements should reach their stable values. Figure 3.8 depicts the result of repetitious application of the Gaussian distribution function on CCV elements. The upper section shows the initialization of CCV, while the lower section gives the result of repetitious implementation of blurring on CCV elements. After termination of the blurring, those elements with amounts greater than the user defined threshold will be taken as the main content and extracted. As mentioned earlier, these two algorithms are dependent on parameter r. If this parameter has greater than usual

Algorithm 7: Content Code Blurring (CCB) (49)

Input: D: HTML document with resolved entities, where $D[i]$ is the i-th character, w: weight vector of length $2\delta + 1$, SC: stop criterion, t: threshold, r: replacement character
Output: D' HTML document, where all characters in additional contents are masked by r

```
begin
    // Creating the content code vector
    k ← 1;
    for i = 1 ... |D| do
        if ¬(isComment (D[i]) ∨ isWhiteSpace (D[i]) ∨ isScript (D[i]) ∨ isStyle (D[i]))
        then
            if inTag (D[i]) then
                c[k] ← 0;
            else
                c[k] ← 1;
            k ← k + 1;
    // Iterative blurring
    repeat
        for i = 1 ... |c| do
            c_tmp[i] ← 0;
            for j = −δ ... + δ do
                c_tmp[i] ← c_tmp[i] + c[i + j] · w[j];
        c ← c_tmp;
    until stop criterion SC is met;
    // Extraction process
    k ← 1;
    for i = 1 ... |D| do
        if ¬(isComment (D[i]) ∨ isWhiteSpace (D[i]) ∨ isScript (D[i]) ∨ isStyle (D[i]))
        then
            if inTag (D[i]) then
                D'[i] ← D[i];
            else
                if c[k] > t then
                    D'[i] ← D[i];
                else
                    D'[i] ← r;
            k ← k + 1;
        else
            D'[i] ← D[i];
    return D'
```

values, then the amount of recall will be increased whereas the precision will have smaller values. On the other hand, if the parameter r is initially smaller than usual, the amount of precision will increase but recall will decrease. In ACCB, all anchor-tags are ignored during the creation of the CCV. Two parameters influence the behaviour of these two algorithms; therefore, tuning these two parameters is important in order to produce quality results (50). Algorithm 7 provides the working method of CCB (49).

3.3.1.4 Naive Bayes (NB) Scoring [2009, Hybrid]

Pasternack and Roth (95) introduced a hybrid approach in 2009 which was a combination of heuristic and supervised learning methods, calling it Maximum Subsequence Segmentation or Maximum Substring Segmentation (MSS). They indeed transformed the problem of finding the main content in HTML documents into the problem of Maximum Subsequence Optimization (MSO). MSO assumes that there is a sequence of numbers and the main objective is to find a contiguous subsequence of numbers among this sequence of numbers in which the sum of the elements for this subsequence reaches its maximum. As an example consider the input sequence $(4, -5, 3, -5, 1, 2, -2, 2, -2, 1, 5)$. The maximum scoring subsequence is $(1, 2, -2, 2, -2, 1, 5)$ with a total score of 7. Formula 3.5 can be used to identify this subsequence, wherein a and b are defined as two indexes having values between 1 and n.

$$(a,b) = arg \max_{(x,y)} \sum_{i=x}^{y} s_i \tag{3.5}$$

Ruzzo et al. (104) published their article entitled *"A Linear Time Algorithm for Finding All Maximal Scoring Subsequence"* in 1999 in which an algorithm has been proposed that finds non-overlapping maximal subsequence in a linear time. The time complexity of this algorithm for recovering an ordered list includes K highest-valued subsequences from one sequence of n elements equal to $O(n * log(K))$. The Algorithm 8 provides a simple and single pass algorithm for finding the maximum subsequence in a linear time (62). Furthermore, Figure 3.9 illustrates an overall flowchart of NB Scoring (30).

Algorithm 8: Finding the Maximum Subsequence in a Linear Time

Input: $S : S = (s_1, s_2, ..., s_n), s_i \in R$
Output: $maxSS : maxSS = (s_i, s_{i+1}, ..., s_j), 1 \leq i \leq j \leq n$
begin
  $start \leftarrow 1$;
  $sum \leftarrow 0$;
  $maxSS \leftarrow (-\infty)$;
  for $i = 1$ *to* n **do**
    $sum \leftarrow sum + s_i$;
    if $sum > value(maxSS)$ **then**
      $maxSS \leftarrow (s_{start}, s_{start+1}, ..., s_i)$;
    if $sum < 0$ **then**
      $start \leftarrow i + 1$;
      $sum \leftarrow 0$;
  return $maxSS$

Within MSS, each web document is tokenized and displayed based on the following steps:

- removing everything between `<script>` and `<style>` tags, since scripts never contain main content
- break up the HTML document into a list of tags, words and numbers
- apply porter stemming (98) to all words
- generalize numeric tokens, all numbers stemmed to "1"

In order to apply MSO on the obtained tokens, MSS uses the local token-level classifier to produce a score for each token of the web document. A negative score implies that the observed token does not tend to be considered as a content token, while a positive token tends to be taken as a content token. Since the MSS uses a global optimization overall score, there is no need to have highly accurate local classifiers. But in order to calculate the score of each token, the Naive Bayes method with two types of features has been adopted for each labeled token in the web document:

- trigram of token: the token itself and its 2 successors
- parent tag of token in the DOM tree

Having calculated the score p by the NB classifier, the value obtained by $f(p)=p-0.5$ is transformed into a new value to yield a sequence of scores located in the range of [-0.5, 0.5]. Then, the application of MSO on the sequence of scores produced by the local token-level classifier would lead to segments having the maximum subsequence. The indexes of this subsequence are indicative of the starting point and ending point of a set of tokens which form the main content. It is important to note here that both phases of learning and prediction in the NB Scoring have a linear time and the obtained results are acceptable. In other words, this semi-supervised algorithm yields an overall F1-measure of 97.94%. One of the problems of semi-supervised algorithms is that they would need to tune their tunable parameters upon application of the algorithm on a new domain. NB Scoring is also not excluded from this rule.

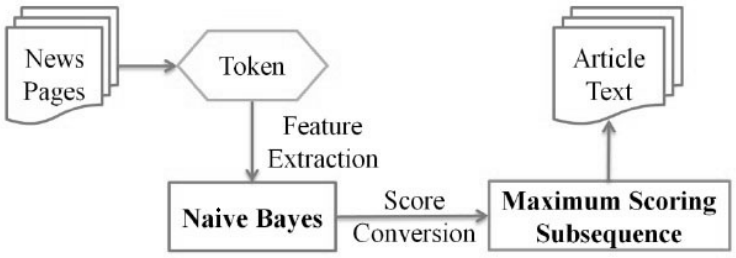


Figure 3.9: The Flowchart of NB Scoring (30)

3.3.2 Block-based

Block-based main content extraction algorithms, divide an HTML file into a number of blocks, and then look for those blocks which contain the main content. Therefore, the output of these algorithms is comprised of some blocks which probably contain the main content.

3.3.2.1 Boilerplate Detection using Shallow Text Features [2010, Machine Learning]

Kohlschütter et al. (69) introduced "Boilerplate Detection using Shallow Text Features" in 2010. The main idea of the proposed algorithm is to classify the individual text elements, at text block level, into the main content and boilerplate text in an HTML web document based on the analyzing the following structural and shallow text features:

- Analyzing the most popular set of shallow text features, i.e. average word length, average sentence length, and the absolute number of words.
- Examining a few heuristic features such as: 1) the absolute number of words that either start with an uppercase letter or are completely upper-case, 2)the ratio of two different kinds of words, i.e. words starting with an uppercase letter or words comprising of completely upper-case, compared to the total number of words, 3) the ratio of full stops to the overall number of words, the number of date/time-related tokens and the number of vertical bars "|".
- Examining the structural features such as: the presence of a particular headline tag (`<H1>, <H2>, <H3>, <H4>, <H5>, <H6>`), a paragraph tag `<P>`, a division tag `<DIV>`.
- Computing the link density: the number of tokens within an `<a>` tag divided by the total number of tokens in the block; for this computation, the `<a>` tag does not regard as a block separator.
- Identifying the local context, i.e. the absolute and relative position of a text block in an HTML document.

Kohlschütter concluded that a combination of two features such as "number of words" and "link density" leads to a simple classification model that achieves competitive accuracy.

The overall approach which is used in this algorithm is simple and can be explained as below.

- First of all, collected web pages are segmented into atomic text blocks. Atomic text blocks are those blocks which are separated by one or more HTML tags, except for `<a>` tags. Then, all atomic text blocks are annotated with structural and shallow text features and on this basis classified into main content or boilerplate text using decision trees method and linear support vector machines.
- The evaluation is performed on two datasets, a news collection for training and testing and the CleanEval collection for validation. The first dataset consists of 621 manually assessed news articles from 408 different web sites. The second dataset consists of 798 raw HTML pages randomly sampled from Web search engines.

3.3.2.2 Content Extraction via Sequence Labeling [2007, Machine Learning]

Gibson et al. (43) used *"Statistical Sequence Labeling Models" (SSLM)* to identify the main content in a news web page. To the best of our knowledge, SSLM has numerous applications in natural language processing (NLP) for performing various tasks including part-of-speech tagging and noun-phrase chunking. Gibson deems main content identification from a web page as a sequence labeling problem, in which the sequence elements are blocks in an HTML file. These blocks are indeed a set of tokens placed in HTML tags (but not in all tags). Then, he used three of the following sequence labeling models to make main content identification possible and it should be noted that the CRF method provides much better results than other methods.

- Conditional Random Field (CRF) (72)
- Maximum Entropy Classifier (MaxEnt)
- Maximum-Entropy Markov Model (MEHH)

By using each of above methods, the existing blocks in a web document are then categorized to content or non-content.

3.3.3 Line-based

In this section we explain line-based algorithms which consider each HTML file as a continuous sequence of lines. Taking into account the applied logic, they introduce those lines of the file which are expected to contain the main content. Then, the main content is extracted and provided to the user from the selected lines.

3.3.3.1 Content Extraction via Tag Ratio (CETR) [2008, Heuristic]

Weninger et al. (124) introduced content extraction via Text-to-Tag Ratios (TTR), called CETR, which is partially based on previous work in web content extraction (123). This method extracts the main content from web pages by using the HTML document's Text-to-Tag ratio. Their method computes the ratio of the number of non-tag characters to the number of tags characters per line and stores these values in a one-dimensional array named T and, afterwards, produces a two-dimensional histogram based on the results (see Figure 3.10). This histogram demonstrated that the lines 220 to 260 in the HTML file are those with rather high Text-to-Tag Ratios and thus could be considered as the main content.

To elucidate this issue, a brief snippet from the news web site of Hutchinson has been provided in Example 3.11 with the method of calculating TR.

The core of the CETR algorithm is described in Algorithm 9. Prior to calculation of TRs, all tags related to script, remark and style are removed from the body of the HTML file since these codes are accounted for non-tag text and this can make the CETR algorithm unable to provide the user with the main content.

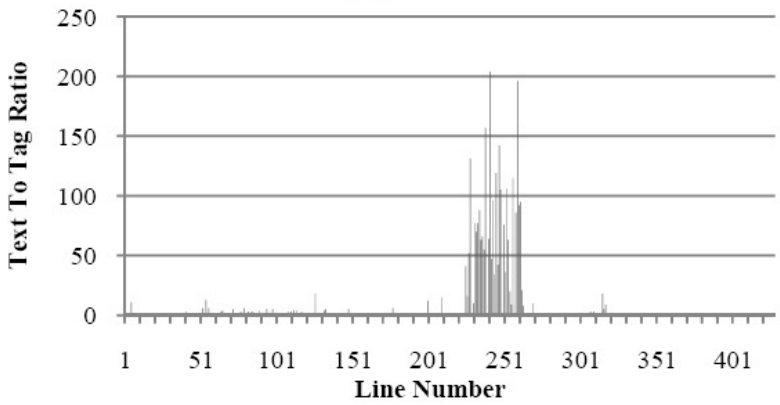


Figure 3.10: Text-to-tag ratio for a web page from the Hutchinson News (124)

```
1. <div id="topnav">
2.   <div id="storyPageContent2">
3.   <div id="author">James Smith</div>
4.   OKLAHOMA CITY - Police were told that...
5.   ...The Oklahoman reported Sunday. <br><br> Jones had...
6. </div></div>
```

The Tag Ratios for these six lines are computed as follows:

```
1. Text=0, Tags=1, TR=0
2. Text=0, Tags=1, TR=0
3. Text=11, Tags=2, TR=5.5
4. Text=37, Tags=0, TR=37
5. Text=41, Tags=2, TR=20.5
6. Text=0, Tags=2, TR=0
```

Figure 3.11: A brief snippet of a webpage news article

Finally, by using the k-means clustering method originally proposed by MacQueen (77), the resulting histogram is clustered into the content and the non-content area. Since CETR is an heuristic method and benefits from the K-means technique to cluster both content and non-content areas, it no longer requires training. It is interesting to note that CETR is one of the best MCE methods and its performance can be claimed as good. Another point which must be pointed out here is that CETR uses a Gaussian smoothing function before clustering the lines of an HTML document to content or non-content. This function can prevent removing short paragraph lines which might be a part of the main content, such as the page title.

The overall flowchart of CETR is shown in Figure 3.12.

3.3.3.2 Density [2009, Heuristic]

Moreno et al. (91) introduced a language independent algorithm called Density (tested on English, Italian and German languages) for the main content extraction. This approach has

Algorithm 9: CETR Pseudocode (123)

```
Input: D: Document
Output: T: Tag ratios for all lines
begin
    D ← removeScriptTags(D) ;
    D ← removeRemarkTags(D) ;
    D ← removeStyleTags(D) ;
    for i = 1 to |D| do
        x ← nonTagChars(D_i) ;
        y ← tags(D_i) ;
        if y = 0 then
            y ← 1 ;
        T_i ← x/y ;
```

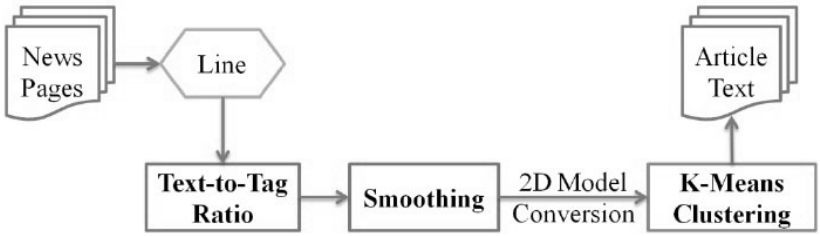


Figure 3.12: The Flowchart of CETR (30)

two phases. In the first step, texts are separated from the HTML tags by using an HTML parser (6); afterwards, the extracted texts are saved in an array of strings L. A graphical representation of the array L is depicted in Figure 3.13 in which the x-axis represents the position of the array and the y-axis represents the length of the strings at the different positions.

In the second step, a region in the array L that has the highest density will be determined as a main content. In addition to finding the highest density area in the array L, two parameters influence the behaviour of the algorithm. The first parameter, $C1$, determines the minimum required length for texts in each element of the array L to be selected and inserted into the new array of String R, which is considered to keep the high density region of text. The second parameter, $C2$, specifies the acceptable distance between lines in R and the lines which want to be added to R.

3.4 Content Extraction Systems

A number of algorithms which are implemented to extract the main content from web pages were studied in sections 3.2 and 3.3. Moreover, several frameworks and systems have been designed and implemented for main content extraction from web pages, two examples of which are explored here.

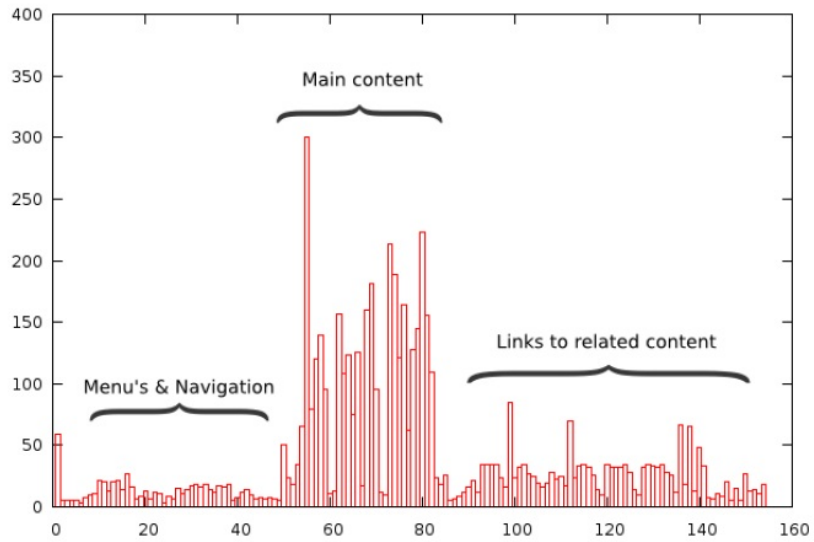


Figure 3.13: Example plot of the document density (91)

3.4.1 Crunch Framework

One of the more prominent solutions for MCE is the Crunch framework of Gupta et al. This framework applies an HTML parser to construct a DOM tree from an HTML document. Then, by navigating the DOM tree recursively, rather than using the raw HTML markup, and utilizing a number of heuristic filtering techniques, the main content of HTML web pages is extracted.

3.4.2 CombinE System [2008, 2009]

CombinE system was first introduced by Thomas Gottron in 2008 (47). The main purpose of this system was to combine the content extraction heuristic algorithms with each other in order to extract the main content from web pages with much more accuracy. He has implemented an http-proxy server which is able to filter web pages on-the-fly. This system also has the ability to configure and evaluate various combinations of CE heuristic algorithms automatically. The *CombinE* system is indeed an extensible collection of content extraction filter modules with a standardized interface. These modules are the original building blocks of filter pipelines. In other words, each filter pipeline is a combination of an arbitrary number of content extraction filter modules. Figure 3.14 depicts the outline of a *CombinE* system.

It should be noted that filter modules can be combined with each other in various forms to build a filter pipeline. Three filter pipelines are considered in *CombinE*, namely: serial, parallel and voting. In serial filter pipelines, content extraction algorithms are executed based on a predefined order and deliver the output of one filter to the next filter. Several content extraction algorithms are selected in parallel filter pipelines, and then each of them is executed on a copy of the original version of the web page. Afterwards, the obtained results are verified or intersected with each other to finally give the main content. In the third form,

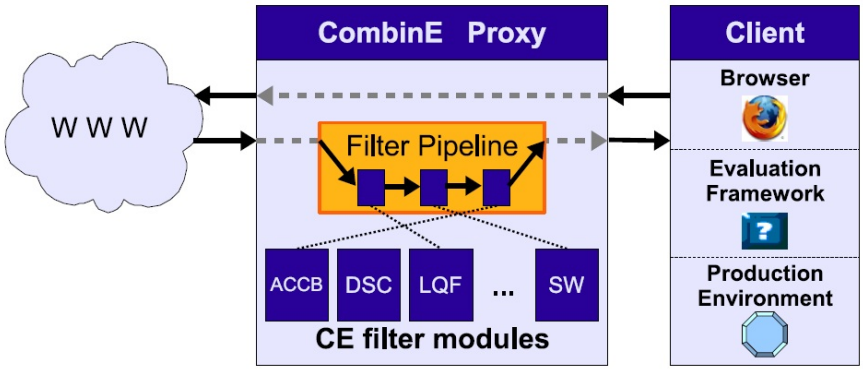


Figure 3.14: Outline of the *CombinE* system

i.e. voting, which is a special case of parallel filter pipeline, a feature has been devised for the content extraction algorithm to vote those parts of web documents which can potentially contain main content.

When a content fragment has received sufficient votes from the filters, it can be introduced as the main content. Various filters were built and executed based on the predefined data sets in the second version of the *CombinE* system (122) to evaluate different filters. However, it should be pointed that the heuristic algorithms which were used by *CombinE* system were: DSC, BTE, ACCB and LQF.

3.5 Comparable Platform for Boilerplate Removal

CleanEval was introduced in 2008 by Baroni et. al (81). The main goal of this project was to establish representative web data, with a gold standard, for use as a corpus for linguistic and language technology research and development. It is obvious that if a web corpus with uncleaned data is fed to layers of linguistic technology, then the most significant bigrams will often be "Click here" or "Further information" and consequently the language model will distorted considerably. So, the organizers of the CleanEval decided to invite scholars to take part in an open competitive evaluation platform (to identify good cleaning algorithms and to foster sharing of ideas and programs) on the topic of cleaning arbitrary web pages.

The evaluated algorithms mainly apply machine learning techniques for the classification. So, similar to all other machine learning algorithms, in CleanEval 57 and 60 web pages in English and Chinese languages have been collected and annotated as development set to be employed in learning phase of machine learning algorithms. In addition, 684 and 653 web pages in English and Chinese languages have been prepared as evaluation data that can be used for measuring the accuracy of the contributed content extraction algorithms. NCleaner (35) and Victor (111) are two machine learning techniques which have been contributed to this competitive evaluation platform.

In this project, all files in the development set were annotated by two annotators. The annotators were instructed to remove all boilerplate text and then add a basic encoding of

the structure of the page using a minimal set of symbols(e.g. p, h, l) to make the beginning of header, paragraphs and list elements.

Chapter 4

Algorithms: R2L, DANA, DANAg, and AdDANAg

This chapter is the main part of this thesis and we will explain all four invented main content extraction methods, namely R2L, DANA, DANAg, and AdDANAg. Based on the observations regarding character encoding in Section 2.4, we develop our R2L (89) algorithm in Section 4.3. With its extensions DANA (85) and DANAg (84) in Sections 4.4 and 4.5, we further enhance and generalize the original idea towards a better performance and a language-independent version. Finally at the end of this chapter, we introduce AdDANAg which is an adaptive version of DANAg and it is able to extract the main content of hyperlink rich web documents. Altogether, in this chapter after introducing data sets in Section 4.1 and evaluation methodology in Section 4.2, we will make three main contributions:

- We develop the idea of using character encoding for developing R2L, a new approach for main content extraction.
- We extend the R2L approach to the algorithms DANA, DANAg, and AdDANAg to further improve the extraction accuracy and develop a language independent version of the method.
- We analyse our approaches under the aspects of efficiency and effectiveness. We compare them to eleven established MCE algorithms (41, 48, 55, 80, 91, 96) and show that we extend the state-of-the-art in terms of both efficiency and effectiveness.

In Chapter 3, we categorized all main content extraction algorithms into two categories as follows:

- MCE algorithms based on the DOM tree structure
- MCE algorithms based on the HTML source code elements

Our presented algorithms are categorized in the second group because we employ only HTML tags to implement our methods. In addition, our approaches can be classified into line-based main content extraction methods which have been described in Section 3.3.3.

Table 4.1: Evaluation corpus of 2,166 web pages

web site	URL	Size	Languages
BBC	http://www.bbc.co.uk/persian/	598	Farsi
Hamshahri	http://hamshahrionline.ir/	375	Farsi
Jame Jam	http://www.jamejamonline.ir/	136	Farsi
Al Ahram	http://www.ahram.org/	188	Arabic
Reuters	http://ara.reuters.com/	116	Arabic
Embassy of Germany, Iran	http://www.teheran.diplo.de/Vertretung/teheran/fa/Startseite.html	31	Farsi
BBC	http://www.bbc.co.uk/urdu/	234	Urdu
BBC	http://www.bbc.co.uk/pashto/	203	Pashto
BBC	http://www.bbc.co.uk/arabic/	252	Arabic
Wiki	http://fa.wikipedia.org/	33	Farsi

Table 4.2: Evaluation corpus of 9,101 web pages

web site	URL	Size	Languages
BBC	http://www.bbc.co.uk/	1,000	English
Economist	http://www.economist.com/	250	English
Golem	http://golem.de/	1,000	German
Heise	http://www.heise.de/	1,000	German
Manual	several	65	German, English
Republica	http://wwwrepublica.it/	1,000	Italian
Slashdot	http://slashdot.org/	364	English
Spiegel	http://www.spiegel.de/	1,000	German
Telepolis	http://www.telepolis.de/	1,000	German
Wiki	http://fa.wikipedia.org/	1,000	English
Yahoo	http://news.yahoo.com/	1,000	English
Zdf	http://www.heute.de/	422	German

4.1 Data sets

For the evaluation of our algorithms proposed in this chapter, we use two different corpora. The first corpus contains 2,166 web documents in Arabic, Farsi, Pashto, and Urdu (see Table 4.1). This data set has been proposed in (89) and has been collected for the evaluation of main content extraction algorithms on Right-to-Left language web pages. The second corpus contains 9,101 web pages from different web sites (see Table 4.2). This data set has been introduced in (48) and has been established for evaluation of main content extraction approaches on Western language documents. To evaluate R2L and DANA algorithms, we use only the first corpus. On the other hand, for evaluation of DANAg and AdDANAg approaches both data sets will be used.

4.2 Evaluation Methodology

As explained in detail in Section 2.3.2, in order to calculate the accuracy of any main content extraction method, it is necessary to provide a manually crafted *gold standard* for the main content of all HTML files. Both corpora provide such a gold standard. For the purpose of evaluation, the output of a main content extraction algorithm is compared with the gold standard of the corresponding HTML document. For comparing the *gold standard file* with the produced *cleaned file*, it is essential to compute an overlap between the two of them. The establish method, introduced in (46) and used throughout several papers on main content extraction (48, 50, 91), is to use the Longest Common Subsequence (LCS) (62) to find this overlap between the gold standard and the cleaned file. Now by counting the number of tokens of gold standard and cleaned files, g and m respectively, and the number k of tokens returned by the LCS function, we can evaluate the accuracy of the main content extraction algorithm by applying the classical information retrieval performance measures – Recall, Precision, and F1-measure (46), as defined again here in formula 4.1:

$$\mathrm{r} = \frac{\mathrm{length(k)}}{\mathrm{length(g)}}, \quad \mathrm{p} = \frac{\mathrm{length(k)}}{\mathrm{length(m)}}, \quad \mathrm{F1} = 2 * \frac{\mathrm{p} * \mathrm{r}}{\mathrm{p} + \mathrm{r}} \tag{4.1}$$

4.3 Algorithm R2L

In the early days of the World Wide web, the content of most of the web pages was written in the English language. By now, and especially in the last decade, a great deal of information is also being published in other languages, for example in Spanish, German, French, etc. Except for the non-English languages mentioned here, there are several other languages using non-ASCII codes for their characters (Figure 4.1 gives an example of web pages with a non-ASCII character set in which we have also highlighted the main content). The Unicode character set (UCS), which was introduced after ASCII and ISO-8859*, reserves an exact interval for each language. Some of these intervals have no common character with the English character set.

The R2L (89) approach presented in this chapter exploits this fact to realize an MCE algorithm for the Arabic, Farsi, Pashto, and Urdu languages. By working on the binary character encoding directly, we achieve an improvement in time performance. Moreover, our approach also outperforms all other MCE algorithms in extraction performance, i.e. detects the main content more accurately and reliably. This provided the motivation for the initial version, R2L, of our algorithms presented in this chapter.

Figure 4.2 shows our R2L system architecture. A user may submit an HTML document to the R2L system as the initial input. The results returned by R2L are text files including main content of corresponding HTML files. The process underlying R2L system can be subdivided into a preprocessing step and four main phases. The individual steps in this process work as follows.

Figure 4.1: A web page with an outlined main content (dotted lines are drawn manually)

4.3.1 Preprocessing step

In the preprocessing step, all JavaScript and CSS codes and comments are removed from the HTML file(see Figure 4.2). There are two reasons for this: (a) they do not directly contribute to the main text content and (b) they do not necessarily affect the content of the HTML document at the same position where they are located in the source code. Particularly this latter incoherence between presentation and technical realisation in the source code could introduce inconsistencies in the downstream analysis[1]. In addition in this step, we normalize line length and, thereby, render the approach independent from the actual line format of the source code.

4.3.2 First Phase: Character Set Separation

In the following, we define the two sets, $S1$ and $S2$, which we will use throughout this section:

$$S1 = \{\text{All characters belonging to UCS R2L languages}\}$$
$$S2 = \{\text{All first 128 characters of UCS}\}$$

We know that the English characters, which are used in HTML tags, have values less than 128 and therefore can be classified to $S2$. All characters of R2L languages use two bytes

[1]This effect has already been observed in related work (48).

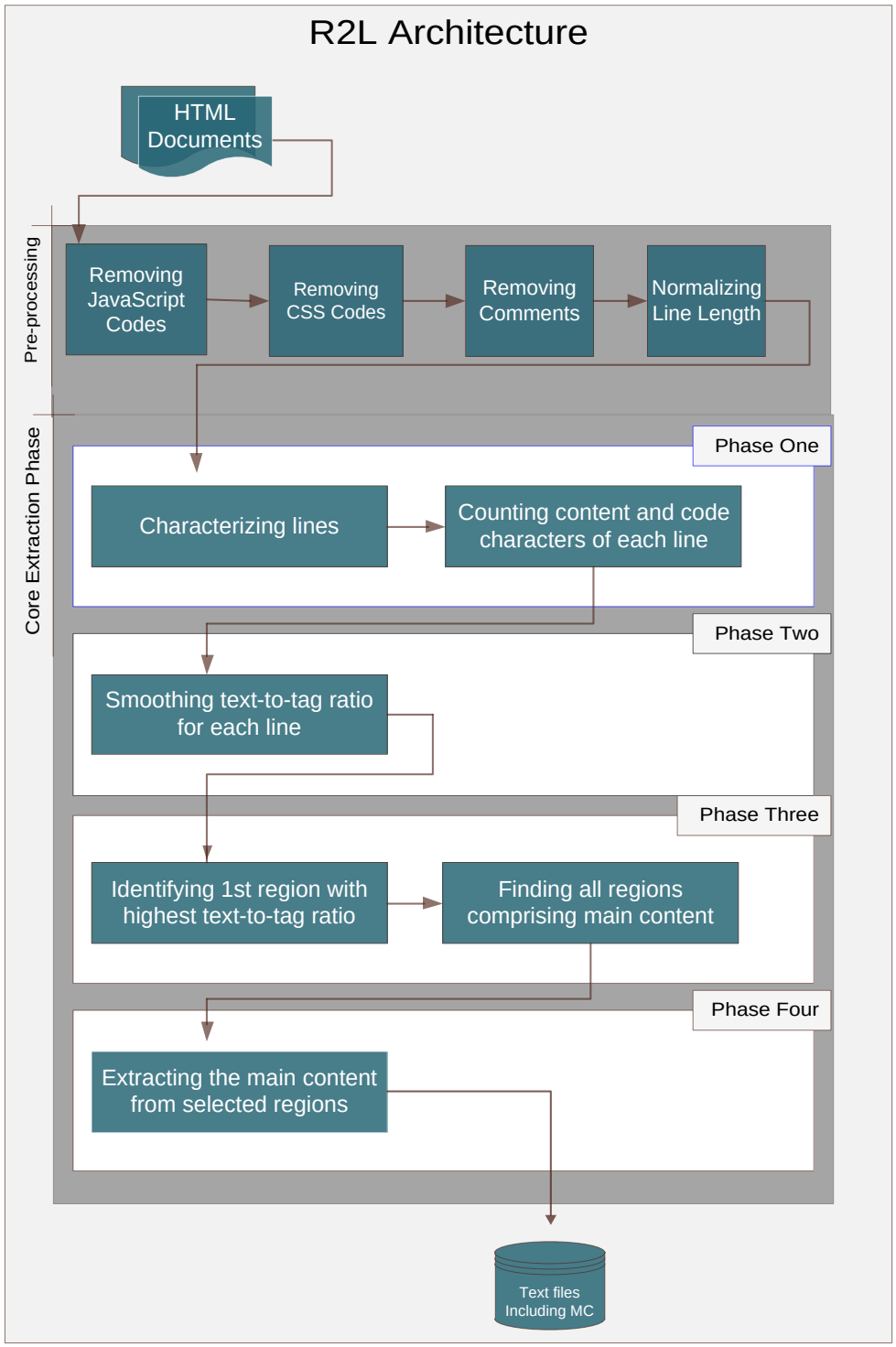


Figure 4.2: R2L main content extraction system architecture

with a value greater than 127, and therefore they are classified to $S1$. This simple rule helps us to efficiently separate R2L language characters from the first 128 characters of UCS.

In this first phase, the algorithm reads an HTML file as a stream of bytes and then by using the above rule, it distinguishes whether the generated byte is a member of $S1$ or $S2$. Now, the characters in each line of the file are separated into two parts: characters that are a member of $S1$, called content characters, and the ones that are a member of $S2$, named code characters. With regard to this condition, we are able to count the number of characters belonging to $S1$ and $S2$ for each line of an HTML file, which is stored in two one-dimensional arrays $T1$ and $T2$, respectively.

4.3.3 Second Phase: Smoothing

After storing the number of R2L and English characters for each line of the HTML file in the two arrays $T1$ and $T2$, we want to recognize areas in the HTML file in which the density of R2L and English characters is high and low, respectively.

To illustrate our approach, we depict two diagrams. In Figure 4.3, we draw two groups of columns above and below the x-axis, with the length equal to the number of R2L and English characters, as stored in $T1$ and $T2$, for each line of the HTML file. For example, suppose that the i-th line of an HTML file has y_1 R2L and y_2 English characters. Then, two lines with the length equal to y_1 and y_2 are drawn above and below the x-axis. Our hypothesis is that the main content is typically located above the x-axis. In Figure 4.3, the measurement unit for the x-axis is the number of lines in the HTML file and the measurement unit for the y-axis, above and below the x-axis, is the number of R2L and English characters, respectively, of each line of an HTML file. Here we interpret Figure 4.3 to find the MC of an HTML file. There are three types of regions:

- Regions that have a low or near zero density of columns above the x-axis while having a high density of columns below the x-axis. We observed that these regions typically consist mainly of HTML tags. We outline examples for such areas with A in Figure 4.3.
- Further, we see some regions which have a high density of columns above the x-axis and low density of columns below the x-axis, one of them marked with B in Figure 4.3. The main content, typically, will be found areas like this. In other words, some of these areas comprise the main content.
- There are some regions that have a medium density of columns above and below the x-axis. These regions form parts of navigation menus, panels, or other related link lists. Here, normally, the density of the columns below the x-axis is somewhat more than the density of the columns above the x-axis because in HTML files we need to write many tags to make menus or extraneous items. One of these areas is outlined in Figure 4.3 and labeled with C.

Now the problem of finding MC in the HTML web pages becomes the problem of finding regions such as region B in Figure 4.3 comprising the main content. In the next two steps and in phase three (Section 4.3.4), we apply an elegant and simple method for finding regions such as B containing the main content in an HTML file:

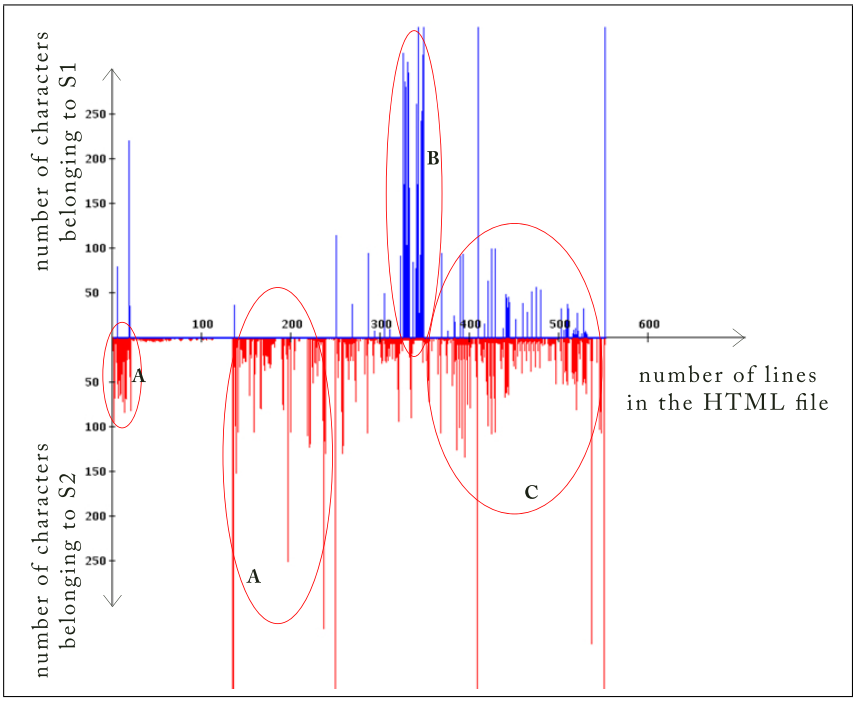


Figure 4.3: An example plot shows the density of the main content and extraneous items

- For all lines i we calculate $diff_i$ by using Formula 4.2. In this formula, $T1_{\text{i}}$ and $T2_{\text{i}}$ are the number of R2L and English characters of line i in an HTML file.

$$diff_i = (T1_{\text{i-1}} - T2_{\text{i-1}}) + (T1_{\text{i}} - T2_{\text{i}}) + (T1_{\text{i+1}} - T2_{\text{i+1}}) \tag{4.2}$$

  This produces a smoothed plot as can be seen in Figure 4.4. Here again, if $diff_i > 0$ we draw a line with length $diff_i$ above the x-axis. Otherwise, we draw a line with length $|diff_i|$ below the x-axis. Unlike Figure 4.3, a large part of the menus and additional news in Figure 4.4 have been hidden.

- Now in Figure 4.4, we identify all regions above the x-axis and, for simplicity, we define a new set $R = \{r_1, r_2, ..., r_n\}$ of all such regions. Each element r_j denotes only one individual line or a range of lines covering one region (see Formula 4.3) and n is the total number of recognized regions above the x-axis. In our example, there are several regions, one near the y-axis, two regions in the middle of the x-axis, and finally some small regions in the interval [450, 500] of the x-axis. In addition, we count the number of characters for each region and also we specify the position of regions in Cartesian coordinate. The strong hypothesis underlying R2L is that among all regions, the region with the maximum number of characters definitely belongs to the main content.

$$r_j = [x_j, y_j], \quad x_j, y_j \in N, \quad x_j \leq y_j \tag{4.3}$$

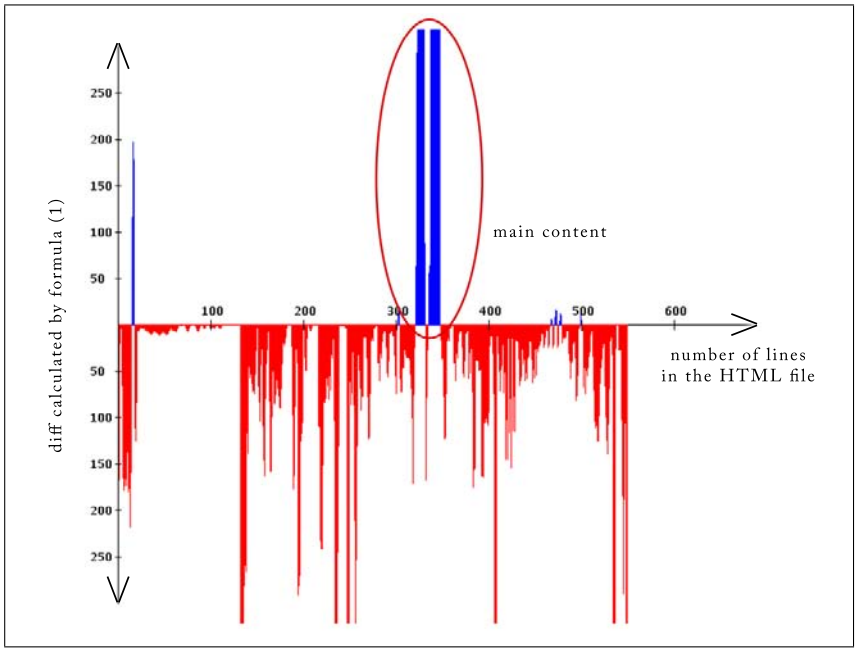


Figure 4.4: Smoothed version of Figure 4.3, in which the MC can be identified more easily

4.3.4 Third Phase: Recognizing the Boundary of the Main Content Area

In this phase, all regions shaping the main content are discovered. Concerning the set $R = \{r_1, r_2, ..., r_n\}$ defined at the end of the previous phase, we have two possible outcomes: (1) we discovered only one region, i.e. $n = 1$, or (2) we have discovered several regions, so $n > 1$. If $n = 1$, then r_1 is the only main content region and this procedure is finished. Otherwise, if $n > 1$, we start by finding the region $r_m \in R$ which contains the highest number of characters. Again, we define a new empty set T, intended to denote the set of all regions comprising the main content at the end of this phase, and add r_m to this set T. For finding all other regions of the main content, we use Algorithm 1. In this algorithm, $d(r_i, r_j)$ returns the distance between two regions r_i and r_j (see Formula 4.4) and the parameter *gap* determines the maximum allowed distance between two sequential regions of main content.

$$\begin{aligned} r_i &= [x_i, y_i], \quad x_i, y_i \in N, \ x_i \leq y_i \\ r_j &= [x_j, y_j], \quad x_j, y_j \in N, \ x_j \leq y_j \\ y_i &< x_j \\ d(r_i, r_j) &= x_j - y_i + 1 \end{aligned} \tag{4.4}$$

In Algorithm 1, the first loop discovers all regions on the left side of r_m comprising the main content. For example in the first iteration of *while*, Algorithm 1 evaluates the distance

between sequential regions r_m and r_{m-1} and if this distance is less than or equal to *gap*, then r_{m-1} is added to the set T. Otherwise, the *while* loop is terminated. Therefore, the *while* loop will be terminated immediately as soon as the distance between two consecutive regions becomes greater than *gap*. In the same way, the second loop distinguishes all regions to the right side of r_m comprising the main content and adds the valid regions to the set T. The result and output of Algorithm 1 is the set T comprising all regions making up the main content of the selected web page. It is clear that T is a subset of R.

Algorithm 1 Finding All Regions Comprising MC

1: $T = \{r_m\}, R = \{r_1, r_2, ..., r_n\}, 1 \leq m \leq n$
2: $i = m$
3: **while** $((i > 1) \quad AND \quad (d(r_i, r_{i-1}) \leq gap))$ **do**
4: $\quad T = T \cup \{r_{i-1}\}$
5: $\quad i - -;$
6: **end while**
7: $i = m$
8: **while** $((i < n) \quad AND \quad (d(r_i, r_{i+1}) \leq gap))$ **do**
9: $\quad T = T \cup \{r_{i+1}\}$
10: $\quad i + +;$
11: **end while**
12: return T

Obviously, the parameter *gap* has an influence on the accuracy of R2L. We used a small set of test pages to empirically find a well performing default value for *gap*. For R2L, a value of $gap = 8$ has proven to demonstrate good results.

4.3.5 Fourth Phase: Extracting the Main Content from Selected Regions

In this phase, all Right-to-Left characters of the areas recognized in phase three are separated from all characters belonging to $S2$ and then are considered as the final output of the R2L algorithm. Effectively, the output then contains the MC.

4.3.6 Results

Table 4.3 shows the F1 scores of R2L algorithm on the corpus composed of Right-to-Left language web documents. Column 3 shows F1 scores when considering the parameter *gap* set to a default value of 8. We can see that the achieved accuracy of R2L is very high in general. For many web sites, such as BBC, Hamshahri, and Jame Jam, it achieves nearly a perfect F1 score very close to 1 and it has a very good F1 score for most other web pages.

Additionally, we investigated the theoretical upper bound for MCE using R2L. Columns 4 and 5 show a theoretical optimal setting for *gap*, which provides the best result that can be achieved with R2L. For example, on the Al Ahram data set, the optimal value for the *gap* parameter would be 7. In this case, we achieve an F1 score of 0.983, which is a significant improvement over the baseline of the fixed *gap* parameter. However, in most cases the optimum value for *gap* is not far from 8 and the best theoretical F1 score does not diverge much from the performance of R2L.

Table 4.3: The Average F1 Scores of R2L based on Table 4.1

web site	Languages	F1 with $gap = 8$	Optimal value for *gap*	F1 with optimal *gap* in Col. 4
BBC	Farsi	0.991	8	0.991
Hamshahri	Farsi	0.991	8	0.991
Jame Jam	Farsi	0.977	3	0.987
Al Ahram	Arabic	0.929	7	0.983
Reuters	Arabic	0.936	4	0.971
Embassy of Germany, Iran	Farsi	0.954	15	0.971
BBC	Urdu	0.956	11	0.997
BBC	Pashto	0.974	8	0.974
BBC	Arabic	0.987	8	0.987
Wiki	Farsi	0.283	16	0.385

4.4 Algorithm DANA

Here, we introduce the first extension of R2L, DANA (85) to improve the effectiveness of R2L. Since the R2L approach determines its output only from the Right-to-Left character set of the identified main content areas of web pages, it might miss some fractions of the MC. This happens when there are some English words or characters in the main content areas of a web page. As the R2L algorithm is incapable of keeping these English words in the extracted main content, the recall score of R2L algorithm will not be optimal in these cases. This conceptual drawback is overcome by DANA.

DANA is divided into one preprocessing step and four phases. Empiric evaluation on our small set of test pages show that for DANA, the best value for the parameter *gap* is 20, so all results produced by DANA are based on a value of $gap = 20$. The preprocessing step as well as phases one, two, and three are equivalent to the R2L approach. Thus, below we explain only the differences in phase four.

4.4.1 Extracting the Main Content from Selected Regions Using a Parser

In this phase of DANA, we feed only those HTML lines determined in the third phase of R2L as an input to an error-tolerant parser (48). Following our hypothesis, the output of the parser is more accurate than the output of the phase four of the R2L approach, so DANA achieves overall extraction performance better than R2L. On the downside, applying the parser to selected fragments of a document causes an overhead in computation, so R2L achieves an overall better time performance than DANA. Concerning these two facts, we will see that the trade-off between efficiency against effectiveness is worth the runtime overhead.

4.4.2 Results

Tables 4.4, 4.5, and 4.6 give statistics showing the recall, the precision and the average F1 scores of DANA and 11 other algorithms on 2,166 selected web pages from 10 different web sites based on a *gap* parameter setting of 20.

Table 4.4: Recall of DANA based on the corpus in Table 4.1

	Al Ahram	BBC Arabic	BBC Pashto	BBC Persian	BBC Urdu	Embassy	Hamshahri	Jame Jam	Reuters	Wikipedia
ACCB-40	0.837	0.911	0.874	0.941	0.939	0.758	0.929	0.802	0.913	0.641
BTE	0.930	0.997	0.945	0.999	0.998	0.980	0.969	0.970	0.999	0.926
DSC	0.877	0.902	0.812	0.947	0.833	0.782	0.933	0.888	0.848	0.687
FE	0.743	0.033	0.099	0.033	0.001	0.009	0.137	0.016	0.147	0.145
KFE	0.901	0.882	0.830	0.807	0.647	0.757	0.643	0.844	0.742	0.597
LQF-25	0.931	0.997	0.951	0.998	0.996	0.974	0.963	0.968	1.0	0.744
LQF-50	0.931	0.997	0.951	0.998	0.996	0.977	0.968	0.968	1.0	0.853
LQF-75	0.931	0.997	0.951	0.998	0.996	0.978	0.968	0.968	1.0	0.890
TCCB-18	0.870	0.921	0.888	0.962	0.982	0.845	0.936	0.888	0.933	0.776
TCCB-25	0.853	0.911	0.883	0.960	0.990	0.840	0.934	0.880	0.931	0.768
Density	0.843	0.183	0.883	0.724	0.932	0.845	0.893	0.859	0.981	0.566
DANA	0.995	0.963	0.938	0.995	1.0	0.930	0.980	0.932	0.974	0.573

In Table 4.6, the bold values show the highest F1 score and the italic numbers represent the highest F1 score among all algorithms except DANA. In addition, in Table 4.7 we compute the processing performance in terms of data throughput (MB/s) of DANA and other methods. By looking to these two tables, the following important points can be noticed:

- As can be seen from six web sites, Al Ahram, BBC Arabic, BBC Persian, BBC Urdu, Hamshahri, and Reuters, DANA achieves an F1 score higher than 0.95 and especially on BBC Urdu with an F1 score of exactly 1. No other method shows such a high effectiveness.

- In Table 4.6, only BTE on Wikipedia web documents achieves an F1 score greater than DANA. Wikipedia documents have already been observed to be very difficult for MCE algorithms in other papers (48). By looking inside the Wikipedia HTML file, we discover that there are big gaps, more than 20, between the regions composing the main content. Looking at DANA's recall of 0.5734, it can be seen that it erroneously discards large parts of the main content. In the previous section, we configured the gap parameter with a value of 20. If the gap parameter is set to 160 instead of 20, then DANA achieves a recall of 0.8364, a precision of 0.8974 and an F1 score of 0.8571. In this case, DANA outperforms all other algorithms. In our outlook at further work, we will suggest some ideas how to overcome this drawback of DANA to parametrize the gap value.

- Among all eleven algorithms, only DSC and TCCB achieve F1 scores close to but never as high as DANA.

- We can see that DANA also shows considerable efficiency of approximately 19.43 MB/S. Therefore, in comparison with the comparable methods in this chapter – DSC, TCCB-18 and TCCB-25, which have an extraction performance close to our algorithm – DANA

Table 4.5: Precison of DANA based on the corpus in Table 4.1

	Al Ahram	**BBC Arabic**	**BBC Pashto**	**BBC Persian**	**BBC Urdu**	**Embassy**	**Hamshahri**	**Jame Jam**	**Reuters**	**Wikipedia**
ACCB-40	0.920	0.760	0.847	0.851	0.958	0.846	0.774	0.890	0.889	0.925
BTE	0.792	0.340	0.782	0.424	0.926	0.696	0.331	0.675	0.804	0.748
DSC	0.871	0.876	0.882	0.957	0.988	0.890	0.958	0.949	0.862	0.856
FE	0.900	0.460	0.667	0.568	0.035	0.204	0.809	0.145	0.800	0.715
KFE	0.573	0.631	0.840	0.701	0.920	0.795	0.719	0.737	0.946	0.681
LQF-25	0.691	0.648	0.760	0.730	0.917	0.773	0.643	0.605	0.774	0.833
LQF-50	0.688	0.645	0.751	0.711	0.917	0.766	0.644	0.588	0.774	0.721
LQF-75	0.671	0.639	0.751	0.698	0.917	0.760	0.629	0.588	0.774	0.671
TCCB-18	0.907	0.753	0.939	0.892	0.997	0.947	0.821	0.981	1.0	0.889
TCCB-25	0.899	0.818	0.939	0.896	0.993	0.946	0.853	0.981	1.0	0.895
Density	0.926	0.349	0.940	0.803	0.997	0.946	0.947	0.985	0.901	0.889
DANA	0.975	0.965	0.936	0.994	1.0	0.953	0.980	0.960	0.971	0.940

Table 4.6: Average F1 Scores of DANA based on the corpus in Table 4.1

	Al Ahram	**BBC Arabic**	**BBC Pashto**	**BBC Persian**	**BBC Urdu**	**Embassy**	**Hamshahri**	**Jame Jam**	**Reuters**	**Wikipedia**
ACCB-40	0.871	0.826	0.859	0.892	0.948	0.784	0.842	0.840	0.900	0.736
BTE	0.853	0.496	0.854	0.589	0.961	0.810	0.480	0.791	0.889	**0.817**
DSC	0.871	*0.885*	0.840	*0.950*	0.896	0.824	*0.948*	0.914	0.851	0.747
FE	0.809	0.060	0.165	0.063	0.002	0.017	0.225	0.027	0.241	0.225
KFE	0.690	0.717	0.835	0.748	0.750	0.762	0.678	0.783	0.825	0.624
LQF-25	0.788	0.780	0.844	0.841	0.957	0.860	0.765	0.737	0.870	0.773
LQF-50	0.785	0.777	0.837	0.828	0.954	0.856	0.767	0.724	0.870	0.772
LQF-75	0.773	0.773	0.837	0.819	0.954	0.852	0.756	0.724	0.870	0.750
TCCB-18	*0.886*	0.826	*0.912*	0.925	0.990	*0.887*	0.871	*0.929*	*0.959*	0.814
TCCB-25	0.874	0.861	0.909	0.927	*0.992*	0.883	0.888	0.924	0.958	0.814
Density	0.879	0.202	0.908	0.741	0.958	0.882	0.920	0.907	0.934	0.665
DANA	**0.984**	**0.963**	**0.936**	**0.994**	**1.0**	**0.935**	**0.978**	**0.945**	**0.967**	0.674

Table 4.7: Average processing performance (MB/s)

Method	Performance (MB/s)
ACCB-40	0.40
BTE	0.17
DSC	7.76
FE	14.33
KFE	11.76
LQF-25	1.25
LQF-50	1.25
LQF-75	1.25
TCCB-18	17.09
TCCB-25	15.86
Density	7.62
DANA	**19.43**

has an acceptable efficiency. On Wikipedia, BTE achieves extraction performance superior to DANA, but DANA is about 100 times faster than BTE.

4.5 Algorithm DANAg

The R2L and DANA algorithms are both language-dependent, while the second extension of R2L, called DANAg (84) (87), is a generalized method which is able to run on web pages written in any language. The extraction process of DANAg is divided into one preprocessing step as well as four phases. The preprocessing step and phases two, three, and four are equivalent to the DANA approach. In the following, we will only explain the differences in phase one.

4.5.1 Calculating the Length of Content and Code of Each Line

In the first phase of the algorithm DANAg, our aim is to count and store the number of characters comprising both the content and the code of the lines of the HTML file into two one-dimensional arrays $T1$ and $T2$, respectively.

To provide two one-dimensional arrays $T1$ and $T2$, we first count and store the number of characters of each line into the one-dimensional array $Length$. In the second step, we feed the HTML file to our parser to extract all words representing the content of the HTML file. By this method, we are able to count and store the number of content characters for each line in a one-dimensional array $T1$. Formula 4.5 shows how the number of characters in each line used in code elements are calculated and stored in array $T2$.

$$T2 = Length - T1 \tag{4.5}$$

Although DANAg generalizes the R2L and DANA to a language-independent approach, it is expected that incorporating the parser directly in this phase of DANAg causes a significant overhead in computation and that DANAg runs slower than DANA.

Table 4.8: Average F1 Scores of DANAg based on the corpus in Table 4.1

	Al Ahram	**BBC Arabic**	**BBC Pashto**	**BBC Persian**	**BBC Urdu**	**Embassy**	**Hamshahri**	**Jame Jam**	**Reuters**	**Wikipedia**
ACCB-40	0.871	0.826	0.859	0.892	0.948	0.784	0.842	0.840	0.900	0.736
BTE	0.853	0.496	0.854	0.589	0.961	0.810	0.480	0.791	0.889	**0.817**
DSC	0.871	*0.885*	0.840	*0.950*	0.896	0.824	*0.948*	0.914	0.851	0.747
FE	0.809	0.060	0.165	0.063	0.002	0.017	0.225	0.027	0.241	0.225
KFE	0.690	0.717	0.835	0.748	0.750	0.762	0.678	0.783	0.825	0.624
LQF-25	0.788	0.780	0.844	0.841	0.957	0.860	0.765	0.737	0.870	0.773
LQF-50	0.785	0.777	0.837	0.828	0.954	0.856	0.767	0.724	0.870	0.772
LQF-75	0.773	0.773	0.837	0.819	0.954	0.852	0.756	0.724	0.870	0.750
TCCB-18	*0.886*	0.826	*0.912*	0.925	0.990	*0.887*	0.871	*0.929*	*0.959*	0.814
TCCB-25	0.874	0.861	0.909	0.927	*0.992*	0.883	0.888	0.924	0.958	0.814
Density	0.879	0.202	0.908	0.741	0.958	0.882	0.920	0.907	0.934	0.665
DANA	**0.984**	0.963	0.936	0.994	**1.0**	**0.935**	0.978	0.945	**0.967**	0.674
DANAg	0.949	**0.986**	**0.944**	**0.995**	**0.999**	0.917	**0.991**	**0.966**	0.922	0.699

4.5.2 Results

Tables 4.8 and 4.9 give statistics showing the average F1 scores of DANAg and other main content extraction algorithms on both data sets. In addition, in Table 4.10 we compute the processing time (MB/s) of DANAg and other approaches on the data described in Table 4.1. By looking at the Tables 4.8 and 4.9, the following important observations can be made:

Results on Arabian language documents

- As can be seen in Table 4.8 from six web pages, Al Ahram, BBC Arabic, BBC Persian, BBC Urdu, Hamshahri, and Reuters, DANAg achieves an F1 score of more than 0.95 and especially on BBC Urdu with an F1 score extremely close to 1. In addition, no other method shows such a high effectiveness.

- Among all eleven algorithms, only DSC and TCCB achieve F1 scores close to but never as high as DANAg.

Results on Western language documents Now, we describe the results in Table 4.9. For higher clarity, this table was divided into three parts. We explain each part of this table below:

- In the middle part of the Table 4.9, DANAg achieves F1 score higher than other algorithms on the six web pages golem, heise, republica, spiegel, telepolis, and yahoo. As can be seen, ACCB is the best algorithm on three web pages among all other algorithms after DANAg.

- The left side of Table 4.9 shows three web pages in which DANAg achieves F1 score less than the DSC, CCB, and ACCB approaches. But as it can be seen, the differences between the F1 score of DANAg and last three mentioned methods are 0.013, 0.0144, and 0.017, and this shows that DANAg could be acceptable on these web pages as well.

- On the right side of Table 4.9, we see the three web pages manual, slashdot, and wikipedia in which DANAg and other algorithms could not achieve a considerable F1 score. For a better explanation about these three web pages, we depict six figures and we describe the behavior of these figures in the next paragraphs.

  Figure 4.5 and Figure 4.6 are original and smoothed diagrams of an example of a Wikipedia web page. The recall, precision and F1 scores of this web page achieved by DANAg are 0.3636, 0.8889, and 0.5161, respectively. In Figure 4.5, we determined the main content area that should be extracted, but Figure 4.6 shows that only a small part of the web page was obtained. If we use Formula 4.6 instead of Formula 4.2 as a smoothing function, then the variables recall, precision, and F1 achieve values of 0.6469, 0.9158, and 0.7582. We conclude that perhaps, for fully structure web pages, it is better to use the smoothing method in Formula 4.6.

$$diff_i \quad = \quad (T1_{\mathrm{i}} - T2_{\mathrm{i}}) \tag{4.6}$$

  Figures 4.7 and 4.8 depict original and smoothed diagrams of a sample of a manual web page. The recall, precision and F1 scores of this web page are 0.9783, 0.5396, and 0.6959, respectively. In Figure 4.7, we draw an ellipse to specify the main content area that should be extracted. Figure 4.8 shows that after running DANAg, a large part of extraneous items, which should not be extracted, was retrieved as main content. In manual web pages, the distance between the main content area and area comprising menus and advertisement is less than the value of gap, 20, which has been defined in our project. In the next section as well as in "Conclusions and Future Directions", we will show what should be done in order to solve this problem.

  In Figure 4.9 and Figure 4.10 we depict the original and smoothed plots of an example of a slashdot web page. As these plots show, these web pages are full of extraneous items. Hence, many main content approaches could not extract the main content of these noisy web pages.

We can see that DANAg also shows a remarkable efficiency of approximately 11.41 MB/s. Therefore, in comparison with the comparable methods in this chapter (DSC, TCCB-18 and TCCB-25, which have an extraction performance close to our algorithm), DANAg has an acceptable efficiency.

As explained in 4.5.1, applying the parser in the first phase of DANAg causes an overhead in computation and as shown in Table 4.7, DANAg has a lower efficiency in comparison to DANA (19.43 MB/s).

Table 4.9: Average F1 Scores of DANAg based on the corpus in Table 4.2

	BBC	**Economist**	**Zdf**	**Golem**	**Heise**	**Republica**	**Spiegel**	**Telepolis**	**Yahoo**	**Wikipedia**	**Manual**	**Slashdot**
Plain	0.595	0.613	0.514	0.502	0.575	0.704	0.549	0.906	0.582	**0.823**	0.371	0.106
LQF	0.826	0.720	0.578	0.806	0.787	0.816	0.775	0.910	0.670	0.752	0.381	0.127
Crunch	0.756	0.815	0.772	0.837	0.810	0.887	0.706	0.859	0.738	0.725	0.382	0.123
DSC	**0.937**	0.881	0.847	0.958	0.877	0.925	0.902	0.902	0.780	0.594	0.403	0.252
TCCB	0.914	0.903	0.745	0.947	0.821	0.918	*0.910*	*0.913*	0.758	0.660	0.404	0.269
CCB	0.923	**0.914**	0.929	0.935	0.841	0.964	0.858	0.908	0.742	0.403	0.420	0.160
ACCB	0.924	0.890	**0.929**	*0.959*	*0.916*	*0.968*	0.861	0.908	0.732	0.682	**0.419**	0.177
Density	0.575	0.874	0.708	0.873	0.906	0.344	0.761	0.804	*0.886*	0.708	0.354	**0.362**
DANAg	0.924	0.900	0.912	**0.979**	**0.945**	**0.970**	**0.949**	**0.932**	**0.952**	0.646	0.401	0.209

Table 4.10: Average processing performance (MB/s)

Method	**Performance (MB/s)**
ACCB-40	0.40
BTE	0.17
DSC	7.76
FE	14.33
KFE	11.76
LQF-25	1.25
LQF-50	1.25
LQF-75	1.25
TCCB-18	17.09
TCCB-25	15.86
Density	7.62
DANA	**19.43**
DANAg	**11.41**

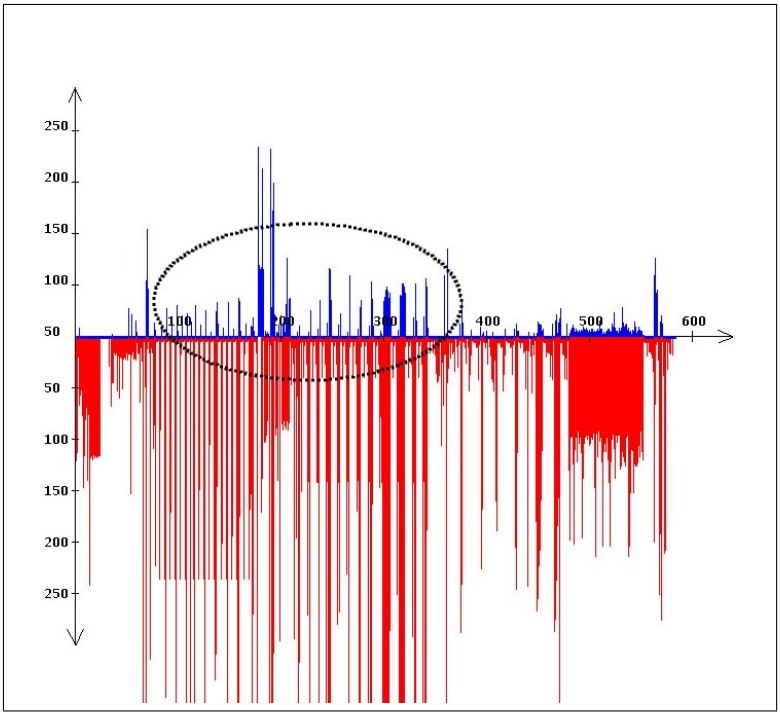


Figure 4.5: Original diagram of wikipedia

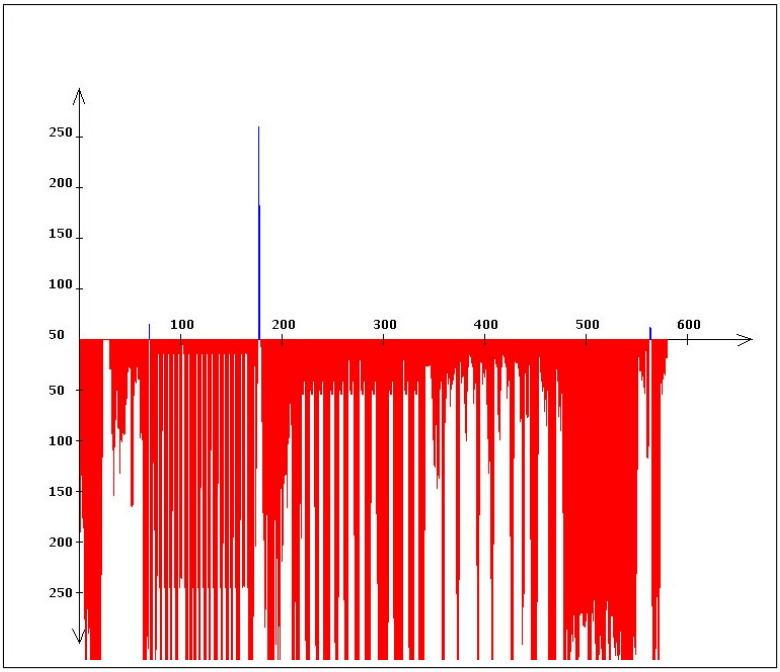


Figure 4.6: Smoothed diagram of wikipedia

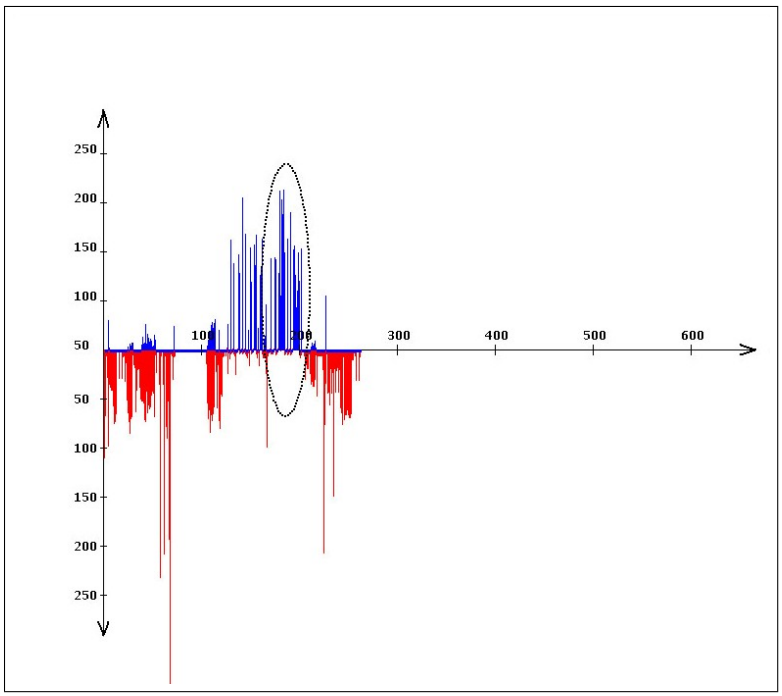


Figure 4.7: Original diagram of manual

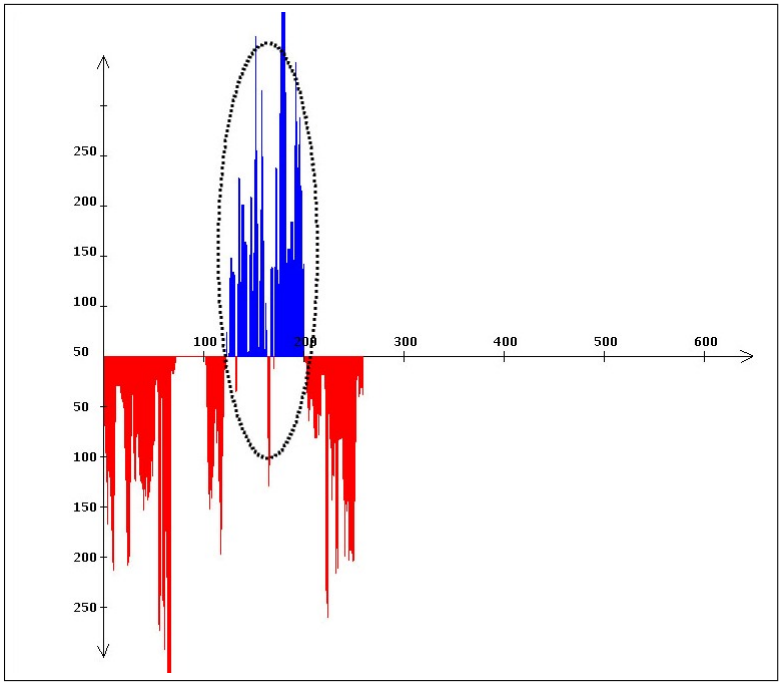


Figure 4.8: Smoothed diagram of manual

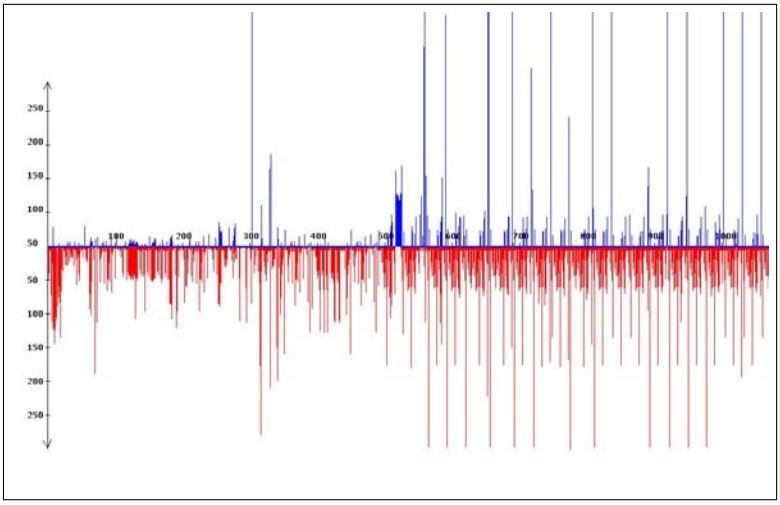


Figure 4.9: Original diagram of slashdot

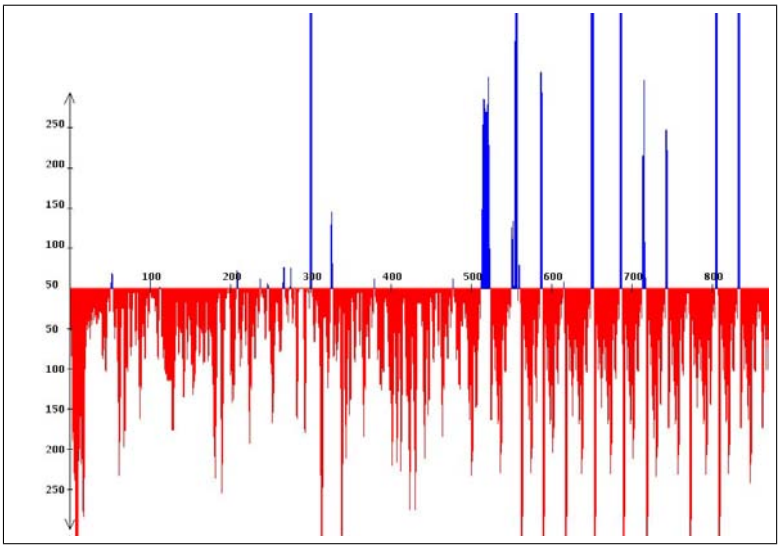


Figure 4.10: Smoothed diagram of slashdot

4.6 Algorithm AdDANAg

AdDANAg (86) is inspired by an adaptation of the preprocessing step of ACCB (48) and DANAg (84). The process behind AdDANAg can be divided into two preprocessing phases and a core extraction phase. While the first preprocessing phase and the core extraction phase are taken from DANAg, the second preprocessing from ACCBs has been adapted and enhanced.

In the first preprocessing phase of the AdDANAg algorithm (which is exactly similar to R2L, DANA, and DANAg), all JavaScript codes, CSS style codes, and comments are removed from the HTML file and retain only the HTML code as the output. Furthermore, it normalizes the distribution of line breaks characters in the source code as it operates on the level of lines.

Our second preprocessing phase normalizes imbalances in the source code structure that hinder typical CE approaches. The imbalances can be motivated due to technical constraints or domain specific deviations from the typical source code patterns. It is important to note that they do not imply a semantic change in the main content specifications.

Below we explain the second preprocessing step of AdDANAg in detail and recall the core extraction phase for the sake of completeness on Wikipedia web pages. In addition, for the purpose of evaluation, we use the datasets in Tables 4.1 and 4.2 introduced in Section 4.1, composed of 9,101 and 2,166, respectively, web pages from different web sites. Furthermore, keep in mind that we prepared a gold standard for each web page.

4.6.1 The Second Preprocessing Step of AdDANAg

A common problem of source code based content extraction methods with hyperlink rich web documents is that the main content can not be detected accurately. This can be explained with the code for hyperlinks prevailing over the actual content items, which contradicts typical assumptions made by the content extraction methods. Figure 4.11 shows some paragraphs of a BBC HTML file and Figure 4.12 represents some portions of a typical Wikipedia source code. In Figure 4.11, there is no hyperlink, so DANAg approach extracts the main content accurately. In comparison, in Figure 4.12, there are plenty of hyperlinks in which the length of attributes are much more than the length of anchor texts.

In this section, the preprocessing step of ACCB will be explained first and will be called Filter 1 hereafter. Then, two new methods of preprocessing will be introduced which will be named Filter 2 and Filter 3 for simplicity. Between Filter 2 and Filter 3, the preprocessor of Filter 3 will be used in the AdDANAg algorithm.

4.6.1.1 Filter 1

As mentioned in Section 3.3.1.3, in the ACCB algorithm, which is an adapted version of CCB, all anchor tags are removed from HTML files during preprocessing step, i.e. Filter 1, and then the core section of algorithm ACCB will be provided with a new HTML file for further processes. It will be demonstrated in the next sections that by the application of Filter 1 with the DANAg algorithm, some results will be obtained with lesser accuracy than that produced from AdDANAg (i.e. application of Filter 3 with DANAg algorithm).

```
</div>
      <p id="story_continues_2">&quot;Especially the under-fives and the pregnant
women, they&#039;re suffering from malnutrition and communicable disease like the
measles, diarrhoea and pneumonia,&quot; he said.</p>

        <p>Earlier this week Mark Bowden, the UN humanitarian affairs co-ordinator
for Somalia, told the BBC that the country was close to famine.  </p>
        <p>Last week Somalia&#039;s al-Shabab Islamist militia - which has been
fighting the Mogadishu government - said it was lifting its ban on foreign aid
agencies provided they did not show a &quot;hidden agenda&quot;. </p>
        <p>Some 3,000 people flee each day for neighbouring countries such as
Ethiopia and Kenya which are struggling to cope.</p>

</div>
```

Figure 4.11: Some Paragraphs of a BBC HTML file

```
<p>The present significance of IE pertains to the growing amount of information
available in unstructured form. <a href="/wiki/Tim_Berners-Lee" title="Tim
Berners-Lee">Tim Berners-Lee</a>, inventor of the <a href="/wiki/World_wide_web"
title="World wide web" class="mw-redirect">world wide web</a>, refers to the
existing <a href="/wiki/Internet" title="Internet">Internet</a> as the web of
<i>documents</i> <sup id="cite_ref-2" class="reference"><a
href="#cite_note-2"><span>[</span>3<span>]</span></a></sup> and advocates that
more of the content be made available as a <a href="/wiki/Semantic_web"
title="Semantic web" class="mw-redirect">web of <i>data</i></a>.<sup
id="cite_ref-3" class="reference"><a
href="#cite_note-3"><span>[</span>4<span>]</span></a></sup> Until this transpires,
the web largely consists of unstructured documents lacking semantic <a
href="/wiki/Metadata" title="Metadata">metadata</a>. Knowledge contained within
these documents can be made more accessible for machine processing by means of
transformation into <a href="/wiki/Relational_database" title="Relational
database">relational form</a>, or by marking-up with <a href="/wiki/XML"
title="XML">XML</a> tags. An intelligent agent monitoring a news data feed
requires IE to transform unstructured data into something that can be reasoned
with. A typical application of IE is to scan a set of documents written in a <a
href="/wiki/Natural_language" title="Natural language">natural language</a> and
populate a database with the information extracted.<sup id="cite_ref-4"
class="reference"><a href="#cite_note-4"><span>[</span>5<span>]</span></a></sup></
p>

<h2><span class="editsection">[<a href="/w/index.php?
title=Information_extraction&amp;action=edit&amp;section=3" title="Edit section:
IE tasks and subtasks">edit</a>]</span>
```

Figure 4.12: Some portions of a Wikipedia HTML file

Algorithm 2 shows the simple logic used in Filter 1. It can be seen that one just needs to remove all the existing hyperlinks in an HTML file which is done at line 5 of this algorithm. The only disadvantage of this preprocessing is that by removing the hyperlinks, the anchor texts are also removed. As a result, this will cause the existing hyperlinks in the main content to finally be removed. Thus, their anchor texts, which must be seen in the main content, do not exist in the final main content. Consequently, the application of Filter 1 will reduce the accuracy or the amount of recall.

Algorithm 2 Filter 1 used in ACCB

1: $Hyper = \{h_1, h_2, ..., h_n\}$
2: $i = 1$
3: **while** $i <= n$ **do**
4: $\quad h_i.remove()$
5: $\quad i = i + 1$
6: **end while**

4.6.1.2 Filter 2

An idea which is utilized in Filter 2 is a little different than that of Filter 1. With respect to Algorithm 3, which shows the codes of Filter 2, one can understand that the attribute of each anchor tag is removed in Filter 2. Thereby, an anchor tag will only contain an anchor text, as shown below:

```
<a>anchor text</a>
```

An advantage of Filter 2 over Filter 1 is that some anchor texts related to anchor tags, which are located in the main content area, can be extracted using Filter 2. In other words, the amount of recall which is obtained from the application of Filter 2 is greater than that produced from Filter 1. The application of Filter 2 with DANAg might possibly cause some problems and reduce the accuracy of the main content extraction algorithm, but how?

We know that each of the lines of the HTML file were smoothed in the second phase of the Algorithms R2L, DANA and DANAg by the application of Formula 4.2. The lines with $diff > 0$ were possibly able to be selected as a part of main content. Now, assume that three anchor tags exist in a Wikipedia file as evident from List 4.1:

By the application of Filter 2 and later using formulae 4.2 on the lines in the List 4.1, the second anchor tag will certainly show $diff > 0$. This means that this line has the potential to be a part of the main content.

Listing 4.1: Sample of HTML file with three anchor tags which are extraneous items

```
<a href="link␣one">anchor text one</a>
<a href="link␣two">anchor text two</a>
<a href="link␣three">anchor text three</a>
```

Having specified the first region or line of the main content in the third phase of the Algorithms R2L, DANA, DANAg, one should now look for other regions and lines of the HTML file to construct the entire main content. As mentioned in Section 4.3.4, a region or a line of the HTML can be a part of MC when it first of all has a $diff > 0$ (the second anchor tag in List 4.1 has a $diff > 0$) and secondly, the distance of that region or line of the HTML document from the MC area specified so far does not exceed 20 lines ($gap = 20$). Therefore, whenever the distance of the second existing anchor tag in List 4.1 from the MC specified so far is smaller than 20 lines, then the second anchor tag will be selected as the MC. However, it is also possible that this tag belongs to a part of extraneous items instead of

MC. This assumption is probable since when it is part of MC, some words and sentences are seen around this anchor tag. For example, each line of a Wikipedia file contains a number of anchor tags which are surrounded by some words and sentences. Consequently, if we consider the anchor text of the second anchor tag as one part of MC, then the precision obtained from the application of Filter 2 will be reduced. It is interesting to note that if Filter 1 was used here in these conditions, the anchor tags would have been completely removed and thus the precision would not have been diminished.

Algorithm 3 Filter 2

1: $Hyper = \{h_1, h_2, ..., h_n\}$
2: $i = 1$
3: **while** $i <= n$ **do**
4: $\quad h_i.href.remove()$
5: $\quad i = i + 1$
6: **end while**

4.6.1.3 Filter 3

In the third preprocessing method, called Filter 3, which is used in AdDANAg, we normalise all HTML hyperlinks using a fast approach based on substitution rules. For better comprehension and simplification, we will explain the approach using a typical example. Suppose the following hyperlink to be contained in an HTML file:

```
<a href="http://www.BBC.com/">BBC Web Site</a>
```

Here, there is only one attribute, which is `href="http://www.BBC.com/"`. Now, the length of the anchor text (in this example: `BBC web Site`) is determined and we refer to this value by LT. Then, we substitute the attribute part of the opening tag with a placeholder text with the length LT - 7, where the subtracted 7 comes from the length of `<a></a>`. Therefore, using the underscore sign _ as placeholder, the new hyperlink for our example should be as below:

```
<a _______>BBC Web Site</a>
```

The purpose of this rule is to normalise the ratio of content and code characters representing hyperlinks. This counterbalances inequalities originating from the URLs in hyperlinks.

These explanations are summarized in Algorithm 4. As can be observed in this algorithm, the inside `while` loop, which is repeated for n times, the length of the anchor text related to each hyperlink is calculated first and stored in the LT variable. Then, a string of LT-7 length is made of the sign _ and then placed in the string variable "Str". Finally, the attribute part of hyperlink is replaced with the Str string.

The advantage of using Filter 3 over Filter 1 is that by the running of the second phase of the Algorithms R2L, DANA and DADAg on the anchor tags in List 4.1, the second anchor

tag will bear *diff* = 0. It means that this line cannot be considered as a part of MC. As discussed earlier in section 4.6.1.2, if this anchor tag is not a part of menus or extraneous items, a number of words and sentences will exist around them, as in Listing 4.2. For the lines in Listing 4.2, the amount of *diff* is certainly greater than 0. This means that these lines will be selected as the MC when conditions of the third phase of the Algorithms R2L, DANA and DANAg are met.

Algorithm 4 Filter 3 used in AdDANAg

1: $Hyper = \{h_1, h_2, ..., h_n\}$
2: $i = 1$
3: **while** $i <= n$ **do**
4: $LT = Length(h_i.anchor\ \ text)$
5: $String\ \ Str = new\ \ String(\text{“ ”}, LT - 7)$
6: $substitute(h_i.attribute, Str)$
7: $i = i + 1$
8: **end while**

Listing 4.2: Sample of HTML file with three anchor tags in the main content area

```
These three anchor tags are not extraneous items. <a href="link␣one">anchor text one</a>
They are located in one of the main content regions. <a href="link␣two">anchor text two</a>
Using Filter 3 we are able to extract anchor texts of hyperlinks. <a href="link␣three">
anchor text three</a> Filter 3 is used in AdDANAg as a preprocessing step.
```

4.6.2 The Core Extraction Phase

The core extaraction phase of DANAg and its new version, AdDANAg, is divided into four phases. In the first phase, it calculates the length of content and code of each line of the HTML file and stores these numbers in two one-dimentional arrays $T1$ and $T2$, respectively. For each line of the HTML file, Figure 4.13 draws two types of columns above and below the x-axis with the length equal to the values of these two one-dimentional arrays, since the measurement unit for the y-axis upward and downward is the length of content and code of each line of an HTML file. Furthermore, in this figure, each column exactly corresponds to an individual line of the Wikipedia web page demonstrated in Figure 4.15, because the measurement unit for the x-axis is the number of lines in the HTML file. In Figure 4.15, all the hyperlinks are highlighted and it can be seen that the density of hyperlinks are extremly high on the Wikipedia web pages. AdDANAg and DANAg algorithms' hypothesis is that the main content is typically positioned above the x-axis. In Figure 4.13, the main content area is located in the interval [15, 540] and the rest of the diagram belongs to the extraneous items such as menus and advertisements.

The normalisation of the hyperlinks in the second preprocessing phase in AdDANAg takes effect, since for each hyperlink, it equally modifies the values in $T1$ and $T2$. This is the key point of AdDANAg. In Figure 4.14, a new plot of Figure 4.13 is drawn based on this outstanding preprocessing phase. In this Figure, the length of codes in each line of the HTML file is shorter than the length of codes in Figure 4.13. In other words, the ratio of

content to code in Figure 4.14 is greater than the same ratio in Figure 4.13. Consequently we will show, using Wikipedia web pages, that combining the Algorithm DANAg with our newly proposed preprocessing phase, now called AdDANAg, we can retain columns located above the x-axis representing the main content.

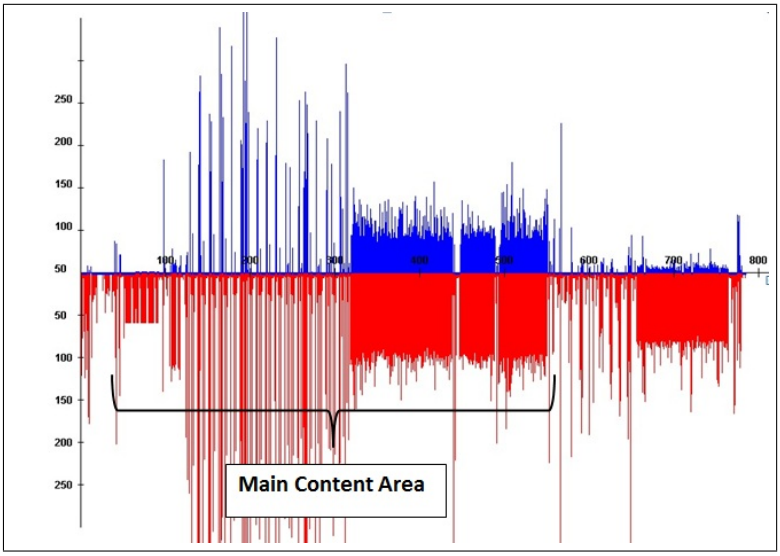


Figure 4.13: Example plot shows the main content area in Wikipedia web pages

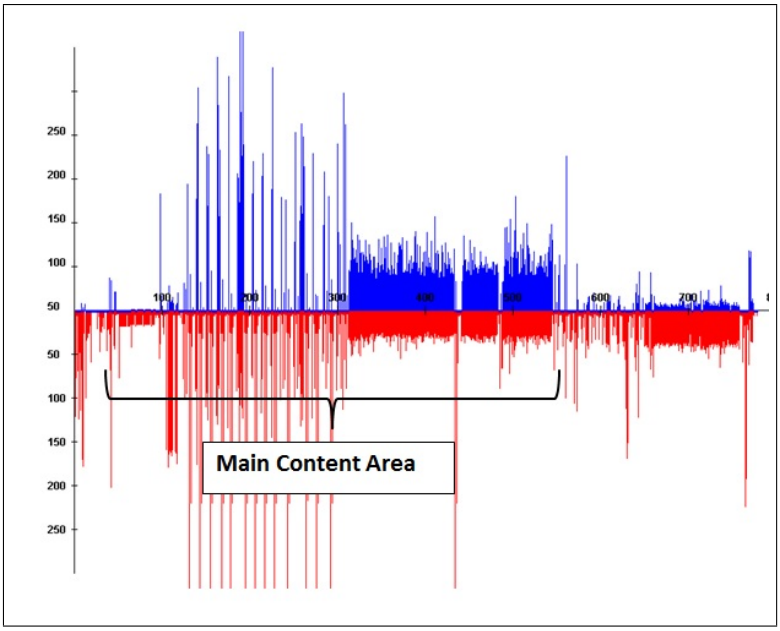


Figure 4.14: New plot of Figure 4.13 affected by using second preprocessing step in AdDANAg

Figure 4.15: Example of Wikipedia web pages

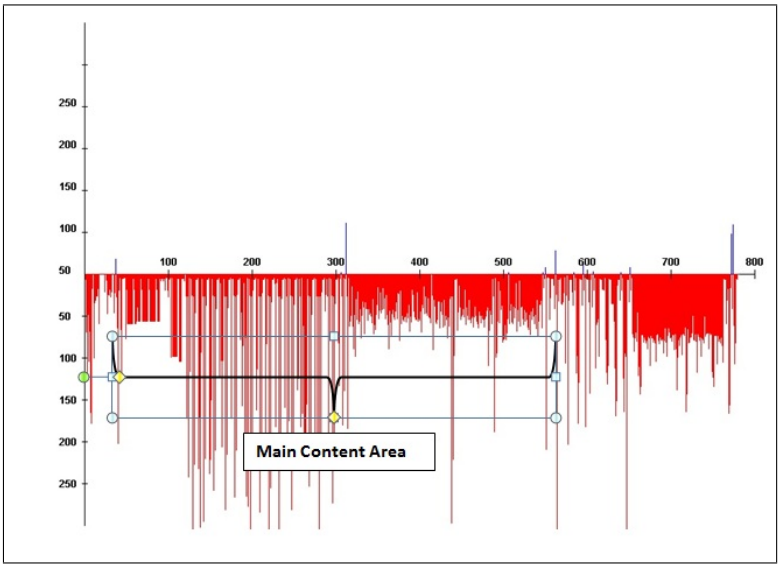


Figure 4.16: Smoothed plot of Figure 4.13

In the second phase, DANAg and AdDANAg compute *diff* through Formula 4.2 for each line of the HTML file, and keep these new numbers in a one-dimensional array $T3$ in order to produce a smoothed plot of original plot. In a smoothed plot, a column is drawn above the x-axis if $diff > 0$. Otherwise, a line with the length $|diff|$ is depicted below the x-axis. The smoothed plot of Figure 4.13 can be seen in Figure 4.16 if we run the DANAg Algorithm. As a result, all the extraneous items in Figure 4.16 have been hidden, but unfortunately, most parts of the main content area have been removed as well and this is not what DANAg expected as the output; in contrast, AdDANAg produces a graph including plenty of columns in the main content area. Figure 4.17 demonstrates a smoothed plot of Figure 4.14. Comparing both Figure 4.16 and Figure 4.17 show that AdDANAg can keep all columns above the x-axis comprising the main content area.

Considering the positive values of $T3$, both algorithms AdDANAg and DANAg identify all regions located above the x-axis and, for simplicity, they define a new set $R = \{r_1, r_2, ..., r_n\}$ of all such regions. Each element $r_j \in R$ denotes only one individual paragraph or line and n is the total number of recognized paragraphs above the x-axis.

In the third phase, AdDANAg discovers all paragraphs shaping the main content using Algorithm 1. Finally, AdDANAg feeds all these extracted paragraphs to a parser to obtain the main content of the HTML file.

4.6.3 Results

The F1 scores of AdDANAg, DANAg, and other main content extraction algorithms on both data sets shown in Tables 4.1 and 4.2 are presented in Table 4.11 and Table 4.12. A first observation is that in comparison to DANAg, AdDANAg does not show major drawbacks. Furthermore, AdDANAg and DANAg for most documents deliver the best results. When focusing on the set of Wikipedia web pages, which have been observed to be extremely difficult

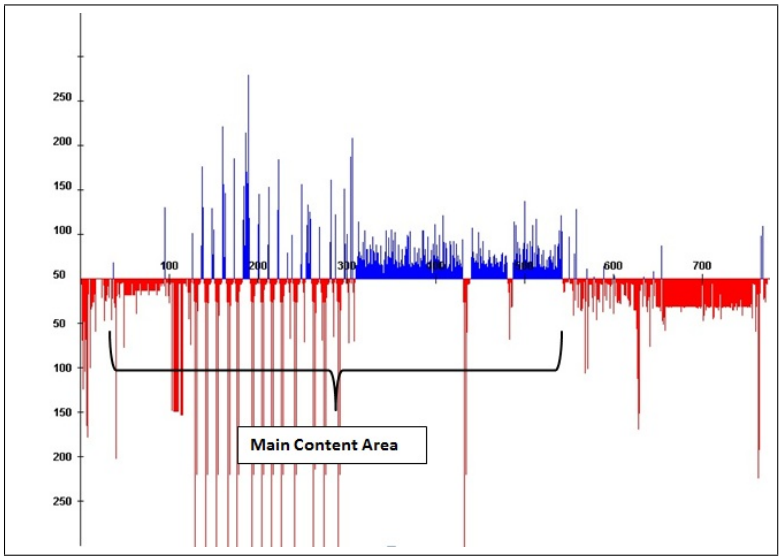


Figure 4.17: Smoothed plot of Figure 4.14

for main content extraction algorithms, we can observe that AdDANAg clearly outperforms DANAg and all other approaches. By looking at the Table 4.11 and Table 4.12, the following observations can be made:

Results on Arabian language documents

- The overall average F1 score for both DANAg and AdDANAg on both Tables 4.11 and 4.12 are 0.8686 and 0.886, respectively. It can be concluded that AdDANAg is able to extract MC at a greater accuracy than the DANAg Algorithm.
- Among all eleven algorithms, only DSC and TCCB achieve F1 scores close to but never as high as AdDANAg.
- As can be seen in Table 4.11, DANAg algorithm has achieved the greatest value of the F1 score on seven web pages. On the other hand, AdDANAg algorithm has the highest value of the F1 score on eight web pages. It is interesting to note that the algorithms DANAg and AdDANAg have equal F1 scores on the web pages of Ahram, BBC Pashto, BBC Persian, BBC Urdu, Embassy and Hamshahri. However, this value is greater than the F1 scores obtained by other algorithms, i.e. ACCB, BTE, DSC, FE, K-FE, LQF, CCB and Density.
- Although the AdDANAg Algorithm has not achieved the highest value of the F1 score on the BBC Arabic web page (Table 4.11), its difference to the F1 score from the DANAg Algorithm is reported as being only 0.001, which can be said to be almost negligible. Generally speaking, the F1 score obtained by the AdDANAg Algorithm is greater than that of other algorithms, except DANAg, on a BBC Arabic web page.

Table 4.11: Average F1 Scores of AdDANAg based on the corpus in Table 4.1

	Al Ahram	**BBC Arabic**	**BBC Pashto**	**BBC Persian**	**BBC Urdu**	**Embassy**	**Hamshahri**	**Jame Jam**	**Reuters**	**Wikipedia**
ACCB-40	0.871	0.826	0.859	0.892	0.948	0.784	0.842	0.840	0.900	0.736
BTE	0.853	0.496	0.854	0.589	0.961	0.810	0.480	0.791	0.889	0.817
DSC	0.871	*0.885*	0.840	*0.950*	0.896	0.824	*0.948*	0.914	0.851	0.747
FE	0.809	0.060	0.165	0.063	0.002	0.017	0.225	0.027	0.241	0.225
KFE	0.690	0.717	0.835	0.748	0.750	0.762	0.678	0.783	0.825	0.624
LQF-25	0.788	0.780	0.844	0.841	0.957	0.860	0.765	0.737	0.870	0.773
LQF-50	0.785	0.777	0.837	0.828	0.954	0.856	0.767	0.724	0.870	0.772
LQF-75	0.773	0.773	0.837	0.819	0.954	0.852	0.756	0.724	0.870	0.750
TCCB-18	*0.886*	0.826	*0.912*	0.925	0.990	*0.887*	0.871	*0.929*	**0.959**	0.814
TCCB-25	0.874	0.861	0.909	0.927	*0.992*	0.883	0.888	0.924	0.958	0.814
Density	0.879	0.202	0.908	0.741	0.958	0.882	0.920	0.907	0.934	0.665
DANAg	**0.949**	**0.986**	**0.944**	**0.995**	**0.999**	**0.917**	**0.991**	0.966	0.945	0.699
AdDANAg	**0.949**	0.985	**0.944**	**0.996**	**0.999**	**0.917**	**0.991**	**0.973**	0.945	**0.852**

- On Reuters web page in Table 4.11, only the algorithms TCCB-18 and TCCB-25 show a greater F1-score than the algorithms AdDANAg and DANAg. The value of the F1 score obtained by TCCB-18 is equal to 0.959, while the calculated F1 score from TCCB-25 is 0.958. However, the values of the F1 score obtained from DANAg and AdDANAg are the same (0.945) in this case. The average F1 score is calculated as 0.784 on Reuters web page, which demonstrate that both DANAg and AdDANAg algorithms have produced a significant F1 score for this web site.

- The most important part of Table 4.11 is related to the Wikipedia web site in which the amount of the F1 score from the AdDANAg Algorithm is equal to 0.852, whereas this value is reported as 0.699 in the DANAg Algorithm. Among other existing algorithms in Table 4.11, BTE, TCCB-18 and TCCB-25 algorithms have the greatest values of the F1 score, reported as 0.817, 0.814 and 0.814, respectively. As a result, the minimum improvement in F1 score which is achieved by the AdDANAg Algorithm is 0.035, which is considerable in this case.

Results on Western language documents Now we will describe the results in Table 4.12. For a better understanding, this table was divided into three parts. These are explained here:

- Three web sites of BBC, Economics and ZDF can be observed on the left side of Table 4.12, where AdDANAg has failed to get the highest values of the F1 score on these web sites. Noteworthy here is that the values of F1 score obtained by AdDANAg and DANAg are almost the same. By looking at the difference between the maximum F1 score obtained on the web sites of BBC, Economics and zdf and the F1 score produced by AdDANAg algorithm (0.015, 0.0144 and 0.018, respectively), one may conclude

Table 4.12: Average F1 Scores of AdDANAg based on the corpus in Table 4.2

	BBC	**Economist**	**Zdf**	**Golem**	**Heise**	**Republica**	**Spiegel**	**Telepolis**	**Yahoo**	**Wikipedia**	**Manual**	**Slashdot**
Plain	0.595	0.613	0.514	0.502	0.575	0.704	0.549	0.906	0.582	*0.823*	0.371	0.106
LQF	0.826	0.720	0.578	0.806	0.787	0.816	0.775	0.910	0.670	0.752	0.381	0.127
Crunch	0.756	0.815	0.772	0.837	0.810	0.887	0.706	0.859	0.738	0.725	0.382	0.123
DSC	**0.937**	0.881	0.847	0.958	0.877	0.925	0.902	0.902	0.780	0.594	0.403	0.252
TCCB	0.914	0.903	0.745	0.947	0.821	0.918	*0.910*	*0.913*	0.758	0.660	0.404	0.269
CCB	0.923	**0.914**	0.929	0.935	0.841	0.964	0.858	0.908	0.742	0.403	0.420	0.160
ACCB	0.924	0.890	**0.929**	*0.959*	*0.916*	*0.968*	0.861	0.908	0.732	0.682	**0.419**	0.177
Density	0.575	0.874	0.708	0.873	0.906	0.344	0.761	0.804	*0.886*	0.708	0.354	**0.362**
DANAg	0.924	0.900	0.912	0.979	**0.945**	**0.970**	0.949	**0.932**	**0.952**	0.646	0.401	0.209
AdDANAg	0.922	0.900	0.911	**0.994**	0.931	**0.970**	**0.951**	**0.932**	0.950	**0.840**	0.404	0.236

that this difference is not considerable. Thus, AdDANAg can also be applied as an acceptable method on these web sites.

- Seven web sites can be observed in the middle part of Table 4.12, where AdDANAg has been able to achieve the greatest F1 score in comparison with the other algorithms. The amount of the F1 score on Heise and Yahoo web sites obtained by DANAg is a little greater than that obtained from AdDANAg, though the difference is really small. The most important point in this section is that the AdDANAg Algorithm shows a considerable improvement over the DANAg Algorithm on the web site of Wikipedia. The F1 score of DANAg algorithm on this web site is 0.646, while the F1 score from the AdDANAg Algorithm is equal to 0.840 which demonstrates a 0.206 growth.

- Two web sites of Manual and Slashdot web sites are seen on the right part of Table 4.12. As explained before in section 4.5.2, no MC extraction algorithm has been able to successfully extract the MC from these web sites.

Comparing Filter 1, Filter 2, and Filter 3 Tables 4.13 and 4.14 list the results obtained, i.e. recall, precision and F1 score, from combining each of the filters introduced in this section with DANAg algorithm. It is well known that the AdDANAg Algorithm is produced from the combination of the DANAg Algorithm with Filter 3. Each of these tables is divided into three parts: the first part contains 3 rows and adopts to compare the recalls; the second part also includes 3 rows and compares the precision; and finally, the third section will compare the F1 score. By looking at Tables 4.13 and 4.14, one can conclude the following points:

- As seen in the third part of both Tables 4.13 and 4.14, Filter 3 has acquired a better F1 score in comparison with the other two filters, i.e. Filter 1 and Filter 2, in most of the 18 cases. In contrast, Filter 2 has obtained the minimum amount of F1 score as compared with Filters 1 and 3.

Table 4.13: Comparing Recall, Presion and F1 of Normalization Methods based on the corpus in Table 4.1

	Al Ahram	**BBC Arabic**	**BBC Pashto**	**BBC Persian**	**BBC Urdu**	**Embassy**	**Hamshahri**	**Jame Jam**	**Reuters**	**Wikipedia**
Recall, Filter 1	0.942	0.987	0.961	0.997	0.999	0.953	0.953	0.963	1.0	0.853
Recall, Filter 2	0.942	0.989	0.961	0.997	0.999	0.953	0.942	0.963	1.0	0.886
Recall, Filter 3	0.942	0.987	0.959	0.997	0.999	0.949	0.993	0.97	1.0	0.81
Precision, Filter 1	0.969	0.952	0.929	0.973	0.999	0.833	0.611	0.97	0.897	0.869
Precision, Filter 2	0.969	0.691	0.918	0.961	0.999	0.831	0.498	0.97	0.897	0.852
Precision, Filter 3	0.969	0.987	0.929	0.994	0.999	0.902	0.989	0.976	0.897	0.915
F1 , Filter 1	**0.949**	0.969	**0.944**	0.985	**0.999**	0.884	0.716	0.966	**0.945**	0.852
F1 , Filter 2	**0.949**	0.804	0.939	0.979	**0.999**	0.883	0.624	0.966	**0.945**	0.861
F1 , Filter 3	**0.949**	**0.985**	**0.944**	**0.996**	**0.999**	**0.917**	**0.991**	**0.973**	**0.945**	0.852

Table 4.14: Comparing Recall, Presion and F1-measure of Normalization Methods based on the corpus in Table 4.2

	BBC	**Economist**	**Zdf**	**Golem**	**Heise**	**Republica**	**Spiegel**	**Telepolis**	**Yahoo**	**Wikipedia**	**Manual**	**Slashdot**
Recall, Filter 1	0.913	0.967	0.963	0.993	0.976	0.995	0.946	0.979	0.954	0.810	0.687	0.399
Recall, Filter 2	0.922	0.967	0.963	0.745	0.965	0.994	0.946	0.979	0.952	0.760	0.690	0.440
Recall, Filter 3	0.890	0.967	0.963	0.999	0.964	0.996	0.941	0.980	0.953	0.787	0.686	0.372
Precision, Filter 1	0.991	0.830	0.880	0.941	0.900	0.872	0.943	0.914	0.948	0.927	0.355	0.208
Precision, Filter 2	0.935	0.732	0.812	0.707	0.830	0.792	0.938	0.914	0.944	0.882	0.356	0.192
Precision, Filter 3	0.991	0.855	0.880	0.989	0.911	0.954	0.974	0.919	0.948	0.927	0.357	0.197
F1 score, Filter 1	**0.939**	0.884	**0.910**	0.965	**0.931**	0.914	0.938	0.930	**0.950**	**0.856**	0.403	**0.248**
F1 score, Filter 2	0.916	0.827	0.871	0.724	0.884	0.865	0.937	0.930	0.948	0.809	0.404	0.239
F1 score, Filter 3	0.922	**0.900**	**0.910**	**0.994**	**0.931**	**0.970**	**0.951**	**0.932**	**0.950**	0.840	**0.404**	0.236

- By looking at the first part of Tables 4.13 and 4.14, it can be observed that Filter 3 has the maximum recall only in 11 web sites among a total number of 22 web sites, while Filter 3 attains the maximum F1 score in 18 web sites.
- In web sites in which the values of recall obtained from Filter 2 or 3 are equal to that of Filter 1, one can conclude that the web site has not contained any hyperlink in its MC. For example, it can be seen on Economics and zdf web sites that the recall is equal for all the three filters.
- When Filter 1 has a recall equal to one in a web site such as Reuters, it can be argued that the web sites certainly include no hyperlink in its MC, thus the other two preprocessors of Filters 2 and 3 have calculated the recall value as one.
- When Filter 2 has a higher recall and a lower precision than the other two filters, it can be concluded that a major part of the extraneous items have been selected as the MC. It is well known that the menus are taken as one of the additional items in the web pages and each item in the menu is usually built by an anchor tag. Therefore, by the application of Filter 2, it would be possible to consider menus as the MC in some of the web sites such as BBC Arabic. However, the value of recall is equal to 0.989 in the web site of BBC Arabic, which is excellent. On the other hand, the value of precision is reported as 0.804 which is rather low and indicates the existence of some words in the final MC which can hardly be taken as a part of MC.

Chapter 5

Headline Extraction

In this chapter we address the problem of identifying the headline of a web article. While most of the headline extraction approaches mainly focus on structural and visual features of the headline (38, 39, 65, 126), here we propose a heuristic and content based method, *TitleFinder*, to identify the headline of a web document based on the content of an article. Given the design of the algorithm it has two main advantages over previous approaches. First, it is an unsupervised approach that does not require training data. Second, it is capable of operating on single documents and is independent of a template-driven layout of the documents. In our evaluation the method shows a very accurate (F1-measure > 0.969) identification of the headlines of 11,218 web documents and outperforms two baseline methods using structural and visual features.

5.1 Related Work

Hu et al. (65) have claimed in 2005 that no special research has concentrated on the problem of headline extraction from HTML files prior to their work. They have also stated that the extraction of the headline from the body of HTML files is not that simple, since various web pages contain different contents and formats. The method proposed by Hu et al., the supervised machine learning approach, has been used to identify and extract the headline. This method benefits from two main phases like many other machine learning methods, namely training phase and extraction phase. The experimental data will be prepared before the first phase and during preprocessing. First, each web document is transformed into several text segments such that each text segment exactly corresponds to a line containing text in the web document. Afterwards, the text segments will be initialized based on specifications such as font size, font weight and position, which are very conspicuous for the headline. Meanwhile, the headline is annotated among the text segments of each web document. Now, a classification model is trained in the first phase according to the training data. This model will determine whether a text segment is a headline or not. However, they have used the classification model as a perceptron with uneven margins during the extraction phase. In the second phase of the algorithm, the extraction phase, text segments of a web document enter the model, while the model allocates a score to every unit. Finally, text segments having the greatest score will be taken as the headline.

Hu et al. (64), in their research on automatic headline extraction, used other models of machine learning such as maximum entropy, maximum entropy marker model and voted perceptron. They concluded that the performance of the perceptron model was the best. Furthermore, the authors of this paper have also assessed the models developed on various domains and in different languages. They finally discovered that the accuracy is not decreased significantly.

Xue et al. (126) have employed supervised machine learning methods to extract the headline from web documents. The methods employed covered SVMs and Conditional Random Fields (CRF). As features in the machine learning models, formatting information and linguistic information have been utilized in this contribution. It is interesting to note that the CRF model is shown to be more successful than SVM in extraction of the headline.

Ibrahim et al. (66) developed an algorithm for automatic extraction of the headline and the main content of news web pages. For doing this, they proposed a supervised machine learning classification technique based on the use of a SVM classifier to extract the desired textual elements. The SVM classifier is trained strictly on structural features to identify the main content and its headline.

Changuel et al. (27) proposed an automatic method for extracting the headline of HTML web documents based on supervised machine learning technique such as Decision Trees and Random Forests. The key point in their contribution is that they employ information in the header of the HTML file in order to obtain labeled training data for title extraction with limited human effort.

Zhang et al. (128) proposed a content-based and domain-independent method for extracting headlines from Chinese research papers using the support vector machine classifier. They claimed their contribution achieved better results than rule based methods and attribute this to using apriori information about the headline's words and the relationship between the headline and the body of the HTML web page.

All the approaches presented so far employ supervised machine learning techniques. Therefore they depend on training data from which they derive typical characteristics of the headline. Given the large variety of web document designs on the web, it is a challenging task to provide an un-biased training set in this setting. A similar problem has already been observed in the related field of main content extraction (36).

An alternative approach is to formalize human experience and domain knowledge into heuristics for solving the task of headline detection. Fan et al. (38, 39) have found that the headline is often annotated by a special HTML tag (H1-H6) and given visual prominence. In this regard, they proposed a two-stage algorithm for characterizing the headline. In the first step, headline candidates are selected based on two of the following criteria: The first criterion states that only those candidates (lines containing text) can be selected as the headline whose horizontal starting position is not greater than that of main text region. Meanwhile, their top position must not be less than the top quarter of the main text region. The second criteria implies that the font size of candidates must not be smaller than that of the main article text. In the second step, a score is calculated for each candidate based on some rules including font size and the position of the candidate. The candidate with the highest score will be selected as headline.

Figure 5.1: Example of web pages with the selected headline

5.2 Problem Setting and Analysis

We already demonstrated the need for identifying and characterizing the headline in HTML web documents. To motivate a content based approach (rather than the structural and visual based approaches of related work), we now look at the problem setting in a more detailed way. Generally speaking, the structure of a web article is comprised of some text paragraphs, some figures or diagrams with their relevant captions, and of course a headline. Besides these main components of news, some additional items come along with the article, which are not related to the main content at all. Moreover, there are some elements which are not accounted as the main article but are somehow related to it. For example the news links located beside the main news directing the user to other pages.

We use the web document in Figure 5.1 to elucidate this issue in a bit more detail. The following parts can be identified in this article taken from the BBC News web site:

- α : Header and main menus of this news page from the BBC web site can be seen in this section which are known as additional items.

- β : This section includes main parts of the news, i.e. the headline, publish date, constituent paragraphs, subtitles, and semi-related useful news β' as well as prominent news tips β''.
- γ : This section involves additional news items which are not closely related to the main news. The following sections are some of them:
  - top stories
  - features and analysis
  - most popular

  It is obvious that most of the subsections contain a series of links which direct users to other pages for getting access to the text of news.
- δ : The browser window's menu bar indicates meta-information of the web document. It shows the text fragment which is encoded in HTML using the content of the title element `<title>...</title>`. This fragment usually provides a good hint of the headline, but typically contains other text as well. In this case, the web site's name is provided as well, other settings add a date or a copyright remark.

5.3 TitleFinder

We now develop our novel and content based method *TitleFinder* for identifying the headline of web articles. As mentioned previously, the position of the headline related to a web document can contribute much to the algorithms, either line-based or block based, which decide to extract the MC from the web document.

The algorithm proposed in this chapter benefits from one of the observations mentioned above: A great proportion of the word tokens inside the headline are similar to the title element. However, the text content between the HTML tags `<title>...</title>` usually contains additional information, e.g. the name of the web site hosting the article, the current date or a copyright notice. This observation is utilized in *TitleFinder* to identify the headline of web articles. It interprets the text in the `<title>` element as a query to be applied to the text fragments of the article's content. This allows for the identification of the most descriptive text fragment for the article. By considering all text nodes in the DOM tree as potential candidates for this text fragment and breaking them down into sentences, *TitleFinder* is independent of structural or visual information.

In more detail, the process behind *TitleFinder* passes through the following phases:

5.3.1 Preprocessing

First, since we do not make use of it, JavaScript code, CSS code, comments and meta tags are removed from the web document. Furthermore, we normalize the source code (86) for line breaks or excessive white space, which allows for easier identification of text fragments and word tokenization in the downstream process. For the same purpose, other characters such as the single quote, double quote, comma and colon are deleted from the content of an HTML file.

5.3.2 Conversion of Text Fragments into Vector Space Representation

Our content based headline identification method maks use of a vector space representation (108) of text fragments. To arrive at this representation we take several steps. After the preprocessing, we extract the text fragments from the source code by considering the text nodes of the DOM tree. For simplicity, we will use the expression *sentence* to distinguish a text fragment from the text of the complete web document. Assuming that N text fragments of the HTML file denote its content, there would be N sentences corresponding to the web document. We tokenize each sentence into words and transform it into a classical vector representation $\overrightarrow{s}_j$:

$$\overrightarrow{s}_j = \{w_{1,j}, w_{2,j}, ..., w_{|V|,j}\} \tag{5.1}$$

The entries $w_{i,j}$ in this vectors correspond to the weight of term $t_i \in V$ in sentence j, wherein $V = \{t_1, t_2, ..., t_{|V|}\}$ is the set of distinctive words of the content of HTML document. In our experiments we considered two very well–established weighting schemes, namely term frequency (*tf*) weights and term frequency–inverse document frequency (*tf_idf*) weights.

5.3.3 Similarity Metrics

In order to identify the headline in a web document, we first obtain the text in the `title` element of an HTML file. Next, we consider this text as a query q. Then, the similarity between the query q and each one of the N sentences is assessed. Based on our observation, the hypothesis behind *TitleFinder* is that the sentence which bears the highest similarity with query q will be the headline. In the vector space framework of *TitleFinder*, we implemented the four following methods to measure the similarity between the query q and each of the N sentences:

- TF: Cosine Similarity based on *tf* weighting Scheme
- TF_ IDF: Cosine Similarity based on *tf_idf* weighting Scheme
- OSM: Overlap Scoring Measure
- Aggregate = TF + TF_ IDF + OSM

The cosine similarity corresponds to measuring the cosine of the angle θ between the query vector $\overrightarrow{q}$ and one of the $\overrightarrow{s}_j$ vectors ($1 \leq j \leq N$) with its value being in the range between 0 and 1. Higher values indicate a higher similarity of the vectors, which is interpreted as a high semantic similarity of the respective text fragments.

OSM, the third similarity metric (78), is a less commonly used similarity metric. It incorporates *tf_idf* weights for the query terms, which is shown by $w_{t,q}$, and *tf* weighting with Euclidean normalization for the sentences, which is shown by $w_{t,s}$. The OSM similarity is provided in the following Formula 5.2:

$$\mathrm{OSM}(\overrightarrow{q}, \overrightarrow{s}_\mathrm{j}) = \sum_{\mathrm{t} \in \overrightarrow{q}} \mathrm{w_{t,q}} \times \mathrm{w_{t,s_j}} \tag{5.2}$$

The fourth and last method aggregates the values of the previous three metrics and is shown in Formula 5.3. It combines values obtained from the three previous techniques for each of N sentences to identify the headline. Thus, it shows similarities to techniques for result list merging.

$$\begin{aligned} \text{Aggregate}(\overrightarrow{q}, \overrightarrow{s}_j) &= \text{Cos}_{\text{tf}}(\overrightarrow{q}, \overrightarrow{s}_j) \\ &+ \text{Cos}_{\text{tf\_idf}}(\overrightarrow{q}, \overrightarrow{s}_j) \\ &+ \text{OSM}(\overrightarrow{q}, \overrightarrow{s}_j) \end{aligned} \tag{5.3}$$

5.4 Experiments

5.4.1 Data sets

For evaluation purposes we use the data sets introduced in (85) (cf. Table 5.1) and (46, 48) (cf. Table 5.2), composed of 2,282 and 8,936, respectively, web pages from different web sites. As we explained in Section 4.1, the first data set is a collection of web documents in Arabic, Farsi, Pashto, and Urdu and the second data set has been established for the evaluation of main content extraction algorithms on Western language documents. The most important point is that the headline in each web page should be as a sub–string of the title element, otherwise *TitleFinder* is not able to extract the headline of web pages. Considering this observation and that the title element in two web sites *Embassy of Germany* and *Manual* has no information about the headline of web page, these web sites are not evaluated in this chapter by *TitleFinder*. For the same reason, we were forced to eliminate 100 web pages in the BBC web site and evaluate only 900 web pages (compare Table 4.2 and Table 5.2).

Table 5.1: Evaluation corpus of 2,282 web pages

web site	URL	Size	Languages
Ahram	www.jamejamonline.ir/	188	Arabic
BBC	www.bbc.co.uk/arabic/	252	Arabic
BBC	www.bbc.co.uk/pashto/	368	Pashto
BBC	www.bbc.co.uk/persian/	598	Farsi
BBC	www.bbc.co.uk/urdu/	213	Urdu
Hamshahri	hamshahrionline.ir/	375	Farsi
Jame Jam	www.jamejamonline.ir/	137	Farsi
Reuters	ara.reuters.com/	116	Arabic
Wiki	fa.wikipedia.org/	35	Farsi

5.4.2 Evaluation Methodology

In order to calculate the accuracy of any headline extraction method, it is necessary to provide a manually crafted *gold standard* for the headline of all HTML files. Both corpora mentioned in Section 5.4.1 provide such a gold standard. For the purpose of evaluation, the output of an headline extraction algorithm is checked for a precise match with the gold standard headline of the corresponding HTML document.

Table 5.2: Evaluation corpus of 8,936 web pages

web site	URL	Size	Languages
BBC	www.bbc.co.uk/	900	English
Economist	www.economist.com/	250	English
Golem	golem.de/	1,000	German
Heise	www.heise.de/	1,000	German
Repubblica	www.repubblica.it/	1,000	Italian
Slashdot	slashdot.org/	364	English
Spiegel	www.spiegel.de/	1,000	German
Telepolis	www.telepolis.de/	1,000	German
Wiki	en.wikipedia.org/	1,000	English
Yahoo	news.yahoo.com/	1,000	English
Zdf	www.heute.de/	422	German

In our experiments, we follow other approaches and employ recall, precision and F1 as performance metrics (64). The metrics are defined in Formula 5.4 and 5.5, in which we define A to be the number of headlines correctly identified as headlines, B as the number of other elements misclassified as headlines and C to be the number of headlines erroneously non–identified as such (cf. also Table 5.3).

$$\text{Recall} = \frac{\mathsf{A}}{\mathsf{A}+\mathsf{C}} \qquad \text{Precision} = \frac{\mathsf{A}}{\mathsf{A}+\mathsf{B}} \tag{5.4}$$

$$\text{F1} - \text{measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5}$$

Table 5.3: Contingence table with regard to headline extraction

	Is headline	**Is not headline**
Extracted	A	B
Not Extracted	C	D

5.4.3 Baselines

As baseline approaches we consider the two methods. First of all, we evaluate the naive approach of assuming the content of the `title` element to be equivalent to the headline. As mentioned previously, extracting the headline of a web document using this method is not reliable because there is some additional information in the `title` element of the web document not related to the headline.

Therefore, we employ a more sophisticated baseline. It follows the observation that the headline often corresponds to the first encounter of the largest font size in the document (27, 65). Thus, we tested the accuracy of such a visual feature–based baseline method. We

implemented a method searching for the first encounter of the highest order headline element in web document.

5.4.4 Results

Table 5.4 presents the results of our experiments showing the average F1-measures and the processing speed (KB/s) of *TitleFinder* and the baseline approaches on the two data sets. On the left side of Table 5.4, the columns labelled TF, TF-IDF, OSM and Aggregate list the accuracy of the different similarity metrics implemented in the *TitleFinder* method for identifying the headline in the HTML files. In addition, the columns labeled Largest Font and `title` shows the accuracy of the baseline approaches in headline extraction from an HTML file. All six columns on the right side of the table demonstrate the processing speed (KB/s) of *TitleFinder* on our data sets. The columns are labeled TF, TF-IDF, OSM and Aggregate, as well as the baseline approaches including the columns labeled Largest Font and `title`. The last row in the table aggregates the values using a macro-average F1-value.

Regarding the baselines, it can be observed that they either always succeed or always fail on the documents taken from one web site. This is not surprising since most modern web sites are template driven and exhibit the same or at least a very similar structure. The naive approach of simply using the `title` element's content fails in most cases (18 out of 20 web sites).

Also the more sophisticated baseline of using the first occurrence of the largest font-size fails quite often (8 out of 20 web sites). The two assumptions considered in it, namely "the headline has the largest font size" and "the headline is the first occurrence of the largest font size", seem not to hold for extracting the headline from HTML documents. We observed that the headline quite often is not contained in the semantically correct elements `h1` to `h6`, but rather in `<div>` and/or `<span>` tags. This technique is used, for instance, on the web sites BBC, Jame jam, and Ahram. Furthermore, the documents from Slashdot, Spiegal, and ZDF web sites have employed `<h1>` tag for displaying the web site's name which is not a headline. An important point to bear in mind is that the extraction of the headline assumes a certain consistency of the underlying document template. It always selects the same first largest headline element in a document. In Table 5.4, there are five entities employing the same web site template: BBC, BBC Arabic, BBC Pashto, BBC Farsi, and BBC Urdu. Looking for explanations of the failure of the baseline algorithm on the BBC Arabic web site, we noticed that—unlike the other BBC websites—it did not encode the headline in the first `<h1>` tag. Hence, we can conclude that trusting a web site template to extract the headline of a web page sometimes misleads us to incorrect results.

Considering the performance of *TitleFinder*, we can make the following observations in Table 5.4:

- In 14 out of the 20 considered web sites, the *TitleFinder* algorithm shows an F1-measure value of 1.0 which is indicative of the high accuracy of this method. Furthermore, in 17 out of 20, our proposed approach demonstrates an F1-measure value greater than or equal 0.997, which is a considerable accuracy.

Table 5.4: Evaluation results of data set in Tables 5.1 and 5.2 based on F1-measure and processing speed (KB/s)

Website	F1-measure					Processing Speed				
	TitleFinder			Baselines		TitleFinder			Baselines	
	TF	**TF-IDF**	**OSM**	**Aggr.**	**Largest Font**	`title`	**TF**	**TF-IDF**	**OSM**	**Aggr.**	**Largest Font**	`title`
BBC	1.0	1.0	1.0	1.0	0.0	0.0	157.6	118.7	119.3	118.6	840.3	1025.3
Economist	1.0	1.0	1.0	1.0	0.0	0.0	168.6	85.3	85.1	84.8	529.6	605.6
Golem	1.0	1.0	1.0	1.0	1.0	0.0	265.6	143.0	143.4	142.4	1031.7	1721.1
Heise	0.999	1.0	0.998	1.0	1.0	0.0	197.9	100.9	101.3	100.5	1027.0	1692.1
Republica	0.999	0.999	0.999	0.999	1.0	0.0	295.1	132.8	133.2	132.4	882.9	1309.3
Slashdot	1.0	1.0	1.0	1.0	0.0	0.0	122.9	49.7	49.7	49.2	834.3	2030.4
Spiegel	0.977	0.995	0.996	0.995	0.0	0.0	196.7	92.1	92.2	91.2	662.2	948.1
Telepolis	1.0	1.0	1.0	1.0	1.0	0.0	200.9	61.3	61.2	61.0	1232.5	2323.9
Wiki	0.861	0.805	0.834	0.828	1.0	0.0	237.0	95.1	94.9	94.1	791.5	932.6
Yahoo	1.0	1.0	1.0	1.0	1.0	0.0	256.9	126.5	126.3	125.7	786.4	1033.4
Zdf	1.0	1.0	1.0	1.0	0.0	0.0	221.7	92.7	92.9	92.2	933.6	1838.9
Ahram	0.952	1.0	1.0	1.0	0.0	0.0	409.7	322.2	321.7	319.8	1094.4	1651.7
BBC Arabic	1.0	1.0	1.0	1.0	0.0	0.0	304.9	139.5	140.9	139.4	1153.5	1032.9
BBC Pashto	1.0	0.997	1.0	1.0	1.0	0.0	139.7	115.3	116.4	115.9	821.6	660.5
BBC Farsi	0.998	0.997	0.998	0.998	1.0	0.0	319.2	144.4	144.8	144.0	1221.4	1695.2
BBC Urdu	1.0	1.0	1.0	1.0	1.0	0.0	224.6	106.7	107.2	106.6	840.9	753.2
Hamshahri	1.0	1.0	1.0	1.0	1.0	1.0	268.2	128.4	128.7	128.0	1184.1	2121.9
Jame Jam	1.0	1.0	1.0	1.0	0.0	1.0	404.2	230.8	231.9	229.9	1241.6	1856.6
Reuters	1.0	1.0	1.0	1.0	1.0	0.0	162.3	116.1	116.5	116.9	291.3	288.6
Wiki	1.0	0.706	0.559	0.735	1.0	0.0	397.1	223.1	223.2	221.8	1285.1	1993.5
Macro-Avg.	**0.989**	0.975	0.969	0.978	0.6	0.1	**247.5**	131.2	131.5	130.7	934.3	1375.8

- None of the four *TitleFinder* variations employing different similarity metrics has failed completely on our data sets as observed, for instance, for the two baseline approaches with a F1-measure of 0. This can be explained by the algorithm to be agnostic towards structural properties of the documents and therefore being more independent of template properties.

- The average F1-measure for all variants of the *TitleFinder* is no lower than 0.969. This demonstrates a general stability of the approach.

- Considering the Macro-Average F1-measure, it can be seen that although *TitleFinder* based on TF does not show the F1-measure equal to 1.0 in all cases, its reported Macro-Average F1-measure value, 0.989 , is greater than the Macro-Average F1-measure value of the other *TitleFinder* implementations. It seems that *TitleFinder* based on the TF weighting scheme can be an appropriate method for the extraction of headlines from web pages, since its Macro-Average F1-measure value, 0.989, is relatively high, although the improvement of TF over the other *TitleFinder* methods is not statistically significant.

- For the web sites Republica, Spiegel, Wiki, and BBC Farsi, none of *TitleFinder* methods were able to find an average F1-measure value of 1.0. This behaviour is mainly attributed to an error which occurs when the total number of headline tokens is smaller than that of tokens related to name and specifications of the web site cited in the title element. In this case, the *TitleFinder* starts from a sub-optimal candidate headline to query the documents content elements and tends to extract name or specifications of the web site instead of its headline.

When comparing the processing time of the approaches, *TitleFinder* is significantly slower than the structural approaches. This can be attributed to the overhead of analyzing all text elements and building their vector representations. In addition, identifying and extracting all content tokens of HTML document using an HTML parser are far more time-consuming. However, so far we did not tune our algorithm for fast processing and there is still potential for streamlining the programs written.

5.5 Summary

In this chapter we proposed *TitleFinder*, a content based method for headline extraction from web pages. The main idea is to use the content of the title element as a candidate headline and then compute the similarity between this candidate and all text fragments in the body of the HTML file. We implemented *TitleFinder* using four different similarity metrics. The results obtained from implementation of four variations of the algorithms on 11,218 web pages indicated a high accuracy of our method. In many cases we observed an perfect extraction-performance with an F1-value of 1.0.

In future work, we aim to overcome some minor weaknesses for particular settings observed in evaluation data. One issue is to improve the quality of the initial selection of the candidate headline. As one solution, name and specifications of a web page appearing in the title element as well as other tokens present in the title element can be searched in a main content. Since

name and specifications related to a web page are not present in the main content related to a web page, it seems promising to remove tokens which are present in the title element but not in the main content of a web page. Also, we will combine our headline detection algorithms with content extraction algorithms to enhance accuracy of the content extraction methods.

Chapter 6

Applications of Main Content Extraction

The extraction of MC from web pages and weblogs has numerous applications which are addressed to some extent in the first chapter of this thesis. Some of these applications will be briefly discussed here:

- Main content extraction can be applied in scenarios where a reduction of a document to its main content is advantageous, e.g. on devices that have limited storage or bandwidth capacity, such as mobile phones, screen readers, etc.

- Main content extraction can be considered as a preprocessing step for text mining and web information retrieval.

- The identification of the MC is beneficial for web search engines: when crawling and indexing the web, knowing the actual main content of each web page can be exploited for the purpose of determining more precise and descriptive index terms for the document.

- Traditionally, building text corpora is a very expensive and time–consuming process. By automatically downloading textual data from the web, extremely large corpora can be built in a short period, at relatively low cost. Therefore, the idea of *Web as Corpus* has been very attractive for many researchers in Natural Language Processing and related areas (113).

In addition to the mentioned items, extraction of MC from web pages have another applications which will be discussed in the future sections of this chapter. The last section of this chapter deals with one of the research works implemented by the author of this thesis with collaboration of other colleagues. The main goal of Section 6.6 is to provide one of the applications of IR. However, the presented methodology can be combined with MC extraction methods in the future in order to investigate trends in different fields using web pages and weblogs as will be discussed in Section 6.5.

6.1 Reading News Web sites for Visually Impaired

It is well known that there are software products which can read a text file for a user. Thus, the text in the MC extracted from web pages can be given to these products to read the text for blind people (125). Most web pages of news web sites including BBC and Spiegel contain a number of related news inside each web page, which make it possible to access news relevant to the current news. Thus, once links of the related news of each web page are identified, the MC of the related news can be read after extraction.

6.2 Removing Advertisements from Web Pages

With rapid growth of advertisements in web sites and web pages, it is seen that a major part of a news web page is allocated to advertisement and just a small part of that web page contains the news. Many individuals certainly feel dissatisfied with this situation and may prefer to see only the MC on the news web pages. This could be made possible for the users by the combination of MC extraction algorithms and web browsers. Thus, only the MC in the news web page will be seen in the web browser (94).

6.3 Main Content Extraction and Opinion Mining

Opinion mining is another application of extracting MC from web pages and especially MC extraction from posts in weblogs. There are millions of weblogs in the web space with hundreds of new weblogs being created every day. Each of these weblogs usually contains several posts. By extraction of the posts in weblogs and the MC in these posts, it will be possible to explore ideas and opinions of various people among millions of posts extracted from weblogs in the virtual space of the web (59).

6.4 Main Content Extraction and Question Answering

One other very useful application of MC extraction from web pages and weblogs can be observed in Question Answering (QA) systems. QA systems can be divided into two general categories below:

- Not Web Based Question Answering Systems (NWBQAS),
- Web Based Question Answering Systems (WBQAS), (37, 52)

In NWBQAS, a great number of documents, which will be used to extract the answers, are first stored in the system. Then, the user puts forward a question. At this step, the QA system provides the user with the document which contains the best answer for this question. Indeed, there are some additional parts in the architecture of a QA system (102), though these are not going to be discussed here. On the other hand, WBQAS does not need to prepare the documents which contain the answers. Thus time, money, labor force, etc. can be stored. These systems try to find answers for the questions put forward by the user from existing documents in the web. A part of these documents can surely be web pages and

weblogs which exist in the web. Therefore, MC extraction algorithms can be combined with the QA systems to create WBQAS (96).

6.5 Trend Analysis

Trend analysis is one of the most interesting issues in most scientific and research fields. However, manufacturers and suppliers of products benefit much from trend analysis because of the numerous applications in product survey, customer relationship and marketing. Based on the statistics of the "Technorati" web site which is a blog search engine, about 1.2 million new entries are added to this web site every day with its volume being duplicated in six months. Thus, it can be said that the blogsphere is an appropriate source for studying the trend of many subjects (31).

6.6 Revealing Trends Based on Defined Queries in Biological Publications Using Cosine Similarity

Massive volumes of scientific data are being produced every day by scientists in all disciplines. Similarly, the number of scientific journals is growing, with many new sub-disciplines launching their own journals. For example, SciVersc Scopus stores 18,500 peer-reviewed journals alone.

To the best of our knowledge, fields and interests of scientific journals are traditionally specified by their titles, aims and scopes. Moreover, journals are categorized and clustered by their scopes and aims to different groups using information stored in databases. However, the core interest of journals may change and evolve over time.

From the time point of view, the appearance and disappearance of the core scope is a dynamic process, rather than a purely random process, but with more underlying principles. Gradual or abrupt changes in the core scope of a scientific journal can be explained by internal and external causes. Typical internal causes include the changing of the editorial board or publisher of a journal. External causes include scientific breakthroughs and opening new directions and sub-disciplines or themes. Furthermore, it is quite possible that different core topics converge or diverge over time periods among or within journals. Understanding the dynamics of a research theme is essential especially for analysts and decision makers for being able to identify emerging and disappearing trends and topics or rapid changes in the body of scientific knowledge. In the rest of this section, an approach, *TrendFinder*, will be described to analyze the trends of several given queries in collected data sets. For doing this, first we represent Abstracts and queries based on a vector space model and afterward by using Cosine similarity metric we compute the similarities between selected query and a group of Abstracts.

6.6.1 Related Work

One of the first approaches to analyze trends and discover emerging patterns from text documents was presented by Feldman et al. (40). They implemented a new system for

Journal	Num. of Abs.	From(year)	To(year)
Conservation Biology	4004	1987	2010
Ecology	10004	1964	2010
The American Naturalist	5002	1960	2010

Table 6.1: Three Journals with Number of Abstracts and Published Date

Tag	Description
TY	Type of Text, Proceeding or Book or Journal
AU	Author
TI	Title
JO	Name of Journal
VL	Volume
IS	Issue
PB	Publisher
SN	Serial Number
UR	URL
SP	Start Page
EP	End Page
PY	Published Date
AB	Abstract

Table 6.2: List of Tags in one of Selected Raw Paper

Knowledge Discovery in Text (KDT), in which documents are labeled by analyzing the co-occurrence frequencies of the expert-defined keywords.

Temporal Text Mining (TTM) is another line of research related to the present work. Mei et al. (82) introduced a probabilistic approach to discover and summarize the evolutionary patterns of themes in a text stream. Another approach to analyze the temporal trends of a given topic in a time stamped document set which works based on time series segmentation is described by Chen et al. (29). They consider topics containing multiple keywords and use a fuzzy set based method to compute a numeric value in order to measure the relevance of a document set to a given topic. The measure of relevance is then used to assign a discrepancy score to a segmentation of time period associated with the document set. The discrepancy score of the segmentation represents the likelihood of the topic across all segments in segmentation.

Naveed et al. (92) proposed a probabilistic approach, ATTention, which analyzes the evolution of users' interests with respect to produced content over time using the Bayesian modelling of relationships between authors, latent topics and temporal information.

6.6.2 Data sets, Queries

As an evaluation data set, some 19,010 papers were collected from three different journals, namely "Conservation Biology", "Ecology", and "The American Naturalist" (see Table 6.1). These three journals have been selected for this analysis because of free accessible data available for a long period of time. In the second column of Table 6.1, the numbers of Abstracts are given for each journal which are 4004, 10004, and 5002, respectively. Columns 3 and 4 in Table 6.1 represent the period of selected journals, for example, the period of "Ecology" was from 1964 to 2010.

Listing 6.1 represents one selected original paper. The description of each tag is explained in Table 6.2. For fast retrieval of data sets, the preprocessor of *TrendFinder* converts all collected original papers with a different structure to a standard XML file format (Listing 6.2). Table 6.3 summarizes four queries with their expert-knowledge keywords, which are selected by the expert of the relevant field. The names of queries are shown in left column while all keywords of each query are listed in the right column. It should be noted that the numbers of keywords in the four queries were not exactly the same.

Listing 6.1: Sample of Original Paper in Conservation Biology Category

```
TY  - JOUR
AU  - MEINE, CURT
AU  - SOUL , MICHAEL
AU  - NOSS, REED F.
TI  - ‘‘A Mission-Driven Discipline’’: the Growth of Conservation Biology
TI  - ‘‘Una Disciplina Dirigida por una Misi n ’’: el Crecimiento de Conservation Biology
JO  - Conservation Biology
VL  - 20
IS  - 3
PB  - Blackwell Publishing Inc
SN  - 1523-1739
UR  - http://dx.doi.org/10.1111/j.1523-1739.2006.00449.x
DO  - 10.1111/j.1523-1739.2006.00449.x
SP  - 631
EP  - 651
PY  - 2006
AB  - Abstract: Conservation biology emerged in the mid-1980s, drawing on established
        disciplines and integrating them in pursuit of a coherent goal: the protection
        and perpetuation of the Earth’s biological diversity.
ER  -
```

Listing 6.2: Sample of an XML File

```
<?xml version=‘‘1.0’’ encoding="utf-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<research>
<journal>
   Wiley
</journal>
<title>
   ‘‘A Mission-Driven Discipline’’: the Growth of Conservation Biology
</title>
<date>
   2006
</date>
<abstract>
   Abstract: Conservation biology emerged in the mid-1980s, drawing on established
   disciplines and integrating them in pursuit of a coherent goal: the protection and
   perpetuation of the Earth’s␣biological␣diversity.
</abstract>
</research>
```

6.6.3 TrendFinder

To explain how *TrendFinder* calculates the content-based trends over selected data sets and defined queries, the provided algorithm was divided into the following three steps:

Queries	Keywords
Vector1-Biodiversity	diversity biodiversity species richness abundance abundant gradient geography geographical biogeography interaction interact abiotic biotic dispersal functiona functional environment
Vector2-Evolution	evolution macroevolution extinction extinct hybridization divergence divergent sexual sex population niche phylogeny phylogenetics phylogeography trait adaptive adaptation diversification speciation sympatric sympatrically speciation gene flow genetic drift mutation natural selection reproductive isolation
Vector3-Conservation	protected areas climate change global warming conservation conserve reserve planning management endemic endemism hotspot hot spot fragmentation ecosystem habitat change land-use deforestation bioindicator
Vector4-Genome	genomics metagenomics genome gene genetics system biology informatic bioinformatic mining sequence sequencing metadata

Table 6.3: List of Four Queries with Selected Keywords

6.6.3.1 Preprocessing

A common problem for processing papers taken from different journals is that each journal has its own structure, so the information we need can not be detected accurately. In Section 6.6.2, it was suggested to transform all original papers with a different structure to a standard XML format using a preprocessing step. By this approach, it was not required to rewrite the core section of *TrendFinder*.

6.6.3.2 Representing Documents Using a Vector Space Model

We know the following notation represents a vector space model of a document:

$$\vec{\mathbf{d}}_j = (w_{1,j}, w_{2,j}, ..., w_{t,j}) \tag{6.1}$$

Each dimension of this vector corresponds to a unique term in which its value is greater than zero if it occurs in the document. To compute the values of the terms, named weights, there are several different ways. Two of most used weights are tf and *tf-idf*. In this section, the first schemes, tf, will be used for calculating similarity (108).

As discussed in Section 6.6.2, there are three representations of the journal documents and four representations of the queries. Hence, it is possible to consider 12 representations of documents and queries based on the VSM, named REP1 to REP12. Each of the four sequential representations i.e. REP1-REP4, REP5-REP8, and REP9-REP12, will be used for computing Cosine similarities and drawing similarity graphs of each journal. For simplicity, the process of building REP1 has just been explained here.

The basic idea of a VSM is depicted in Figure 6.1 (78). The main hypothesis is that each query has m terms, so a dictionary of m terms is kept in the first column of Figure 6.1 which will be sorted alphabetically. On the other side, the Abstract of each paper can be viewed as a vector with one component to each term in the dictionary, along with a weight w for each component that could be between one and zero. For dictionary terms which do not occur

in an Abstract, this weight is zero and for dictionary terms that occur in an Abstract, this weight is greater than zero. For simplicity, it has been supposed that each Abstract has a unique serial number, known as the Abstract identifier (ID). If there are n Abstracts in each journal, then all n vectors corresponding to n Abstracts will be kept in the VSM, starting from the second column to column $n + 1$.

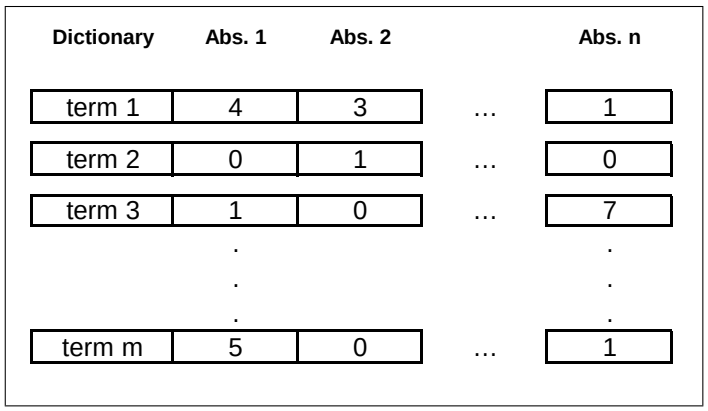


Figure 6.1: Example of a Vector Space Model

6.6.3.3 Cosine Similarity

In order to compute the Cosine similarities between one query and all n Abstracts of a specific journal, the following notations must be considered:

- $\overrightarrow{V}(a_i) = \{w_{1,i}, w_{2,i}, ..., w_{m,i}\}, 1 \leq i \leq n$: the vector derived from Abstract a_i, with one component in the vector for each dictionary term.
- $\overrightarrow{V}(q_j) = \{w_{1,j}, w_{2,j}, ..., w_{m,j}\}, 1 \leq j \leq 4$: the vector derived from query q_j, the components of this query make up the dictionary in the first column of VSM.

Now, Formula 6.2 calculates the Cosine similarity between the query vector, q_j, and a sample Abstract vector, a_i, as a measure for the score of the Abstract in that query.

$$\text{score}(\text{q}_\text{j}, \text{a}_\text{i}) = \frac{\overrightarrow{V}(\text{q}_\text{j}).\overrightarrow{V}(\text{a}_\text{i})}{|\overrightarrow{V}(\text{q}_\text{j})||\overrightarrow{V}(\text{a}_\text{i})|} \tag{6.2}$$

Table 6.4 demonstrates the Cosine similarities for four queries and all n Abstracts of "Conservation Biology", as defined in this section considering the published dates. After having calculated the Cosine similarities between selected query and all Abstracts in a specific year, for example 1987, the averages of all these scores are saved in Table 6.4. Due to the fact that the Cosine similarity will range from 0 to 1, each score in this table is multiplied by 1000 for the purpose of simplicity. For example, the Cosine similarity score between first query and all published papers in year 2000 is 0.727, while that of query four and published papers in 2010 is 0.071.

Year	Query 1	Query 2	Query 3	Query 4
1987	350	392	567	98
1988	455	336	491	84
1989	580	382	481	66
1990	530	377	525	122
1991	458	376	565	64
1992	528	379	605	58
1993	555	300	512	47
1994	581	380	587	40
1995	687	374	570	63
1996	596	376	538	79
1997	705	393	556	63
1998	502	212	341	48
1999	706	360	583	60
2000	727	281	593	62
2001	572	299	628	60
2002	711	332	721	72
2003	800	252	635	45
2004	716	261	600	52
2005	760	264	772	72
2006	614	304	739	79
2007	754	281	782	59
2008	763	242	808	36
2009	775	240	746	62
2010	674	236	863	71

Table 6.4: Similarity Values in Conservation Biology Journal (Multiplied by 1000)

6.6.4 Results

The features of *TrendFinder* with a case study on pioneer journals in the field of ecology and evolution including "The American Naturalist", "Ecology", and "Conservation Biology" have been demonstrated. Aims and scopes of the two first journals are evolution and ecology, respectively. "Conservation Biology", however, focuses on biodiversity conservation. The main emphasis is only on gradual or abrupt changes in core scope of the journals. The dataset included 19,010 abstracts during a 51 year period of time (1960 - 2010) for "The American Naturalist", 47 years for "Ecology" and 24 years (1987 - 2010) for "Conservation Biology". Four vectors of words were designed as four queries. These queries contained 17, 31, 23, and 13 expert-knowledge keywords, respectively, selected by the expert of the relevant field. These keys represented a certain subtopic within the science of ecology and evolution, including "biodiversity", "evolution", "conservation", and "genetics". These subtopics (scopes) are somehow related to the missions of the selected journals. In other words, except for "genetics" each vector represents one of the scopes declared by journals. Keywords were unique for the respective vector and were not repeated in other vectors. The results, shown in Figures 6.2, 6.3, and 6.4 reveal some interesting points:

1. One vector alone may not explain the scope and mission of a journal through time. "The American Naturalists" and "Conservation Biology", for example, show close scores for two subtopics across the time. In contrast, the biodiversity vector differs significantly from other vectors of the"Ecology" journal.

2. All four subtopics remain active for all three journals. No vector has decreased to zero,

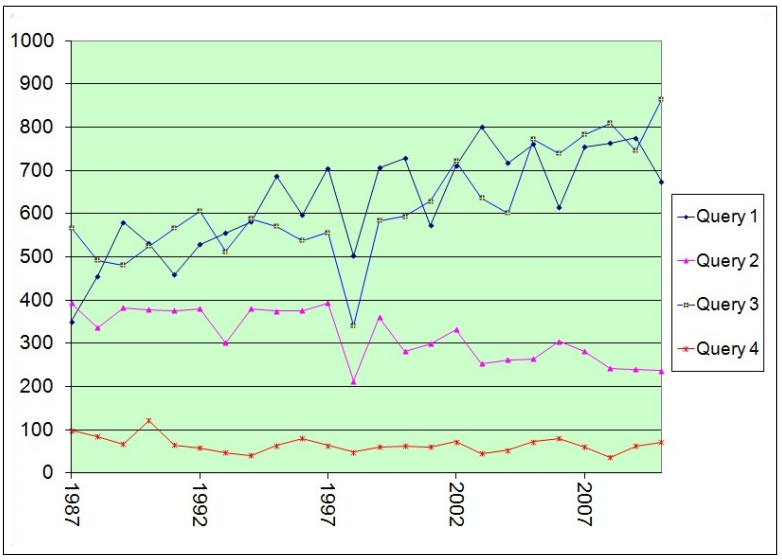


Figure 6.2: Trends Over Conservation Biology Journal

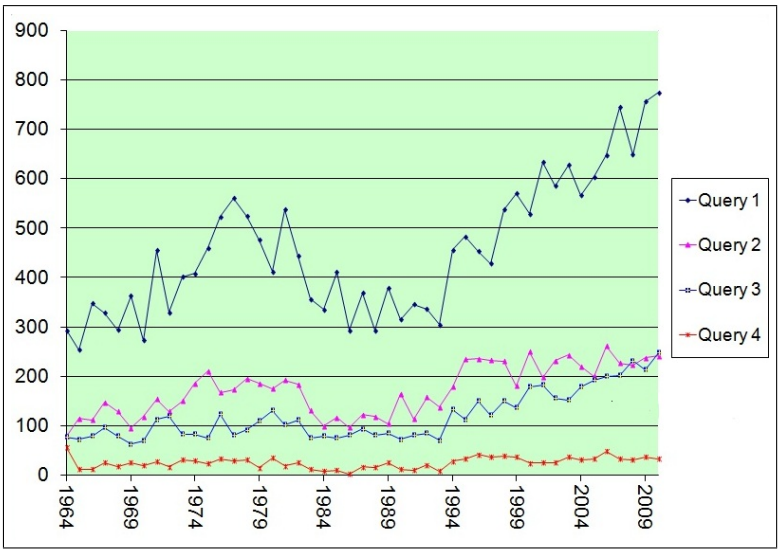


Figure 6.3: Trends Over Ecology Journal

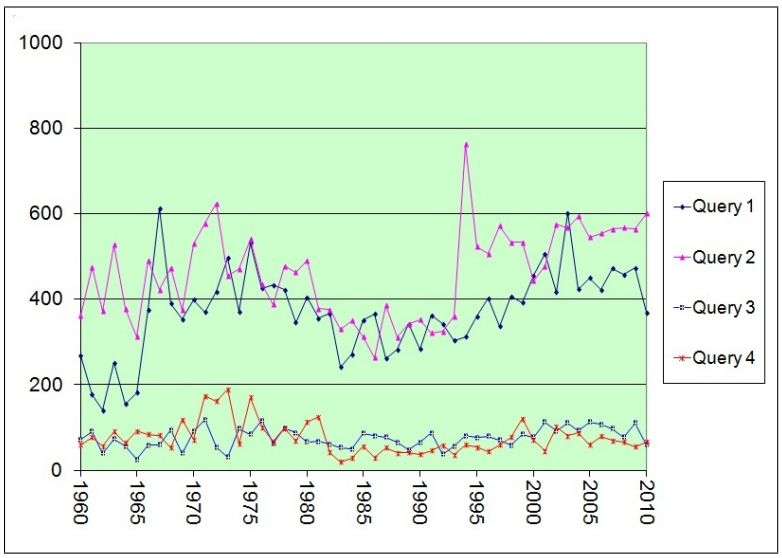


Figure 6.4: Trends Over The American Naturilist Journal

indicating that the journals, although they define a narrow mission, cover additional subtopics.

3. Subtopics may dramatically change during a short period of time or very slowly during a long period of time. Some subtopics are growing rapidly for one journal and decreasing for another journal, which can be attributed to the growing interest on specific topics. For example, the biodiversity vector increased simultaneously for both "Ecology" and "The American Naturalist" journal during the sixties and seventies, while the vector decreased for both during eighties and once again increased sharply since the nineties only for "Ecology". In "Conservation Biology", values of two biodiversity and conservation vectors have experienced an increase since 1987. Evolution vector, however, reveals a gradual continuous decrease. Figures 6.2, 6.3, and 6.4 show that trends of subtopics vary with time, although the rank or importance of the subtopics remain almost the same.

4. Three examined journals do not show a substantial shift in their major aims and missions. At the small time scale, however, there are repeating shifts between priorities. In "Conservation Biology", for example, one of two biodiversity and conservation vectors is considered as the prime vector alternatively.

6.6.5 Summary

This chapter concentrates on the study of trend changes and their applications on web scientific literature search. More specifically, we provided an understanding for the dynamic and structural characteristics of the trends within different scientific journals and we demonstrated the potentiality of a new and rather easy approach to the visualization and analysis of trends of certain topics within a specific web source (scientific journal). This approach

can easily be applied for any other types of web sources. Nevertheless, the current version of *TrendFinder* is only for text based sources, developed to test the concept. We need to develop it in some ways. For example, it would be interesting to know how the similarity values are related with the "quality" of keywords selected by different experts or even different algorithms (68). Furthermore, it is important to estimate and show how often abstracts contain no keyword belonging to the list of preselected keywords.

In future studies, we are going to investigate how topics (vectors) interact over time. Scientific topics are developed and expanded through time and are repeatedly broken or diverged into small, new emerging topics. These new topics, however, can connect at some points during their evolution and development. It would be important to know how often a scientific topic (a field of work) breaks or diverges to new topics and how often and when these new topics emerge. These trends can be also recognized from the scientific literature. We will furthermore focus on various practical issues in order to detect emerging trends and abrupt changes in transient research fronts. Detecting and understanding emerging trends and abrupt change is important because it can significantly improve the ability of the scientists to deal with changes in a timely manner.

Chapter 7

Conclusions and Future Directions

For this thesis we performed research on a number of topics related to information retrieval from web documents. In particular, navel solutions in the field of main content extraction from HTML documents have been studied, developed, and evaluated. In this chapter we summarize the contributions and formulate the conclusion which can be drawn from the results. Afterwards we will take a look at possible directions in which to drive future courses.

7.1 Conclusions

The first part of this work, i.e. chapters one to four, has been focusing on the "main content extraction" from web documents. Our text started by outlining the context of my research in Chapter 1, which was expanded in Chapter 2, where we have discussed four basic topics, i.e. :

- Understanding Text and Web Page Documents
- Information Retrieval
- Content Extraction
- R2L Languages, Unicode, and UTF-8 Encoding Form

As we have compared our novel main content extraction algorithms with previous approaches in this field, a comprehensive survey of content extraction methods has been addressed in Chapter 3. All main content extraction methods have been classified by several aspects including:

- Single Document based Approaches vs. Multi Document Template Detection Approaches
- Stand-alone vs. Integrated Approaches
- Heuristic Techniques vs. Machine Learning Approaches
- Methods Based on DOM Tree Structure vs. Methods Based on HTML Source Code Elements

These could give researchers the opportunity to concentrate on a special category or even compare all categories with each other. in Chapter 4, we introduced two valuable data sets and evaluation methodology which can be used for the evaluation of main content extraction algorithms. Afterward, a novel main content extraction algorithm, R2L, and its three extensions DANA, DANAg, and AdDANAg have been addressed in this chapter. In the following we summarize all these approaches:

- The first proposed algorithm, R2L, very accurately extracts the main content from web documents with an F1 score > 0.929. This algorithm has two further technical advantages: 1) It is DOM tree and HTML-format independent; therefore, errors or non-standard compliant HTML documents do not pose a problem. 2) We do not need to use a parser for our algorithm. This improves runtime efficiency over many of the previous MCE methods which employed the DOM tree structure or used other output of HTML parsers for their purpose. On the contrary, the R2L algorithm is not able to achieve an F1 score closer to 1 in the case of documents where there are some Non-R2L characters among words in the main content area.

- To overcome the problem of R2L, we introduced DANA which feeds entire lines of an HTML file with an outline of the identified MC to an HTML parser. The output of our parser is exactly the main content of selected regions highlighted in the third phase of DANA. The first extension of R2L determines the main content with previously unseen accuracy. Achieving an average F1 score > 0.935 on the test corpus used in this thesis, it outperforms all previous methods. Also, DANA succeeded to achieve an F1 score greater than 0.96 on over six web sites and a perfect value of 1 on BBC Urdu.

- The second extension of R2L, DANAg, is a language-independent version of DANA, with considerable effectiveness. Results show that DANAg determines the main content with high accuracy on many standard data sets. Achieving an average F1 score > 0.90 on the test corpora used in this thesis, it outperforms the state of the art methods in MCE.

- AdDANAg, the third extension of R2L, is a combination and variation of DANAg and the preprocessing of ACCB with considerable effectiveness. Results show AdDANAg determines the main content with high accuracy on many web documents. AdDANAg shows a previously unseen excellent performance, especially on the difficult to handle hyperlink rich web documents.

  In comparison to DANAg, AdDANAg does not show major drawbacks. Furthermore, AdDANAg and DANAg deliver the best results for most documents. When focusing on the set of Wikipedia web pages, which have been observed to be extremely difficult for main content, we can observe that AdDANAg clearly outperforms DANAg and all other approaches. The overall average F1 score for both DANAg and AdDANAg are 0.8099 and 0.8284, respectively.

The subsequently research activity, i.e. Chapter 5, has been focused on the study of the "Headline Extraction" from HTML documents. In this chapter, we proposed *TitleFinder*, a content based method for headline extraction from web pages. The main idea is to use

the content of the title element as a candidate headline and then compute the similarity between this candidate and all text fragments in the body of the HTML file. We implemented *TitleFinder*, using four different similarity metrics. The results obtained from implementation of four variations of the algorithms on 11,218 web pages indicated a high accuracy of our method. In many cases we observed a perfect extraction-performance with an F1-value of 1.0.

In Chapter 6, the application of main content extraction from HTML documents and weblogs have been discussed. In addition, we have concentrated on the study of trend changes and their applications on web scientific literature search. We know, extracting valuable information in terms of number and content of published papers in any field of research will simplify decision making for future researches and investments. As a result, a novel and simple text mining approach, called *TrendFinder*, has been developed in this chapter to reveal the content-based trends of expert-defined queries in selected biological published papers during the last five decades. Therefore, in order to evaluate the results, three different data sets, including a total number of 19,010 papers, were collected and four vectors of selected keywords were considered as the four queries. In order to show the trend between each query and the Abstract of each paper, Cosine similarity method was used by *TrendFinder*. Afterwards, three diagrams demonstrated the content-based trends of the four defined queries on the three provided data sets.

7.2 Future Directions

Considering the anarchic structure of web pages which was mentioned in the First and Second Chapters of this thesis and taking into consideration the ambiguity of main content location in web pages, the issue of main content extraction from web pages is still interesting for researchers. In other words, the researchers try to implement the algorithms which extract the main content in web pages at a great accuracy. Some novel ideas are addressed here which can surely be useful in main content extraction from web pages.

7.2.1 Identification of Upper and Lower Bound of Main Content Area

As was mentioned before in Section 4.3, the R2L Algorithm (in addition to DANA, DANAg and AdDANAg) in its third phase, i.e. what was discussed in Section 4.3.4, looks for the first area in the web page where the number of content characters is more than that of code characters. This area can be comprised of just one line or several lines. Having initially found that area of the web page which contain part of the main content, the R2L Algorithm (and its next versions) moves up and down the area at the same third phase with a length of 20 lines ($gap = 20$) in order to find other areas containing the main content in the HTML file. This threshold might fail to reach the final goal which is main content extraction with a very high accuracy since it is possible that some parts of the HTML file which do not contain main content may incorrectly be considered as the main content. Therefore, when it is possible to identify both the upper and lower limits of the main content area, other main content regions can also be found with no need for the threshold. In this case, the accuracy of the main content extraction will be increased significantly. An algorithm is fortunately proposed

in Chapter 5 of this thesis which is able to identify and extract a headline from web pages. It is well known that the headline of each web page can be taken as the upper limit of the main content area. Thereby, R2L (and of course DANA, DANAg and AdDANAg) can be combined with TitleFinder to implement new algorithms which are capable of extracting the main content at a considerably higher accuracy. However, the TitleFinder algorithm can also be combined with other algorithms discussed in Chapter 3 including CCB, CETR, Density, DSC and BTE. It is expected that such a combination would improve the accuracy of main content extraction from web pages. After specification of the upper limit of the main content area, one may bear in mind the question of whether or not it is possible to identify the lower limit. It is well known that the main content area is often regarded as a sub-tree of the HTML DOM tree so one can search for the smallest sub-tree in the DOM tree which contains headlines of that main content area. The lower limit of main content area could be easily recognized by finding this sub-tree. In other words, all content characters in this sub-tree would be considered as the main content characters.

7.2.2 Posts Identifications and Main Content Extraction from Weblogs

It is clear that a great deal of data is placed on the web every day in the forms of text, image, video or audio. Part of this information is recorded by bloggers in personal weblogs, which could be never seen or accessed in other places. Therefore, the ability to extract this information could be significantly useful. All weblogs are comprised of some posts which are archived in the form of reverse-chronological order. Furthermore, in many weblogs it is possible to write comments for the existing posts there. These comments are also placed in the form of reverse-chronological order under each post. Now, one can address the following issues with respect to all this useful information in weblogs:

- How can one find out the number of posts in each weblog? When the number of posts in weblogs is known, it would be possible to get a trend of growth for the number of posts in continuous years. Meanwhile, since weblogs usually concentrate on a special subject, i.e. sports, politics, economics and other events, the interest of bloggers of various subjects can be determined in specific years.

- How can one identify and extract the existing main content in each post (30)? This issue has also been explored previously (120) (101), but since R2L algorithms and their next versions were more successful in main content extraction compared to other algorithms, using the algorithms proposed by this thesis (R2L, DANA, DANAg, AdDANAg) is expected to be done successfully for the extraction of main content from weblogs.

- How can one identify and extract the comments written for each post of a weblog? It is evident that the posts which are more important than others would contain a greater number of comments. Another idea suggested here is the classification of the comments. However, the extraction of the comments written for each post can be useful. It seems interesting to note that these comments are often classified under the same sub-tree of the DOM tree with the main content in that post. As a result, one can easily access to the comments of that post by the identification of the main content in each post using the relevant sub-tree which involves the post.

Finally, it must be noted that the weblogs have a more anarchic structure than web pages. Thereby, the algorithms which are going to be implemented for each of the above mentioned issues for weblogs become more complicated than those of web pages.

7.2.3 Extracting the Main Content of Web pages using Similarity Methods

Most of the algorithms proposed for the extraction of main content from web pages utilize some features such as tag density, character encoding and/or DOM tree structure in order to find and extract the location of main content. Thus, they have rarely used semantics of the HTML annotations for main content extraction. Based on what was mentioned previously in Chapter 5 of this thesis, the issue of main content extraction from web pages can be introduced as below:

It is obvious that by removing stopwords and running of stemming algorithms on the content tokens in a web page, one can expect that the sentences and paragraphs which comprise the main content to be similar to each other in terms of semantics. Therefore, by application of a proper similarity metric (such as cosine similarity or overlap scoring similarity), it would be possible to extract the main content from web pages at a great accuracy regardless of HTML DOM tree structure and HTML source code elements. It is clear that each similarity metric can use different weighting schemes (such as tf or *tf-idf*). Thus, appropriate selection of the weighting scheme can also contribute to the accuracy of the algorithm for main content extraction. Fortunately, this algorithm has been implemented to some extent with its results being promising.

Abbreviations

Recurring abbreviations used throughout the text, in alphabetical order:

ACCB	adapted content code bluring
AdDANAg	adapted DANA general
BM	boolean model
BTE	body text extraction
CBN	content base node
CCB	content code blurring
CCR	content code ratio
CCV	content code vector
CE	content extraction
CETD	content extraction via text density
CETR	content extraction via tag ratio
CRF	conditional random field
DANAg	DANA general
DOM	document object model
DSC	document slope curve
FE	FeatureExtracter
GSA	gaussian smoothing algorithm
GSWCE	gaussian smoothing-based Web content extraction
IR	information retrieval
K-FE	K-FeatureExtracter
KDT	knowledge discovery in text
LCS	longest common subsequence
LQF	link quota filter
MCE	main content extraction
MCEA	main content extraction algorithms
MDTDA	Document Template Detection Approaches
MEHH	maximum-entropy markov model
MSO	maximum subsequence optimization
MSS	maximum substring segmentation
NLP	natural language processing
NWBQAS	non-web based question answering system
PCT	potential content token
PCE	primary (main) content extraction

QA	question answering
QAS	question answering system
R2L	right to left
SDBA	single document based approaches
SMLM	supervised machine learning methods
SMO	sequential minimal optimization
SSLM	statistical sequence labeling models
SVM	support vector machines
TCCB	token based content code blurring
TD	template detection
TL	tokens list
TTM	temporal text mining
TTR	text-to-tag ratios
UCS	Unicode character set
VIPS	VIsion-based Page Segmentation
VSM	vector space model
WBQAS	web based question answering system
WCMS	Web content management system

Glossary

This appendix is devised to eliminate the ambiguity in some special words and phrases used in this thesis.

Body text extraction (BTE): An MCE algorithm in which the tokens in Web documents are tokenized into a number of content tokens and tag tokens. Then, a continuous subsequence, which contains the most content tokens and the least tag tokens, will be introduced as the MC in this sequence of tokens.

Adapted content code blurring(ACCB): ACCB algorithm is an adapted version of a Content Code Blurring (CCB) algorithm. Before Web documents are processed for MC extraction, all hyperlinks are removed from them in this algorithm. This version of a CCB algorithm makes it possible to extract MC from the Web pages which contain a great number of hyperlinks with a much higher accuracy.

Content code blurring(CCB): An MCE algorithm which uses a technique similar to Gaussian Blurring Filters for MC extraction from Web pages. The blurring filter is utilized on the content code vector made in this algorithm. Meanwhile, its purpose is to select the regions with a homogenously formatted text as the MC.

Web page: A Web page in this thesis is taken as a resource which is published on Web and is usually accessed through a Web browser. When a request is issued from a client via a Web browser, the user will be provided with a set of information in the form of an HTML file which contains images, CSS codes, Script codes, etc.

Web document: An HTML file, the MC of which has been attempted to be identified and extracted by this thesis.

Data set: A set of Web documents which are usually collected from several domains of Websites.

Noise; Extraneous items: Includes all unwanted information retrieved by an MCE algorithm together with relevant information. MCE algorithms try to remove the unwanted information during MC extraction and stop displaying them in the output. The most common

noises in extraction of MC from Web pages are navigation menus, advertisements, interaction elements and references to other information sources.

Main Content extraction (MCE): Includes the process of MC identification in a Web document. The algorithms which are usually employed for MC extraction are either categorized under machine learning based algorithms or heuristic algorithms. The later involves more algorithms in comparison with the former.

Crunch: It is a DOM-based CE framework which employs an heuristic method for filtering the Web pages. The main purpose of this framework is to improve accessibility for screen readers or small screen devices.

Document Slope Curve (DSC): It is a function for displaying the distribution of content tokens and tag tokens in a Web document. A number is attributed to the existing $i-th$ token in the sequence of tokens, which is equal with the number of tag tokens observed so far. It is evident that in the region where MC is located, tokens, i.e. content tokens, have the same values. Afterwards, in the DSC algorithm, the region with the shape of a plateau in the curve which is calculated and generated by the DSC function is taken as MC.

F1-measure: F1-measure is used to measure the quality of IR methods. F1-measure is obtained from the combination of Recall and Precision.

FeatureExtractor: This MCE algorithm is based on dividing a Web document into several blocks and then attributing certain features to each of these blocks. These features can be indicative of the number of words or images in a block and even the number of hyperlinks. Later on, the algorithm will look for blocks which have the desired features. The desired features include text in the field of MCE.

K-FeatureExtractor: This algorithm is an extended version of the FeatureExtractor algorithm. However, this algorithm selects K blocks as the MC instead of selecting one block. For this purpose it benefits from an adaptive k-means clustering.

Link quota Filter (LQF): It is a heuristic algorithm for extracting MC from Web pages. This algorithm looks for regions which include a high ratio of hyperlinks to identify noises in a Web page because these regions probably contain a navigation menu and link lists. Thus, removing them from one Web page will cause MCE to be accomplished with a much greater accuracy.

Longest common subsequence (LCS): The longest common subsequence (LCS) problem is to find the longest subsequence common to all sequences in a set of sequences (often just two).

Precision: Precision is used to measure the quality of IR methods. It includes the attribution of relevant items to a result set.

Recall: Recall is used to measure the quality of IR methods. It includes attribution of relevant retrieved items to all relevant items.

R2L: R2L is a new and simple approach to extract the main content of Right to Left language Web pages. Independence of DOM tree and HTML tags is one of the most important features of the proposed algorithm. In practice, HTML tags have been written in English and we know that the English character set is located in the interval [0,127]. In most languages which are written from Right-to-Left (R2L) such as the Arabic language, however, a definite interval of the Unicode character set is used that is certainly not in this interval. In the first phase of R2L, this distinction is applied to separate the R2L character set from the English ones. Then for each HTML file, the density of the R2L character set and the density of the Non-R2L character set is determined. That part of the HTML file with a high density of the R2L character set and a low density of the Non-R2L character set contains the main content of the Web page with high accuracy.

DANA: DANA is a novel approach for extracting the main content from Web documents written in languages not based on the Latin alphabet and it is the new version of R2L. In practice, the HTML tags are based on the English language and, certainly, the English character set is encoded in the interval [0,127] of the Unicode character set. On the other hand, many languages, such as the Arabic language, use a different interval for their characters. In the first phase of our approach, this distinction is used for a fast separation of the Non-ASCII from the English characters. After that, some areas of the HTML file with a high density of the Non-ASCII character set and a low density of the ASCII character set are determined. At the end of this phase, this density is used to identify the areas which contain the main content. Finally, we feed those areas to our parser in order to extract the main content of the Web page.

DANAg: DANAg a novel language-independent method for extracting the main content of web pages and it is the new version of DANA. The extraction process of DANAg is divided into four phases. In the first phase, we calculate the length of content and code of fixed segments in an HTML file. The second phase applies a naive smoothing method to highlight the segments forming the main content. After that, we use a simple algorithm to recognize the boundary of the main content in an HTML file. Finally, we feed the selected main content area to our parser in order to extract the main content of the targeted web page.

AdDANAg: AdDANAg is a language-independent approach to extract the main content of web documents and it is a new version of DANAg. This combination of techniques brings together two pre-processing steps, e.g. to normalize the document presentation and reduce the impact of certain syntactical structures, and four phases for the actual content extraction.

TrendFinder: TrendFinder is a novel and simple text mining approach that has been developed to reveal the content-based trends of expert-defined queries in selected published biology papers during the last five decades.

TitleFinder: TitleFinder is a content-based and domain-and language-independent approach for unsupervised extraction of the headline of web articles. TitleFinder starts by using an heuristic to select a candidate headline. In a second step the contents of each text fragment in the HTML file are compared to the candidate headline. Four types of similarity are implemented for this comparison: two variations of the cosine similarity based on *tf* and *tf-idf* weighting schemata, an overlap scoring similarity and an aggregated metric combining the scores of the previous three similarities.

References

[1] Aidin finn. BTE : Body text extraction, 2005. http://www.aidanf.net/archive/software/bte-body-text-extraction. [Online; accessed 26-January-2013].

[2] Document object model (DOM) level 4 specification. W3C recommendation, working draft. http://www.w3.org/DOM/DOMTR.html#dom4. [Online; accessed 26-January-2013].

[3] JDOM. http://www.jdom.org/. [Online; accessed 26-January-2013].

[4] JTidy. http://sourceforge.net/projects/jtidy/. [Online; accessed 26-January-2013].

[5] openxml. http://www.openxml.org/. [Online; accessed 26-January-2013].

[6] The swing HTML parser. http://java.sun.com/products/jfc/tsc/articles/bookmarks/. [Online; accessed 26-January-2013].

[7] Document object model (DOM) level 1 specification. W3C recommendation. http://www.w3.org/TR/REC-DOM-Level-1/, 1998. [Online; accessed 26-January-2013].

[8] Document object model (DOM) level 2 specification. W3C recommendation. http://www.w3.org/TR/DOM-Level-2-HTML/, 2000. [Online; accessed 26-January-2013].

[9] Document object model (DOM) level 3 specification. W3C recommendation. http://www.w3.org/DOM/DOMTR.html#dom3, 2004. [Online; accessed 26-January-2013].

[10] Homepage elements, sharp school helping schools succeed, how to edit your homepage. http://www.sharpschool.com/, 2008. [Online; accessed 26-January-2013].

[11] Internet 2011 in numbers. http://royal.pingdom.com/2012/01/17/internet-2011-in-numbers/, 2011. [Online; accessed 26-January-2013].

[12] Miniwatts marketing group, internet world stats, usage and population statistics. http://www.internetworldstats.com/, April 2012. [Online; accessed 26-January-2013].

[13] The unicode consortium. http://www.unicode.org/, March 2012. [Online; accessed 26-January-2013].

[14] Html: The markup language. `http://www.w3.org/TR/html-markup/syntax.html`, 2013. [Online; accessed 26-January-2013].

[15] Stopwords definition. `http://opensitesearch.sourceforge.net/docs/helpzone/dbb/dbb_50-10-12r.html`, 2013. [Online; accessed 26-January-2013].

[16] Stopwords definition. `http://en.wikipedia.org/wiki/Stop_words`, 2013. [Online; accessed 26-January-2013].

[17] Arasu, A., and Garcia-Molina, H. Extracting structured data from web pages. In *Proceedings of the 19th International Conference on Data Engineering* (Bangalore, India, 2003), IEEE Computer Society, p. 698.

[18] Baeza-Yates, R., and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison Wesley, 1999.

[19] Bar-Yossef, Z., and Rajagopalan, S. Template detection via data mining and its applications. In *Proceedings of the 11th international conference on World Wide Web* (2002), ACM Press, pp. 580 – 591.

[20] Boronat, X. A. A comparison of HTML-aware tools for web data extraction. Master thesis, University of Leipzig, Germany, Sep. 2008.

[21] Broder, A. Z., Fontoura, M., Josifovski, V., and Riedel, L. A semantic approach to contextual advertising. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Amsterdam, The Netherlands, 2007), ACM Press, pp. 559 – 566.

[22] Cai, D., Yu, S., Wen, J. R., and Ma, W. Y. Extracting content structure for web page based on visual representation. In *5th Asia Pacific Web Conference* (2003), Springer, pp. 406 – 417.

[23] Cai, D., Yu, S., Wen, J. R., and Ma, W. Y. VIPS: a vision-based page segmentation. In *Microsoft Technical Report, MSR-TR-2003-79* (2003).

[24] Cai, D., Yu, S., Wen, J. R., and Ma, W. Y. Block-based web search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (Sheffield, United Kingdom, 2004), ACM Press, pp. 456 – 463.

[25] Caillian, T. B.-L. R. World Wide Web: Proposal for a hypertext project. `http://www.w3.org/Proposal.html`, 1990. [Online; accessed 26-January-2013].

[26] Caumanns, J. *A Fast and Simple Stemming Algorithm for German Words*. Report / B. Freie Univ., Fachbereich Mathematik und Informatik, 1999.

[27] Changuel, S., Labroche, N., and Bouchon-Meunier, B. A general learning method for automatic title extraction from HTML pages. In *6th International Conference of Machine Learning and Data Mining in Pattern Recognition* (2009), Springer, pp. 704 – 718.

[28] Chen, L., Ye, S., and Li, X. Template detection for large scale search engines. In *Proceedings of the 2006 ACM Symposium on Applied Computing(SAC)* (2006), ACM Press, pp. 1094 – 1098.

[29] Chen, W., and Chundi, P. Trends analysis of topics based on temporal segmentation. In *Proceedings of 11th International Conference on Data Warehousing and Knowledge Discovery* (2009), pp. 402 – 414.

[30] Chen, Z. *Automatic Extraction of Blog Post from Diverse Blog Pages.* Phd thesis, National Central University, China, 2011.

[31] Chi, Y., Tseng, B. L., and Tatemura, J. Eigen-trend: trend analysis in the blogosphere based on singular value decompositions. In *Proceedings of the 15th ACM international conference on Information and knowledge management* (New York, NY, USA, 2006), CIKM '06, ACM Press, pp. 68 – 77.

[32] Comstock, S. *An Introduction to Unicode.* The Trainer's Friend, Inc., 2011.

[33] Debnath, S., Mitra, P., and Lee Giles, C. Automatic extraction of informative blocks from webpages. In *Proceeding of the Special Track on Web Technologies and Applications in the ACM Symposium of Applied Computing* (2005), pp. 1722 – 1726.

[34] Debnath, S., Mitra, P., and Lee Giles, C. Identifying content blocks from web documents. In *Foundations of Intelligent Systems* (2005), vol. 3488 of *Lecture Notes in Computer Science*, pp. 285 – 293.

[35] Evert, S. A lightweight and efficient tool for cleaning web pages. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)* (may 2008), European Language Resources Association (ELRA).

[36] Fairon, C., Naets, H., Kilgarriff, A., and de Schryver, G.-M., Eds. *WAC3: Proceedings of the 3rd web as corpus workshop, incorporating cleaneval* (Louvain-la-Neuve, Belgium, sep 2007), Cahiers du CENTAL, Presses universitaires de Louvain.

[37] Falco, M.-H., Moriceau, V., and Vilnat, A. Kitten: a tool for normalizing HTML and extracting its textual content. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)* (Istanbul, Turkey, May 2012), N. C. C. Chair), K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, Eds., European Language Resources Association (ELRA), pp. 2261 – 2267.

[38] Fan, J., Luo, P., and Joshi, P. Title identification of web article pages using HTML and visual features. In *Proceedings of SPIE Digital Library* (2011), vol. 7879.

[39] Fan, J., Luo, P., Lim, S. H., Liu, S., Parag, J., and Liu, J. Article clipper: a system for web article extraction. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2011), ACM Press, pp. 743 – 746.

[40] Feldman, R., Dagan, I., and Hirsh, H. Mining text using keyword distribution. *Journal of Intelligent Information Systems 10* (1998), 281 – 300.

[41] Finn, A., Kushmerick, N., and Smyth, B. Fact or fiction: Content classification for digital libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries* (2001).

[42] Gibson, D., Punera, K., and Tomkins, A. The volume and evolution of web page templates. In *WWW '05: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web* (New York, NY, USA, 2005), ACM Press, pp. 830 – 839.

[43] Gibson, J., Wellner, B., and Lubar, S. Adaptive web-page content identification. In *9th ACM International Workshop on Web Information and Data Management (WIDM 2007)* (2007), ACM Press, pp. 105 – 112.

[44] Giuliani, A. *Studying, Developing, and Experimenting Contextual Advertising Systems.* Phd thesis, University of Cagliari, Italy, 2012.

[45] Giustina, F., Tripp, A., Gold, R., Peskin, G. L., and Lempinen, S. JTidy - HTML parser and pretty-printer in java. `http://jtidy.sourceforge.net/`, 2008. [Online; accessed 26-January-2013].

[46] Gottron, T. Evaluating content extraction on HTML documents. In *ITA '07: Proceedings of the 2nd International Conference on Internet Technologies and Applications* (Wrexham, Wales, UK, 2007), pp. 123 – 132.

[47] Gottron, T. Combining content extraction heuristics: the *combine* system. In *iiWAS'2008 - The Tenth International Conference on Information Integration and Web-based Applications Services* (2008), ACM Press, pp. 591 – 595.

[48] Gottron, T. Content code blurring: A new approach to content extraction. In *DEXA'08: 19th International Workshop on Database and Expert Systems Applications* (Turin, Italy, 2008), IEEE Computer Society, pp. 29 – 33.

[49] Gottron, T. *Content extraction: Identifying the main content in HTML documents.* Phd thesis, Johannes Gutenberg-Universität Main, Germany, 2008.

[50] Gottron, T. An evolutionary approach to automatically optimise web content extraction. In *IIS'09: Proceedings of the 17th International Conference Intelligent Information Systems* (2009), pp. 331 – 343.

[51] Gudivada, V. N., Raghavan, V. V., Grosky, W. I., and Kasanagottu, R. Information retrieval on the World Wide Web. *IEEE Internet Computing 1*, 5 (1997), 58 – 68.

[52] Gupta, P., and Gupta, V. Article: A survey of text question answering techniques. *International Journal of Computer Applications 53*, 4 (September 2012), 1 – 8. Published by Foundation of Computer Science, New York, USA.

[53] Gupta, S. *Context Based Content Extraction of HTML Documents.* Phd thesis, Colombia University, USA, 2004.

[54] Gupta, S., Kaiser, G., Grimm, P., Chiang, M. F., and Starren, J. Automating content extraction of HTML documents. *World Wide Web 8*, 2 (2005), 179–224.

[55] Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. DOM-based content extraction of HTML documents. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web* (New York, NY, USA, 2003), ACM Press, pp. 207 – 214.

[56] Gupta, S., Kaiser, G., and Stolfo, S. Extracting context to improve accuracy for HTML content extraction. In *WWW (Special interest tracks and posters)* (2005), ACM Press, pp. 1114 – 1115.

[57] Gupta, S., and Kaiser, G. E. Extracting content from accessible web pages. In *Proceedings of the International Cross-Disciplinary Workshop on Web Accessibility* (2005), vol. 88, ACM Press, pp. 26–30.

[58] Haralambous, Y. *Fonts and Encodings.* O'Reilly, 2007.

[59] Harb, A., Plantié, M., Dray, G., Roche, M., Trousset, F., and Poncelet, P. Web opinion mining: how to extract opinions from blogs? In *CSTST '08: Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology* (New York, NY, USA, 2008), ACM Press, pp. 211 – 217.

[60] Hickson, I. HTML 5. `http://www.whatwg.org/specs/web-apps/current-work/multipage/`, 2004 – 2013. [Online; accessed 26-January-2013].

[61] Hiemstra, D. Information retrieval models. In *Information Retrieval: Searching in the 21st Century*, A. Goker, J. Davies, and M. Graham, Eds. John Wiley and Sons, 2009, pp. 1 – 19.

[62] Hirschberg, D. S. A linear space algorithm for computing maximal common subsequences. *Communication ACM 18*, 6 (1975), 341 – 343.

[63] Htwe, T. Cleaning various noise patterns in web pages for web data extraction. *International Journal of Network and Mobile Technologies 1*, 2 (Nov. 2010), 74–80.

[64] Hu, Y., Li, H., Cao, Y., Teng, L., Meyerzon, D., and Zheng, Q. Automatic extraction of titles from general documents using machine learning. In *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2005* (2005), ACM Press, pp. 145 – 154.

[65] Hu, Y., Xin, G., Song, R., Hu, G., Shi, S., Cao, Y., and Li, H. Title extraction from bodies of HTML documents and its application to web page retrieval. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information* (August 2005), ACM Press, pp. 250 – 257.

[66] Ibrahim, H., Darwish, K., and Madany, A.-R. Automatic extraction of textual elements from news web pages. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008* (2008), pp. 1600 – 1603.

[67] Kaiser, G., Gupta, S., and Becker, H. Crunch - a web proxy for HTML content extraction. `http://www.psl.cs.columbia.edu/650/crunch/`, 2005. [Online; accessed 26-January-2013].

[68] Kim, S. N., Medelyan, O., Kan, M.-Y., and Baldwin, T. SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of 5th International Workshop on Semantic Evaluation* (2010), pp. 21 – 26.

[69] Kohlschütter, C., Fankhauser, P., and Nejdl, W. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining* (New York, NY, USA, 2010), WSDM '10, ACM, pp. 441–450.

[70] Kolcz, A., and tau Yih, W. Site-independent template-block detection. In *Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases* (Warsaw, Poland, 2007), Lecture Notes in Computer Science, Springer, pp. 152 – 163.

[71] Kovacic, T. Evaluating web content extraction algorithms. Master thesis, University of Ljubljana, Ljubljana, 2012.

[72] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)* (San Francisco, CA, USA, 2001), Morgan Kaufmann Publishers Inc., pp. 282 – 289.

[73] Liu, B. *Web Data Mining*. Springer, 2011.

[74] Liu, C., and Liao, B. Gaussian smoothing-based web content extraction. *International Journal of Advancements in Computing Technology(IJACT) 3*, 8 (2011), 285 – 293.

[75] Louvan, S. Extracting the main content from web documents. Master thesis, Eindhoven University of Technology, Netherlands, 2009.

[76] Ma, L., Goharian, N., Chowdhury, A., and Chung, M. Extracting unstructured data from template generated web documents. In *Proceedings of the twelfth international conference on Information and knowledge management* (New Orleans, LA, USA, 2003), CIKM '03, ACM Press, pp. 512 – 515.

[77] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematics Statistics and Probability* (1967), University of California Press, pp. 281–297.

[78] Manning, C., Raghavan, P., and Schütze, H. *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[79] Mantratzis, C., and Cassidy, S. DOM-based XHTML document structure analysis separating content from navigation elements. In *CIMCA/IAWTIC* (2005), IEEE Computer Society, pp. 632 – 637.

[80] Mantratzis, C., Orgun, M., and Cassidy, S. Separating XHTML content from navigation clutter using DOM-structure block analysis. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia* (New York, NY, USA, 2005), ACM Press, pp. 145 – 147.

[81] Marco Baroni, Francis Chantree, A. K., and Sharoff, S. Cleaneval: a competition for cleaning web pages. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)* (may 2008), European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[82] Mei, Q., and Zhai, C. Discovering evolutionary theme patterns from text-an exploration of temporal text mining. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Illinois, USA, 2005), ACM Press, pp. 285 – 293.

[83] Mohammadzadeh, H., Gottron, T., Schweiggert, F., and Heyer, G. TitleFinder: Extracting the headline of news web pages based on cosine similarity and overlap scoring similarity. In *WIDM'12: Proceedings of the 12th International ACM Workshop on Web Information and Data Management* (Hawaii, USA, October 2012), ACM Press, pp. 65 – 71.

[84] Mohammadzadeh, H., Gottron, T., Schweiggert, F., and Nakhaeizadeh, G. Extracting the main content of web documents based on a naive smoothing method. In *KDIR'11: International Conference on Knowledge Discovery and Information Retrieval* (Paris, France, 2011), SciTePress, pp. 470 – 475.

[85] Mohammadzadeh, H., Gottron, T., Schweiggert, F., and Nakhaeizadeh, G. A fast and accurate approach for main content extraction based on character encoding. In *TIR'11: Proceedings of the 8th International Workshop on Text-based Information Retrieval(DEXA'11)* (Toulouse, France, 2011), IEEE Computer Society, pp. 167 – 171.

[86] Mohammadzadeh, H., Gottron, T., Schweiggert, F., and Nakhaeizadeh, G. The impact of source code normalization on main content extraction. In *WEBIST'12: 8th International Conference on Web Information Systems and Technologies* (Porto, Portugal, 2012), SciTePress, pp. 677 – 682.

[87] Mohammadzadeh, H., Gottron, T., Schweiggert, F., and Nakhaeizadeh, G. Extracting the main content of web documents based on character encoding and a naive smoothing method. In *Software and Data Technologies, CCIS Series, Springer*, M. J. E. Cuaresma, B. Shishkov, and J. Cordeiro, Eds. Springer-Verlag Berlin Heidelberg, January 2013, pp. 217 – 236.

[88] Mohammadzadeh, H., Paknia, O., Schweiggert, F., and Gottron, T. Revealing trends based on defined queries in biological publications using cosine similarity. In

BIOKDD'12: 23th International Workshop on Database and Expert Systems Applications (DEXA'12) (Vienna, Austria, 2012), IEEE Computer Society, pp. 218 – 222.

[89] Mohammadzadeh, H., Schweiggert, F., and Nakhaeizadeh, G. Using UTF-8 to extract main content of right to left language web pages. In *ICSOFT 2011 - Proceedings of the 6th International Conference on Software and Data Technologies* (Seville, Spain, 2011), M. J. E. Cuaresma, B. Shishkov, and J. Cordeiro, Eds., vol. 1, SciTePress, pp. 243 – 249.

[90] Moreno, F. J. A. Search engine customization and data set builder. Master thesis, Katholieke Universiteit Leuven, Belgium, 2009.

[91] Moreno, J., Deschacht, K., and Moens, M. Language independent content extraction from web pages. In *Proceeding of the 9th Dutch-Belgian Information Retrieval Workshop* (2009), pp. 50 – 55.

[92] Naveed, N., Sizov, S., and Staab, S. ATTention: Understanding authors and topics in context of temporal evolution. In *Proceedings of European Conference on Information Retrieval* (2011), pp. 733 – 737.

[93] Nguyen, D. Q., Nguyen, D. Q., Pham, S. B., and Bui, T. D. A fast template-based approach to automatically identify primary text content of a web page. In *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering* (Washington, DC, USA, 2009), KSE '09, IEEE Computer Society, pp. 232 – 236.

[94] Parmar, H. R., and Gadge, J. Article: Removal of image advertisement from web page. *International Journal of Computer Applications 27*, 7 (August 2011), 1 – 5. Published by Foundation of Computer Science, New York, USA.

[95] Pasternack, J., and Roth, D. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of the 18th international conference on World Wide Web* (New York, NY, USA, 2009), WWW '09, ACM press, pp. 971 – 980.

[96] Pinto, D., Branstein, M., Coleman, R., Croft, W. B., King, M., Li, W., and Wei, X. 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* (New York, NY, USA, 2002), ACM Press, pp. 46 – 55.

[97] Pomikalek, J. *Removing Boilerplate and Duplicate Content from Web Corpora*. PhD thesis, Masaryk University, Czech Republic, 2011.

[98] Porter, M. F. An algorithm for suffix stripping. *Program 14*, 3 (July 1980), 130 – 137.

[99] Raggett, D. Clean up your web pages with HTML TIDY. `http://www.w3.org/People/Raggett/tidy/`, 1994 – 2012. [Online; accessed 26-January-2013].

[100] Rahman, A. F. R., Alam, H., and Hartono, R. Content extraction from HTML documents. In *WDA 2001: Proceeding of the First International Workshop on Web Document Analysis* (2001), pp. 7 – 10.

[101] Ramoji, E. D. B., and Saravanakumar, K. Summarizing and clustering the blog content for improved blog mining. *International Journal of Research and Reviews in Computer Science(IJRRCS) 2*, 2 (2011), 547 – 553.

[102] Ramprasath, M., and Hariharan, S. Article: A survey on question answering system. *International Journal of Research and Reviews in Information Sciences (IJRRIS) 2*, 1 (2012), 171 – 178.

[103] Rijsbergen, C. J. V. *Information Retrieval*. Butterworths, 2nd edition, 1979.

[104] Ruzzo, W. L., and Tompa, M. A linear time algorithm for finding all maximal scoring subsequences. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology* (1999), AAAI, pp. 234 – 241.

[105] Salton, G. *The SMART Retrieval System; Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[106] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24*, 5 (Aug. 1988), 513 – 523.

[107] Salton, G., and Lesk, M. Computer evaluation of indexing and text processing. *Journal of the ACM (JACM) 15*, 1 (1968), 8 – 36.

[108] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Communications of the ACM 18* (1975), 613 – 620.

[109] Salton, G., and Yang, C. S. On the specification of term values in automatic indexing. *Journal of Documentation 29*, 4 (1973), 351 – 372.

[110] Sivakumar, P., and Parvathi, R. M. S. An efficient approach of noise removal from web page for effectual web content mining. *European Journal of Scientific Resaech* (2011), 345 – 356.

[111] Spousta, M., Marek, M., and Pecina, P. Victor: the Web-Page Cleaning Tool. In *Proceedings of the 4th Web as Corpus Workshop, LREC* (2008).

[112] Sun, F., Song, D., and Liao, L. DOM based content extraction via text density. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR* (2011), ACM Press, pp. 245 – 254.

[113] Sun, F., Song, D., and Liao, L. DOM based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM Press, pp. 245 – 254.

[114] Taghva, K., Beckley, R., and Sadeh, M. A list of farsi stopwords. In *Technical Report 2003-01, Information Science Research Institute, University of Nevada* (Las Vegas, USA, 2003).

[115] Taghva, K., Beckley, R., and Sadeh, M. A stemming algorithm for the farsi language. In *International Symposium on Information Technology: Coding and Computing (ITCC)* (Las Vegas, Nevada, USA, 2005), vol. 1, IEEE Computer Society, pp. 158 – 162.

[116] Tim Berners-Lee, C. Information management: A proposal. `http://www.w3.org/History/1989/proposal.html`, 1990. [Online; accessed 26-January-2013].

[117] Tremante, M. Netcraft services. `http://www.netcraft.com/`, 2012. [Online; accessed 26-January-2013].

[118] Vieira, K., da Silva, A. S., Pinto, N., de Moura, E. S., Cavalcanti, J. a. M. B., and Freire, J. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM international conference on Information and knowledge management* (Arlington, Virginia, USA, 2006), CIKM '06, ACM Press, pp. 258 – 267.

[119] Wang, Y., Fang, B., Cheng, X., Guo, L., and Xu, H. Incremental web page template detection. In *Proceedings of the 17th International Conference on World Wide Web* (2008), ACM Press, pp. 1247 – 1248.

[120] Weerkamp, W., and Hofmann, K. Content extraction for information retrieval in blogs and intranets. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008* (Napa Valley, California, USA, 2008).

[121] Weiss, S. *Unicode and UTF-8.* Graphical User Interface Programming, 2011.

[122] Weissig, Y., and Gottron, T. Combinations of content extraction algorithms. In *LWA'09: Workshop Information Retrieval. Darmstadt* (2009).

[123] Weninger, T., and Hsu, W. H. Text extraction from the web via text-tag-ratio. In *TIR '08: Proceedings of the 5th International Workshop on Text Information Retrieval* (Turin, Italy, Sept. 2008), IEEE Computer Society, pp. 23 – 28.

[124] Weninger, T., Hsu, W. H., and Han, J. CETR: content extraction via tag ratios. In *Proceedings of the 19th International Conference on World Wide Web* (2010), ACM Press, pp. 971 – 980.

[125] Xiao, J., Huang, G., and Tang, Y. An open source web browser for visually impaired. In *Proceedings of the intelligent computing 3rd international conference on Advanced intelligent computing theories and applications* (Qingdao, China, 2007), ICIC'07, Springer-Verlag, pp. 90 – 101.

[126] Xue, Y., Hu, Y., Xin, G., Song, R., Shi, S., Cao, Y., Lin, C.-Y., and Li, H. Web page title extraction and its application. *Information Processing and Management: an International Journal 43*, 5 (2007), 1332 – 1347.

[127] Yi, L., Liu, B., and Li, X. Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge*

discovery and data mining (Washington, D.C., 2003), KDD '03, ACM Press, pp. 296 – 305.

[128] Zhang, Z., Sun, M., and Liu, S. Automatic content based title extraction for chinese documents using support vector machine. In *Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering* (2005), IEEE, pp. 553 – 558.

Index

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

Ulm, den 10. December 2013

. . . . . .
(Unterschrift)