AUTOMATED QUERY-BIASED AND STRUCTURE-PRESERVING DOCUMENT SUMMARIZATION FOR WEB SEARCH TASKS

by
Fatma Canan Pembe
B.S., Computer Engineering, Boğaziçi University, 2002
M.S., Computer Engineering, Boğaziçi University, 2004

A Thesis Proposal
submitted to the Thesis Supervisory Committee
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering
Boğaziçi University
2010

AUTOMATED QUERY-BIASED AND STRUCTURE-PRESERVING DOCUMENT SUMMARIZATION FOR WEB SEARCH TASKS

APPROVED BY:

Assoc. Prof. Tunga Güngör . . . .
(Thesis Supervisor)

Assist. Prof. Murat Saraçlar . . . .

Prof. A. C. Cem Say . . . .

DATE OF APPROVAL: .....

ACKNOWLEDGEMENTS

…

ABSTRACT

AUTOMATED QUERY-BIASED AND STRUCTURE-PRESERVING DOCUMENT SUMMARIZATION FOR WEB SEARCH TASKS

With the drastic increase of available information sources on the Internet, people with different backgrounds in the world share the same problem: locating useful information for their actual needs. Search engines provide a means for users to locate documents on the Web via queries. However, users still have to perform the sifting process by themselves; i.e., to decide the relevance of each document with respect to their actual information needs. At this point, automatic summarization techniques can complement the task of search engines.

Currently available search engines, such as Google and AltaVista, only show a limited capability in summarizing the Web documents; e.g. displaying only two or three lines of text fragments which consist of the query words and their surrounding text as the summary. In the literature, most of the research in automatic summarization has focused on creating general-purpose summaries without considering user needs. Also, summarization approaches have mostly seen a document as a flat sequence of sentences and ignored the structure within the documents. In the summarization literature, the effect of query-biased techniques and document structure have been considered only in a few studies and separately investigated. This research is distinguished from previous work by combining these two aspects in a coherent framework. In this thesis, we propose a novel summarization approach for Web search, i.e., query-biased and structure-preserving document summarization.

The proposed system consists of two main stages. The first stage is the structural processing of Web documents in order to extract their section and subsection hierarchy together with the corresponding headings and subheadings. A document in the system is represented as an ordered tree of headings, subheadings and other text units. First, we formed a rule-based approach based on heuristics and HTML Document Object Model tree processing. Then, we developed a machine learning approach based on the tree representation using support vector machine (SVM) and perceptron algorithms. The methods were evaluated based on the accuracy of heading extraction and hierarchy extraction.

The second stage of the research is to develop automatic summarization methods by utilizing the document structures obtained in the first stage. In the proposed method, the summary sentences are extracted in a query-biased way based on two levels of scoring: sentence scoring and section scoring. Document structure is utilized both in the summarization process and in the output summaries. The performance of the proposed system has been determined using several task-based evaluations. These include information retrieval tasks where the summaries will actually be used. The results of the experiments on Turkish and English documents show that the proposed system summaries are superior to Google extracts and unstructured query-biased summaries of the same size in terms of accuracy with reasonable judgment times. User ratings verify that query-biased and structure-preserving summaries are also found to be more useful by the users.

ÖZET

ARAMA MOTORLARI İÇİN BİLGİ İSTEĞİNE VE METİN YAPISINA DAYALI OLARAK OTOMATİK DOKÜMAN ÖZETLENMESİ

İnternet'teki bilgi kaynaklarındaki büyük artışla birlikte, dünyada değişik arka planlara sahip insanlar aynı problemi paylaşmaktadır: Gerçek ihtiyaçlarına uygun bilgileri bulmak. Arama motorları, kullanıcıların, bilgi istekleri vasıtasıyla İnternet'teki dokümanları bulmaları için bir araç sağlamaktadır. Ancak, kullanıcıların eleme işlemini, yani her bir dokümanın gerçek bilgi ihtiyaçlarıyla ilgisine karar verme işlemini, halen kendilerinin yapması gerekmektedir. Bu noktada, otomatik özetleme yöntemleri, arama motorlarının görevini tamamlayabilir.

Günümüzde mevcut arama motorları, örneğin Google ve AltaVista, İnternet dokümanlarını özetlemede, sadece bilgi isteğindeki kelimeler ve çevrelerindeki metni içeren iki ya da üç satırlık özetler sunmak gibi, sınırlı bir yetkinlik göstermektedir. Literatürde, otomatik özetleme konusundaki araştırmaların çoğu, kullanıcı ihtiyaçlarını dikkate almayarak genel amaçlı özetler oluşturma üzerine odaklanmıştır. Ayrıca, özetleme yaklaşımları bir dokümanı çoğunlukla düz bir cümle dizisi olarak görmekte ve dokümanlardaki yapıyı göz ardı etmektedir. Özetleme literatüründe, bilgi isteğine dayalı yöntemler ve doküman yapısı sadece az sayıda çalışmada ve ayrı ayrı ele alınmıştır. Bu çalışma, önceki çalışmalardan bu iki yönü tutarlı bir çerçevede bir araya getirmesiyle ayrılmaktadır. Bu tezde, İnternet araması için özgün bir özetleme yaklaşımı öneriyoruz: Bilgi isteği ve doküman yapısına dayalı özetleme.

Önerilen sistem, iki temel aşamadan meydana gelmektedir. İlk aşama, İnternet dokümanlarının bölüm ve alt bölüm hiyerarşilerinin ilgili başlık ve alt başlıklarla birlikte ortaya çıkarılması için yapısal olarak işlenmesidir. Sistemdeki her bir doküman, başlıklar, alt başlıklar ve diğer metin birimlerinden oluşan sıralı bir ağaç yapısı ile temsil edilmektedir. İlk olarak, buluşsal yöntemler ve HTML Belge Nesne Modeli'ndeki ağaç yapısının işlenmesine dayalı kural tabanlı bir yaklaşım oluşturduk. Daha sonra, destek vektör makineleri ile algılayıcı algoritmalarını kullanan ve ağaç gösterimine dayalı bir makine öğrenmesi yaklaşımı geliştirdik. Yöntemler, başlık ve hiyerarşi çıkarma işlemlerinin başarımına göre değerlendirildi.

Çalışmanın ikinci aşaması, ilk aşamada elde edilen doküman yapılarından faydalanılarak otomatik özetleme yöntemlerinin geliştirilmesidir. Önerilen yöntemde, özet cümleleri, bilgi isteğine dayalı olarak iki seviyede değerlendirmeyle seçilmektedir: Cümle bazında puanlama ve bölüm bazında puanlama. Doküman yapısı, hem özetleme işlemi sırasında hem de üretilen özetlerde kullanılmaktadır. Sistemin başarımı, göreve yönelik değerlendirmelerle belirlenmiştir. Değerlendirmeler, özetlerin gerçekte kullanılacağı gibi bilgiye erişim görevleri içermektedir. Türkçe ve İngilizce dokümanlar üzerinde yapılan deneylerin sonuçları, önerilen sistemin özetlerinin, Google özetleri ve aynı boyutlardaki doküman yapısı bilgisini kullanmayan bilgi isteğine yönelik özetlere göre, makul karar süreleriyle, doğruluk açısından üstünlük sağladığını göstermektedir. Kullanıcı derecelendirmeleri de, bilgi isteği ve doküman yapısına dayalı özetlerin kullanıcılar tarafından daha faydalı bulunduğunu doğrulamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

A	Accuracy
$e(p,c)$	Existence of an edge between node pair (p,c)
F	F-measure
F_{ij}	Composite features of units u and u_{ij} in hierarchy extraction
$F_{n(n\pm k)}$	Composite features of units u_n and $u_{n\pm k}$ in heading extraction
FN	Number of false negatives
FNR	False negative rate
FP	Number of false positives
FPR	False positive rate
M_i	A variation of the testing approach
P	Precision
PC_i	Set of parent-child node pairs for a document hierarchy i
R	Recall
$s_{heading}$	Heading score
$s_{location}$	Location score
s_{query}	Query score
$s_{section}$	Section score
$s_{sentence}$	Sentence score
$s_{subsection}$	Subsection score
s_{tf}	Term frequency score
$T_{document}$	Time to view a document
T_{page_load}	Time to load a Web document into the browser
$T_{summary}$	Time to view a summary
TN	Number of true negatives
TP	Number of true positives
u_i	Text unit i in a document
u_{ij}	The unit i levels above a unit u, and j units to its left in the hierarchy
w	Weight vector in a machine learning model
$w_{heading}$	Weight of heading score

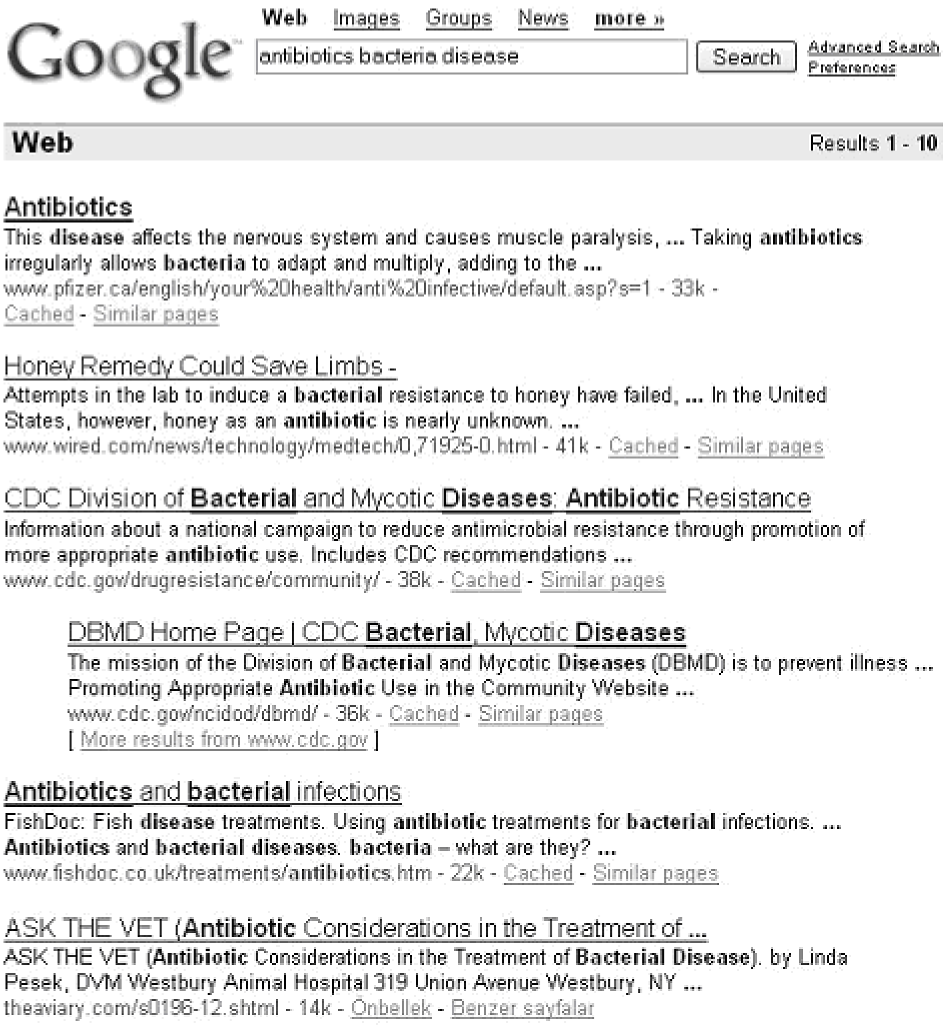
$w_{location}$	Weight of location score
w_{query}	Weight of query score
w_{tf}	Weight of term frequency score
α	Parameter vector in a machine learning model
Φ	Feature vector in a machine learning model
ANOVA	Analysis of Variance
CRF	Conditional Random Fields
CSS	Cascading Style Sheets
DOM	Document Object Model
DUC	Document Understanding Conferences
GATE	General Architecture for Text Engineering
HTML	Hypertext Markup Language
ID	Identification Number
PCFG	Probabilistic Context Free Grammar
PDA	Personal Digital Assistant
RBF	Radial Basis Function
SVM	Support Vector Machine
TREC	Text Retrieval Conference
XML	Extensible Markup Language

1. INTRODUCTION

The drastic increase of available documents on the World Wide Web resulted in the wide-spread problem of "information overload" [1]. That is, Internet users now have access to vast amounts of information especially with the advent of search engines; however, it is becoming more and more difficult and time-consuming to locate useful information with respect to their actual information needs. The available information needs to be efficiently used and there is no time to read everything individually. Search engines usually return large numbers of results in response to user queries. A research on European users showed that about 50% of documents viewed by users are irrelevant to their actual needs [2]. Users need to explore several links in the search engine results in order to find useful information. This is especially the case for specific and complex queries (e.g. best countries for retirement) and for tasks such as background search (e.g. literature survey on Mexican air pollution) rather than the ones with commonplace answers (e.g. the capital city of Sweden).

In currently available search engines, such as Google and AltaVista, each link in the results is associated with a short 'summary' (e.g. two-line extracts) of its content [3, 4]. The users need to scroll down the result pages, have a look at the links together with their extracts one by one and click only the ones that seem relevant to their information need. Such extracts may be very useful in directing the users to relevant documents. However, in practice, they are too short and although they show some of the document fragments containing the query words, they fail to reveal their context within the whole document in a higher level. In other words, especially in the case of long and complex documents, such extracts do not give sufficient information about the document contents. As a result of these inadequacies, the users can either miss relevant results or spend time with irrelevant ones.

In Figure 1.1, the first six results of Google in response to the TREC-2004 query (i.e. topic) *antibiotics bacteria disease* are given [5]. In that task, the aim of the user is to find documents that discuss how and why antibiotics become ineffective to some bacteria types. Examining the documents corresponding to the results in Figure 1.1, it can be seen that only half of the given extracts may effectively direct users. As an example, the second result seems relevant considering its extract; however, when it is opened in the browser, it turns out to be irrelevant. For the fourth result, the opposite is the case: considering the search engine extract, the document seems irrelevant; however, in fact, it is relevant. Finally, the extract for the fifth result does not provide sufficient information to make a decision for its relevancy. Besides the deficiencies of such extracts, the alternative strategy of opening each link in the results without taking the search engine extracts into account is also not feasible. The reason is that page loading takes time and determining relevancy may be difficult and time-consuming in the case of long and complex documents. Also, there are usually a large number of returned results. Better approaches of summarization may be used to overcome these problems and to improve the search experience of Internet users.



antibiotics bacteria disease

Antibiotics
This disease affects the nervous system and causes muscle paralysis, ... Taking antibiotics irregularly allows bacteria to adapt and multiply, adding to the ...
www.pfizer.ca/english/your%20health/anti%20infective/default.asp?s=1 - 33k - Cached - Similar pages

Honey Remedy Could Save Limbs -
Attempts in the lab to induce a bacterial resistance to honey have failed, ... In the United States, however, honey as an antibiotic is nearly unknown. ...
www.wired.com/news/technology/medtech/0,71925-0.html - 41k - Cached - Similar pages

CDC Division of Bacterial and Mycotic Diseases: Antibiotic Resistance
Information about a national campaign to reduce antimicrobial resistance through promotion of more appropriate antibiotic use. Includes CDC recommendations ...
www.cdc.gov/drugresistance/community/ - 38k - Cached - Similar pages

DBMD Home Page | CDC Bacterial, Mycotic Diseases
The mission of the Division of Bacterial and Mycotic Diseases (DBMD) is to prevent illness ... Promoting Appropriate Antibiotic Use in the Community Website ...
www.cdc.gov/ncidod/dbmd/ - 36k - Cached - Similar pages
[More results from www.cdc.gov]

Antibiotics and bacterial infections
FishDoc: Fish disease treatments. Using antibiotic treatments for bacterial infections. ... Antibiotics and bacterial diseases. bacteria – what are they? ...
www.fishdoc.co.uk/treatments/antibiotics.htm - 22k - Cached - Similar pages

ASK THE VET (Antibiotic Considerations in the Treatment of ...
ASK THE VET (Antibiotic Considerations in the Treatment of Bacterial Disease). by Linda Pesek, DVM Westbury Animal Hospital 319 Union Avenue Westbury, NY ...
theaviary.com/s0196-12.shtml - 14k - Önbellek - Benzer sayfalar

Figure 1.1. An example output of Google

At this point, automatic summarization research gains importance. Automatic summarization is a rather old field of research dating back to late 1950s [6]. However, there has been an increasing attention to this field from governments, academia and industry in recent years with the rapid growth of accessible information sources, mostly the World Wide Web. Automatic summarization research has traditionally aimed at creating human-like and generic summaries. However, as stated by Sparck-Jones [7], creating automatic summaries as successful as human summaries may be a long-term research direction, but meanwhile summaries which are not perfect can be utilized for improving the effectiveness of other tasks. Automatic summaries may be especially helpful in the information retrieval task; i.e., the task of finding relevant documents in a large collection in order to satisfy user queries for particular items of information.

Most of the automatic summarization studies focus on creating general-purpose summaries of documents. However, in an information retrieval paradigm, it has become important to bias summaries towards user queries in order to be effective. Also, traditional approaches have usually considered a document as a flat sequence of sentences and ignored the inherent structure of documents during the summarization process and in the output summaries. This aspect becomes especially important in the context of Web documents which typically show complex organization of content, having sections and subsections with different topics and formatting.

In this thesis, we propose a novel summarization approach for Web search which combines these two aspects, namely, document structure and query-biased techniques, both in the summarization process and in the output summaries. In the summarization literature, these aspects have been investigated separately and only in a small number of studies. To the best of our knowledge, the effects of explicit document structure and query-biased techniques on Web search have not been investigated together in previous studies. The intuition behind the proposed method is that providing the context of searched terms in a way that is preserving the structure of the document (i.e. sectional hierarchy and heading structure) may help the users to determine the relevancy of the results better. First, the document structure is used to determine important sections and subsections of a document depending on the user query. Second, the structure is also provided as a part of the summary (i.e. headings and subheadings under which the important sentences are located).

We developed a two-stage approach: Structural processing and summary extraction. In the first stage, the structure of a given Web document is automatically analyzed and its sectional hierarchy is extracted together with the headings and subheadings in the hierarchy. This is based on the idea that a document can be represented as a tree with order and containment relations between physical and logical components of the document [8]. We first developed a rule-based approach for document structure analysis using heuristics. Then, we formed a machine learning approach which can be more flexible than the rule-based approach. Here, we adapted a tree-based learning method using support vector machines and perceptron. The second stage of the proposed system is summary extraction and depends on the output of the first stage; i.e., document structure. We developed a query-biased summarization approach which uses the structural information during the summarization process and in the output summaries. For this purpose, basic statistical approaches to summarization are adapted.

The proposed system has been evaluated on two levels. First, the proposed document structure analysis methods are evaluated based on the accuracy of heading extraction and document sectional hierarchy extraction. Then, the outputs of the proposed summarization approach are tested using a task-based evaluation method, where the task is information retrieval. The results are compared with both Google extracts and unstructured summaries. The evaluation sets include Turkish and English document collections and queries.

The main contributions of this thesis are outlined in the following. Some of the thesis results have been published:

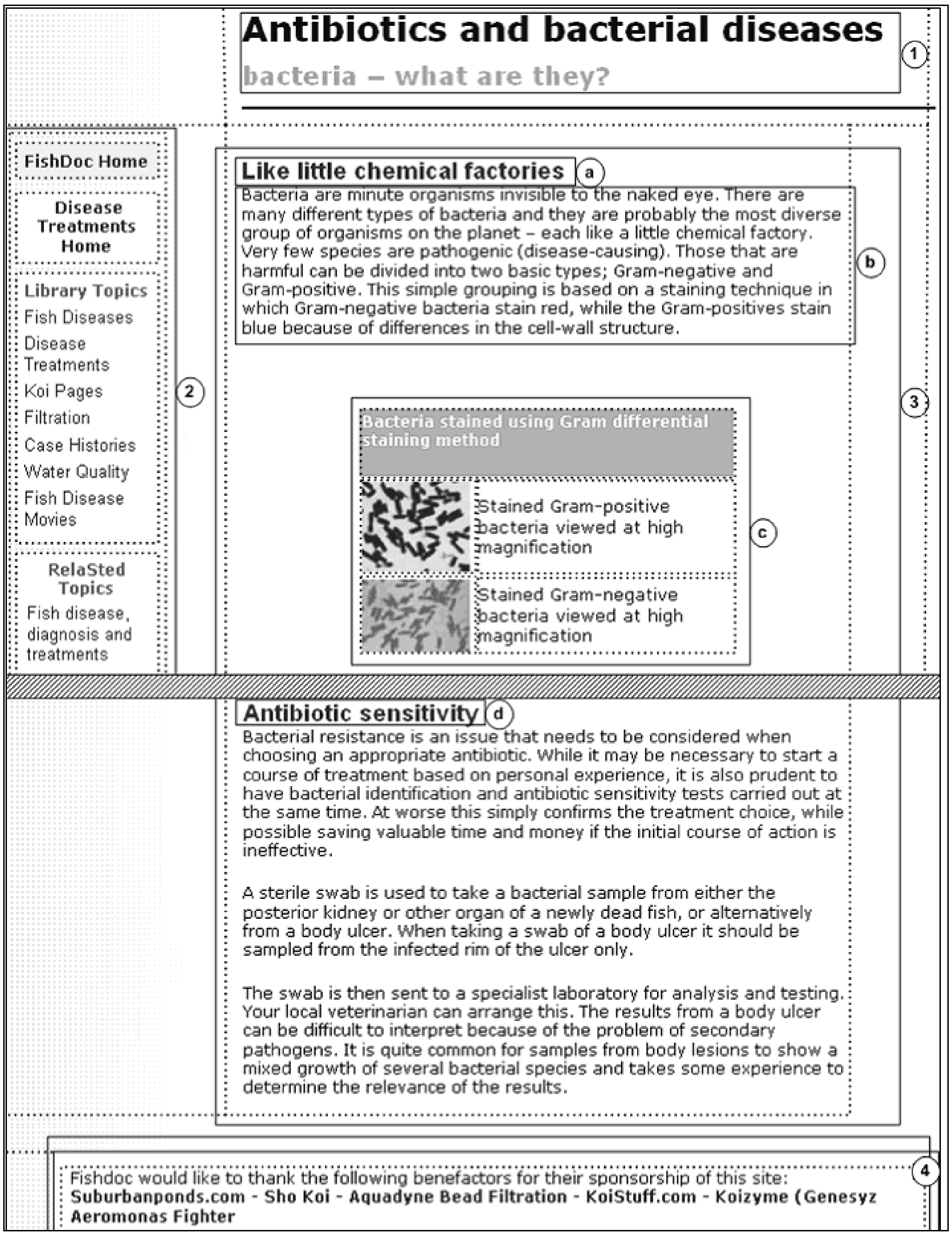
- A novel summarization approach based on document structure and query-biased techniques [9].
- Automatic analysis of domain-independent Web documents to obtain a hierarchy of sections and subsections together with the headings using a rule-based approach [10] and machine learning approaches [11].
- The first automatic summarization study for Turkish targeting Web search [12].
- Evaluation of the structure-preserving and query-biased summaries in Web search tasks [13, 14].

1.1. Research Goals

We state the main research goal as creating more effective summaries than the ones provided by traditional search engines in order to help the users judge the relevancy of search results better. The targeted documents are general Web documents with no domain restriction. Web documents are typically heterogeneous documents containing images, texts in different formats, interactive forms, menus, etc. Their content may also be diversified with sections on different topics, advertisements, etc. The screenshot of top and bottom parts of a rather long Web document is given in Figure 1.2. The circles in the figure mark different parts of the document and will be used for reference in the following discussion.

We concentrate on several issues related to the main research goal. One issue is the structural and semantic analysis of Web documents, which is a challenging task because documents on the Web are generally prepared for visual access and for browsing by Internet users. Automatic analysis of them is rarely considered when they are authored. However, in order to improve summarization, both document content and document structure need to be automatically analyzed and semantically exploited. Traditionally, Web documents are prepared in HTML (Hypertext Markup Language) format whose primary purpose is presentation of data which brings limitations when a semantic interpretation of document content is desired. To eliminate this problem, semantic markup languages such as XML (Extensible Markup Language) have been developed. However, HTML documents still dominate the Web; our analysis on Google results with respect to document types showed that there exist nearly 6.1 billion HTML pages but only 52 million XML pages. Therefore, better methods for processing HTML documents are needed. In this thesis, we address the particular problem of finding the sectional hierarchy of an HTML document, where a document can be considered as consisting of sections and subsections with corresponding headings and subheadings.

The document in Figure 1.2 corresponds to the fifth extract (snippet) in Figure 1.1. By looking at the Google snippet, which is a linear concatenation of some document fragments containing the query terms, it is hard to determine whether it is a relevant document with respect to the query. However, if the context of the searched terms is made explicit using structural clues and heading hierarchy, we expect that the users can more easily make a decision on its relevancy. In the example, the relevant part resides near the end of the document under the heading *Antibiotic sensitivity*. Extracting sentences from that part and showing them under that subheading in the summary, as well as displaying other parts and headings, may help the user to decide that this is actually a relevant document.



Antibiotics and bacterial diseases

bacteria – what are they?

FishDoc Home

Disease Treatments Home

Library Topics
Fish Diseases
Disease Treatments
Koi Pages
Filtration
Case Histories
Water Quality
Fish Disease Movies

RelaSted Topics
Fish disease, diagnosis and treatments

Like little chemical factories

Bacteria are minute organisms invisible to the naked eye. There are many different types of bacteria and they are probably the most diverse group of organisms on the planet – each like a little chemical factory. Very few species are pathogenic (disease-causing). Those that are harmful can be divided into two basic types; Gram-negative and Gram-positive. This simple grouping is based on a staining technique in which Gram-negative bacteria stain red, while the Gram-positives stain blue because of differences in the cell-wall structure.

Bacteria stained using Gram differential staining method

Stained Gram-positive bacteria viewed at high magnification

Stained Gram-negative bacteria viewed at high magnification

Antibiotic sensitivity

Bacterial resistance is an issue that needs to be considered when choosing an appropriate antibiotic. While it may be necessary to start a course of treatment based on personal experience, it is also prudent to have bacterial identification and antibiotic sensitivity tests carried out at the same time. At worse this simply confirms the treatment choice, while possible saving valuable time and money if the initial course of action is ineffective.

A sterile swab is used to take a bacterial sample from either the posterior kidney or other organ of a newly dead fish, or alternatively from a body ulcer. When taking a swab of a body ulcer it should be sampled from the infected rim of the ulcer only.

The swab is then sent to a specialist laboratory for analysis and testing. Your local veterinarian can arrange this. The results from a body ulcer can be difficult to interpret because of the problem of secondary pathogens. It is quite common for samples from body lesions to show a mixed growth of several bacterial species and takes some experience to determine the relevance of the results.

Fishdoc would like to thank the following benefactors for their sponsorship of this site:
Suburbanponds.com - Sho Koi - Aquadyne Bead Filtration - KoiStuff.com - Koizyme (Genesyz Aeromonas Fighter

Figure 1.2. Some parts of an example Web document

In Figure 1.2, the boundaries of tables (<*table*> tag in HTML) that visually divide the document into parts are shown as dotted lines. Such structural clues from HTML tags can be utilized to identify sections and subsections. However, it should be noted that such tags do not always correspond to meaningful division of content. Additionally, it is necessary to distinguish the main content from secondary parts such as menus (e.g. (2) in Figure 1.2), advertisements, etc. Also, headings and subheadings like (1), (a), (d) should be distinguished from other text (e.g. (b) in Figure 1.2). These are nontrivial tasks due to the underlying HTML format.

Another issue we consider is the use of the structural and semantic information (i.e. document sectional hierarchy and content) during the summarization process as well as in the output summaries. This involves the identification of the importance of sections and subsections in a document given the user query. Important sections (i.e. sections with high scores) and corresponding headings should be more heavily represented in the output summary. This is contrary to the traditional summarization approaches where a document is considered as a flat sequence of sentences. This task also involves query-biased summarization techniques.

1.2. Outline of the Thesis

The rest of this dissertation is organized as follows. In Chapter 2, we give a general literature survey on search engines, query types, automatic analysis of documents and automatic summarization. This is followed by an overview of the proposed system in Chapter 3, including the main approach, system architecture, implementation and data collection. Then, the proposed rule-based and machine learning approaches for structural processing of documents are detailed in Chapter 4 and Chapter 5, respectively, together with the performance evaluations. In Chapter 6, the proposed summarization method is presented which is based on the output of the structural processing stage; i.e., query-biased and structure-preserving summarization. The chapter also includes the task-based evaluations of the system comparing it with Google and unstructured summaries. Finally, the thesis overview and conclusions are given in Chapter 7.

2. LITERATURE SURVEY

In this chapter, we first give a brief overview of search engines and query types. This is followed by a survey on automatic document analysis. Then, we present background information and related work on automatic summarization.

2.1. Search Engines and Query Types

Information retrieval discipline deals with the storage, retrieval and maintenance of information. The general objective of an information retrieval system is to minimize the time spent by a user in locating the needed information [15, 16]. Information retrieval on the Web, although it is a variant of classical information retrieval, shows significant differences when compared with traditional text retrieval systems. These differences mainly stem from a number of typical characteristics of the Web such as its distributed architecture, the heterogeneity of the available information, its size and growth rate, and so on [17, 18].

One of the major components of a Web information retrieval system is the searching component (the search engine). A search engine allows the user to enter search terms (queries) that are run against a database and retrieves from its database Web pages that match the search terms. We can identify several types of query form supported by modern search engines [16, 17, 19, 20]. The basic and most-widely used mechanism is Boolean search, where one or more keywords separated by (implicit or explicit) Boolean operators are entered. Phrase search is a variant of Boolean search, in which the user requires a set of contiguous words to be found in the given order. A generalized form of this idea is proximity search in which a sequence of words or phrases is given together with a maximum allowed distance between them. Another mechanism is limiting the search with some restrictions on the search items or on the Web page properties. The most common types are range searching (specifying a range for a word) and field searching (restricting the content of a field such as the title, language or file type to a particular value).

Some of the search engines support a number of more advanced query types [16, 20]. Natural language search is a generalization of Boolean search, where any reference to Boolean operators is eliminated and the user formulates the query as a question or a statement. The search algorithms underlying this model of searching need to be quite different from those of simple searching models. In thesaurus search, the search terms supplied by the user are expanded to also include similar words or concepts. Finally, fuzzy search refers to the capability of the system of handling the misspellings and variations (e.g. stemmed form) of the same words.

The information needs of users can be classified based on three dimensions: the intentionality or goal of the searcher, the kind of knowledge currently known by the searcher, and the quality of what is known [21]. In the case of well-defined knowledge of the user, specific information sources are searched, whereas in ill-defined (muddled) cases, the search process is rather exploratory. In a related work, a summarization system is evaluated based on four types of information need in Web search: search for a fact, search for a number of items, decision search, and background search [22].

It is worth mentioning some statistical figures about search engines in order to better evaluate their capabilities. Researchers estimate that the content of the Web is doubling each year. Search engines can index only a fraction of the Web. A research dated 2003 reported that the largest search engine at that time (Google [3]) indexes about 3.8 billion Web pages, which corresponds to only 16% of the Web [17]. Due to the huge size of the Web, response time is a critical factor for search engines. When a query is given by a user, a search engine normally scans all the pages indexed. However, in order not to have an adverse effect on the response time, usually a few of the first result screens are generated more quickly (e.g. by using some templates), exploiting the fact that most of the users do not pass beyond a few screens [23].

2.2. Automatic Analysis of Documents

Information retrieval generally focuses on documents in electronic form which are prepared for access and browsing by humans. Automatic analysis of documents can be useful to enhance this process. In this section, we consider the general problem of document analysis and review the state of the art in Web document analysis.

2.2.1. General Document Analysis

In recent years, there has been a dramatically increasing usage of documents in electronic form which have several advantages over traditional paper documents, such as easy maintenance, efficient retrieval and transmission. Electronic documents can also be processed and utilized structurally. They can be partitioned into physical components, such as paragraphs, words, figures, etc., and logical components such as titles, authors, sections, etc. This structural information can be used in indexing and retrieving useful information contained in the documents. As a result, there have been several studies for the conversion of paper documents into electronic form and the automatic analysis of document structure [8].

In general, document structure analysis can be considered as a syntactic analysis problem [8]. The order and containment relations between physical and logical components of a document can be represented as an ordered tree. A tree grammar can be used to describe a document as consisting of regions or blocks. This is analogous to a sentence in syntactic parsing which can be described as a tree with grammatical relationships among its words. Therefore, some syntactic analysis approaches in natural language processing can be adapted to the document analysis problem. In [24], transformation-based learning method which was previously applied to the syntactic parsing problem is adapted for the conversion of HTML (Hypertext Markup Language) documents into XML (Extensible Markup Language) format.

In a related work, the logical structure of a document is represented with a generalized n-gram model based on a statistical approach [25]. The document structure is constructed in a hierarchical way where local tree node patterns are defined similar to n-grams in natural language processing. The tree patterns for $n = 3$ are given in Figure 3.1. As seen in the figure, the context of a node is represented by its local neighbors such as its siblings, ancestors or descendants. The document logical tree is incrementally built using best-first search.

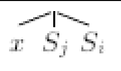
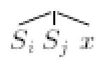
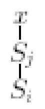
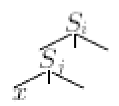
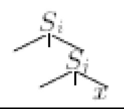
Tree pattern	Relation between x and S_i, S_j	Notation
	The node x is the left neighbor of the pattern S_j S_i.	$L(x, S_j, S_i)$
	The node x is the right neighbor of the pattern S_i S_j.	$R(x, S_j, S_i)$
	The node x is the ancestor of S_j, which is the ancestor of S_i.	$A(x, S_j, S_i)$
	The node x is the first descendant of S_j which is the first descendant of S_i.	$B(x, S_j, S_i)$
	The node x is the last descendant of S_j which is the last descendant of S_i.	$E(x, S_j, S_i)$

Figure 2.1. The notation used to represent tree patterns [25]

Another approach is to model the hierarchical segmentation of a document by means of a grammar [26]. Then, the document analysis problem is converted to the problem of finding the optimal parse of the document given the grammar. In Figure 3.2, an example grammar that can be used to describe printed pages is given where a page consists of paragraphs, a paragraph consists of lines and so on. However, such pure probabilistic context free grammars (PCFGs) have some limitations such as inclusion of feature information. To overcome this limitation, the grammar can be replaced by an attributed grammar; e.g. replacing the *paragraph* non-terminal by *paragraph(lMargin, rMargin, linespace, justification)*. Nevertheless, continuous attributes are problematic and the grammar is generative. As an alternative, non-generative (i.e. discriminative) grammars allow much more powerful models of terminal dependencies without an increase in grammar complexity. Then, in learning, a set of parameters can be estimated which assign high scores to correct grammatical groupings and low scores to incorrect groupings [26].

A recent approach in syntactic analysis is incremental parsing using beam search [27]. In that approach, the parse tree is incrementally constructed and simple corrective updates are performed to the parameters during training. The heuristic approach of beam search is incorporated to reduce the size of the exponential search space of possible parses. A similar approach has been applied to the problem of automatically generating the table-of-contents for a long document (e.g. a book) [28]. In that study, a hierarchical discriminative approach which consists of a local and a global model is developed. It is assumed that the hierarchical segmentation of the document is provided in the input. In the local model, a list of candidate titles for each document segment is generated together with their individual likelihoods. Then, in the global model, the table-of-contents is incrementally created using beam search together with the information obtained in the local model.

```
(Page → ParList)
(ParList → Par ParList)
(ParList → Par)
(Par → LineList)
(LineList → Line LineList)
(LineList → Line)
(Line → WordList)
(WordList → Word WordList)
(WordList → Word)
(Word → terminal)
```

Figure 2.2. An example grammar to describe printed pages [26]

2.2.2. Web Document Analysis

Structural and semantic analysis of Web documents is a relatively young field of research. Web documents are usually encoded in HTML [29] and can contain rich structural information. However, since HTML is mostly concerned with the presentation of content, it does not always correspond to the semantics of the data. As a result, Web documents are usually considered as “semi-structured” documents [30].

One of the motivations in Web document analysis is to filter important content from Web pages by eliminating advertisements and other cluttered parts which are very common to Web pages [31]. Another motivation is to convert HTML documents into semantically-rich XML documents to be utilized later [32]. This analysis may also be used for obtaining a hierarchical structure for the document including its sections and subsections [33, 34, 35, 36]. Some of this work is motivated by the need of displaying content in small-screen devices such as PDAs (Personal Digital Assistants) [34, 35], while others leave the usage open, including more intelligent retrieval of information, summarization, etc.

Most of the related work concentrates on exploiting HTML tags for the analysis; some of them do the analysis by building the explicit DOM (Document Object Model) [37] tree. The approaches used are mostly either rule-based or machine learning-based. Moreover, some of them target a certain domain such as resume documents [32], whereas others are domain-independent. In the following subsections, we give a brief overview of Hypertext Markup Language and Document Object Model, and present the rule-based and machine learning-based approaches in the literature.

2.2.2.1. Hypertext Markup Language and Document Object Model.

Hypertext Markup Language (HTML) is the universal publishing format on the World Wide Web [29]. It allows web authors to publish online documents with headings, text, tables, lists, photos, etc., and allows retrieval of documents via hyperlinks. It also enables the use of forms for conducting transactions with remote services such as searching for information or ordering products.

HTML documents are composed using markup tags (e.g. <*font*>), attributes, attribute values (e.g. the size attribute in <*font size = 3*>), and text. Start and end tags (e.g. <*p*> and </*p*>) are used to annotate content. In HTML syntax, there are two main types of tag: container tags (<*table*>, <*td*>, <*tr*>, etc.) which include other HTML tags or text, and format tags (<*b*>, <*font*>, <*h1*>, <*h2*>, etc.) which are usually concerned with formatting of the text. In most of the Web documents, the organization of content is achieved by the use of container tags; mostly <*table*> and related tags such as <*tr*> (corresponding to table rows) and <*td*> (corresponding to table cells). Also, multiple levels of such tags may be used in a nested way to obtain a complex organization. Other tags to group related content include <*div*> to define a section in a document, <*span*> to group text level elements, and <*li*> to define a list.

The Document Object Model is a platform and language independent interface and allows programs to dynamically access and update the content, structure and style of documents [37]. The DOM presents documents as a hierarchy of nodes, which is referred as the DOM tree. In Web document analysis, DOM tree of a Web document has started to be increasingly utilized because it provides a more global view for the document structure [30].

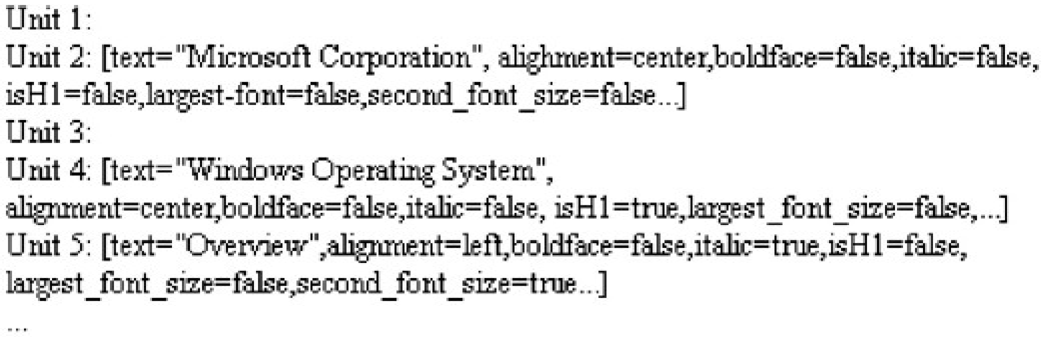
2.2.2.2. Rule-Based Approaches. The DOM tree can be used to extract useful content from Web documents by eliminating cluttered parts. One approach is to navigate the DOM tree with different filtering techniques to remove and adjust specific nodes and thus to leave only the useful content [31]. In [32], HTML pages in a specific domain are converted into semantically-rich XML documents to be utilized in later processing. For this purpose, the document trees are transformed by making use of concepts related to the domain and document restructuring rules. The accuracy of the system was evaluated based on the number of wrong parent-child and sibling relationships in the obtained document hierarchies.

HTML documents can be decomposed into coherent segments in a flat or hierarchical way. In hierarchical structure analysis, the document is processed to obtain a hierarchy of segments and subsegments. One approach to obtain the hierarchical structure is to group visually similar objects, such as document parts with similar formatting [33]. Alternatively, the document can be partitioned iteratively into smaller blocks by detecting separators such as table borders and blank space between contents [34]. This analysis can also be performed by partitioning the document into semantic textual units and arranging the hierarchy based on emphasis differences between the units (e.g. the use of smaller or larger fonts) [35]. Other approaches to hierarchical structure analysis include application of a string matching algorithm on the DOM tree paths [36] and using presentation regularities [38]. Our research differs from all these studies in the sense that we concentrate on section and subsection headings and make use of these in building the hierarchy, whereas other studies do not particularly concentrate on heading-based hierarchy.

2.2.2.3. Machine Learning Approaches. In the literature, there is some related work on the extraction of the main title from documents in electronic form using machine learning techniques. In [39], classification techniques such as support vector machines (SVMs) and conditional random fields (CRFs) are used to extract the main title (i.e., a single title) from the content of Web documents. The features are obtained from a DOM tree based method (such as tag and formatting information) and from a vision-based method used to segment the rendered Web document. After the title extraction, its use in document retrieval was evaluated. In this thesis, we investigate a more general problem than the extraction of the main title; i.e., the problem of finding all the headings in a Web document together with the underlying hierarchy.

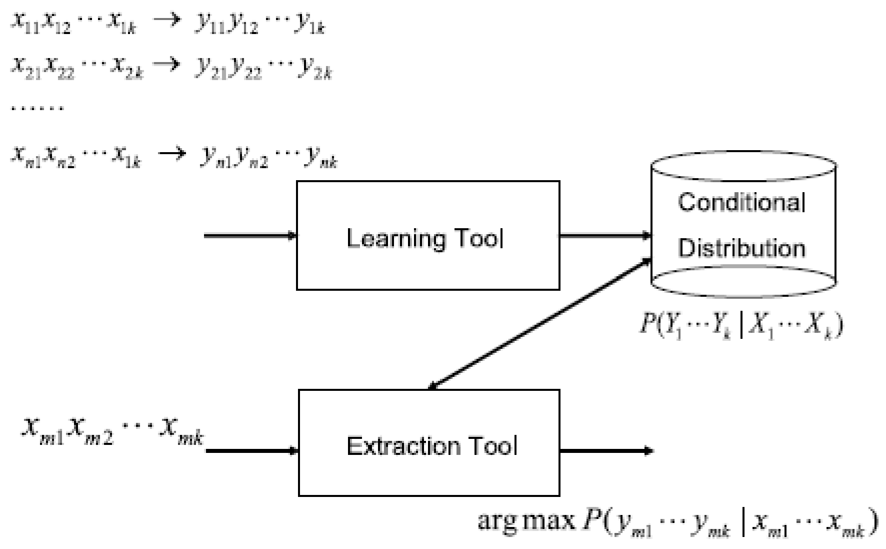
In another machine learning-based work, the aim is to segment a given Web document into blocks [40]. Binary classification (e.g. SVM) with DOM tree and formatting features is used to determine whether two consecutive text nodes in the document constitute an information block boundary or not. For each pair of text nodes, a set of features is defined to represent the distance and the difference of them, such as the difference between their formatting. The Web document is segmented in a flat way without considering the hierarchical structure. Then, the blocks are classified into semantic categories, such as page title, form and menu.

A document is modeled as a sequence of consecutive text units in [39] and [40]. Each unit usually corresponds to a line in the document with its particular formatting and feature set as in Figure 2.3; a unit may also have no textual content [39]. The model for the extraction of the main title is given in Figure 2.4. The learning tool takes documents as input, each as a sequence of units $x_{i1}, x_{i2}, ..., x_{in}$ aligned with a sequence of labels $y_{i1}, y_{i2}, ..., y_{in}$ denoting whether a unit is the main title or not. Then, in extraction, a previously unseen document, given as a sequence of units, is assigned with a sequence of labels. Based on this output, the main title of the document is determined.



```
Unit 1:
Unit 2: [text="Microsoft Corporation", alighment=center,boldface=false,italic=false,
isH1=false,largest-font=false,second_font_size=false...]
Unit 3:
Unit 4: [text="Windows Operating System",
alignment=center,boldface=false,italic=false, isH1=true,largest_font_size=false,...]
Unit 5: [text="Overview",alignment=left,boldface=false,italic=true,isH1=false,
largest_font_size=false,second_font_size=true...]
...
```

Figure 2.3. A sequence of units in an example HTML document [39]



$x_{11}x_{12}\cdots x_{1k} \rightarrow y_{11}y_{12}\cdots y_{1k}$
$x_{21}x_{22}\cdots x_{2k} \rightarrow y_{21}y_{22}\cdots y_{2k}$
$\cdots\cdots$
$x_{n1}x_{n2}\cdots x_{1k} \rightarrow y_{n1}y_{n2}\cdots y_{nk}$
Learning Tool
Conditional Distribution
$P(Y_1\cdots Y_k \mid X_1\cdots X_k)$
$x_{m1}x_{m2}\cdots x_{mk}$
Extraction Tool
$\arg\max P(y_{m1}\cdots y_{mk} \mid x_{m1}\cdots x_{mk})$

Figure 2.4. A title extraction model [39]

In [41], the problem of identifying important blocks within a Web document is considered. First, a vision-based page segmentation algorithm is used to partition the document into blocks with a hierarchical structure. Then, SVM and neural network methods are used with spatial and content features of the blocks to assign importance values to the blocks.

2.3. Automatic Summarization

2.3.1. Background Information

Automatic summarization can be defined as the process of distilling the most important information from a source (or sources) to produce a shortened version for particular users and tasks [1, 42]. There are several uses of automatic summarization in today's information world. Firstly, it can be used as an aid for browsing large documents one by one or sets of documents. Next, it can be utilized in sifting process as an aid to locate useful documents in a large collection. Also, automatic summarization can aid report writers by providing abstracts.

Automatic summarization is related to and influenced by the research in information retrieval and information extraction, where the former is concerned with finding documents whereas the latter with finding useful information inside documents. In fact, summarization can be considered as a special kind of "information extraction" where the summary is the extracted information [15]. Another field related to summarization is text mining which differs from information retrieval and extraction by creating new information from existing information. In this context, summarization constitutes a borderline case. It cannot be considered as a text mining process when it simply extracts information from a document. However, if it provides critical review or links to documents not referenced in the original text or if it synthesizes information (as in cross-document summarization), it can be considered under the field of text mining.

There are several types of summaries. One distinction is made between extracts and abstracts. An *extract* is formed simply by extracting words or sentences from the source text, whereas forming an *abstract* involves reformulation of information in a text using deeper techniques such as natural language generation. Summaries can also be categorized as *generic* or *query-relevant*. Generic summaries are general-purpose summaries which do not focus on a particular topic, whereas query-relevant summaries are formed considering the requirements of a particular user query. Query-relevant summaries can be useful for large documents (e.g. manual or textbook) and documents containing diverse subject matter (e.g. court opinions). Furthermore, summaries can be *single-document* or *multi-document*. In the multi-document case, a single summary is obtained by considering the contents of a set of documents. Another classification is related to the function of the summary. A summary can be *indicative*, that is it can only briefly indicate the topics addressed in the text, or it can be *informative*, covering the concepts of the text in a more detailed manner.

Automatic text summarization can be described in three phases as in Figure 2.5 [43]. In the first phase, the input text is analyzed. In the second phase, it is transformed into a summary representation. Finally, in the third phase, an output summary is generated, i.e. synthesized, using the summary representation. During these phases, three different types of *condensation operations* can be applied to obtain a condensed form of the text. These are *selection* of important and non-redundant information, *aggregation* of information and *generalization* of specific information [42, 43].

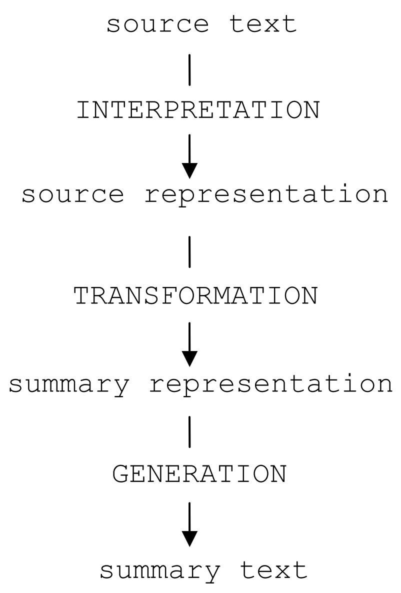


Figure 2.5. Summarization stages [43]

There may be several ways of categorizing automatic summarization approaches. One useful way is to categorize them according to the level of processing as *surface-level*, *entity-level* and *discourse-level* approaches. There may also be hybrid versions of these three approaches [1].

Surface-level approaches usually use shallow features to identify salient (i.e. important) information in the text. These include thematic features (e.g. based on term frequency statistics), location (e.g. position in text or paragraph, section depth, particular sections), background (e.g. presence of terms from title, headings, initial part of text or user query), cue words and phrases (e.g. a sentence beginning with the phrase "in summary") [1, 6, 44].

Entity-level approaches build an internal representation of the text by modeling text entities and their relationships. They usually use the connectivity of entities, e.g. using graph topology, to determine what is salient in the text. Different types of relationships between entities include similarity (e.g. vocabulary overlap), proximity (distance between text units), co-occurrence (words occurring in common contexts), thesaural relationships (e.g. WordNet [45]), coreference, logical relations (e.g. agreement, contradiction), syntactic relations (e.g. based on parse trees) and meaning representation-based relations (e.g. predicate-argument relations).

Discourse-level approaches model the global structure of the text and its relation to communicative goals. These include the format of the document (e.g. hypertext markup or document outlines), threads of topics as seen in the text and rhetorical structure of the text (e.g. argumentation or narrative structure).

Evaluation of automatic summarization methods is another important area of research. Methods of evaluation are usually categorized into two: *intrinsic* and *extrinsic* evaluation. In intrinsic evaluation, the summary itself is evaluated. This can be done either by direct user judgments of the quality or by calculating the similarity to an "ideal" summary. Intuitively, the informativeness, i.e. the extent important information is preserved in the summary, and coherence, including the readability of the summary, are two different measures on the quality of a summary [46]. The measures used in most of the practical systems are precision and recall which are used to determine the similarity of the produced summary with the "ideal" summary. Other measures include Kappa, relative utility, cosine similarity and longest common subsequence. In extrinsic evaluation (i.e. task-based evaluation), the quality of a summary is determined based on how it affects the completion of another task as in [47]. In ad hoc task, the aim is to correctly identify the relevance of a document with respect to a given topic by using the summary. In categorization task, the aim is to correctly categorize a document with respect to a given set of topics by using the summary. Measures like time spent and accuracy obtained during these tasks can be used to assess the quality of the summary.

2.3.2. Related Work

Early research in automatic summarization began in the late 1950s mainly with a surface-level approach [6]. The first entity-level approaches started in the early 1960s. The interest in the field renewed in the early 1970s, extensions are made to the surface-level approach, and first commercial applications were developed. In the late 1970s, more extensive entity-level approaches were developed and first discourse-based approaches appeared. The 1980s saw a variety of different work including entity-level approaches based on artificial intelligence concepts. The late 1990s were described as the renaissance of the field. During this period, all the three types of approaches were considered with government and commercial interest. The focus was on extracts rather than abstracts with renewed interest in surface-level approaches. Also, new areas of research including multi-document summarization, multilingual summarization and multimedia summarization began to be developed. Automatic summarization research has also been influenced by the studies of abstracting behavior, including the psychological study of human summarization in the laboratory and the study of professional abstractors [1]. The last decade showed a valuable progress in the field due to the rapid growth of publicly accessible text on the Web and evaluation programmes such as DUC (Document Understanding Conferences) [43]. The recent studies include mostly extractive approaches using statistical and/or shallow symbolic methods.

In the following subsections, the main approaches in automatic summarization are considered with representative systems from the literature. These include classical, corpus-based and discourse-based approaches. Then, more recent work is presented in the last subsection.

2.3.2.1. Classical Approaches. Luhn's paper is one of the oldest work in automatic summarization [6]. It describes an algorithm which scores sentences based on term frequencies and extracts highest scored sentences. The algorithm filters terms using a stop list containing closed-class words such as pronouns and articles. Also, some kind of normalization is applied on terms, e.g. the words "similar" and "similarity" are aggregated together, and low-frequency terms are removed. Luhn's basic statistical approach has had significant influence to the subsequent research in automatic summarization. Edmundson extended earlier work by using three other features in addition to word frequencies [44]. These are cue phrases (e.g. "significant", "impossible", etc.), title and heading words, and sentence location.

2.3.2.2. Corpus-based Approaches. In one of the representative works in the literature, extraction task is approached as a statistical classification problem [48]. The system uses a training set of documents and corresponding human abstracts. Each sentence in the training set is labeled based on whether it is included in the corresponding summary or not. Then, a classification function is defined in order to estimate the probability that a given sentence will be included in the summary. For this purpose, each sentence is described by a set of features such as sentence length, sentence position in the paragraph, presence of high-frequency content words, etc. Each sentence in an input text is ranked according to the calculated probability and a user-specified number of top scoring sentences are selected to be incorporated into the summary. This study has been followed by several other corpus-based studies in this field [1].

2.3.2.3. Discourse-based Approaches. Discourse-based approaches can be classified into two categories based on the distinction between cohesion and coherence [1]. *Cohesion* involves relations between words or referring expressions and is related to how tightly the document is connected. Such relations may include anaphora, ellipsis, conjunction and lexical relations such as synonymy and hypernymy. The work by Barzilay and Elhadad is based on lexical cohesion [49]. *Coherence* corresponds to macro-level relations between sentences or clauses; e.g. clauses linked by “although” have a contrast relation. Marcu’s work uses a coherence model based on rhetorical structure theory [50]. Based on several earlier psycholinguistic studies, Marcu states that the structure of a text is essential in summarizing the text [51].

2.3.2.4. Recent Work. Sparck Jones overviews the current state of automatic summarization, and suggests methodologies and research strategies for this developing field in her position papers [7, 43]. Sparck Jones underlines the importance of context factors in summarization research. The proposed methodology is to determine the operational factors for individual cases summarization is used. There are three types of context factors: input factors, purpose factors and output factors [7]. *Input factors* include the form (e.g. structure, genre, language), the subject type (ordinary, specialized or restricted) and the unit (single or multiple sources) of the input. *Purpose factors* are the situation (i.e. the context within which the summary is used), the audience and the use of the summary. *Output factors* include the material (that is, whether the summary covers all the main points or is partial), the format and the style (e.g. informative, indicative, critical, etc.) of the output summary. Within these three factors, the most important ones to consider as a part of research methodology are stated as the purpose factors.

Most of the related work on automatic summarization aim at creating generic summaries rather than query-biased ones and employ extraction methods based on sentence weighting [52, 53, 54]. In a related work, the effects of several sentence weighting schemes were investigated, including sentence length, query term order, and query term frequency [55]. It was found that using a combination of weighting components improves the performance compared to any single component.

Currently available major Web search engines, such as Google [3] and AltaVista [4], use short extracts of document contents (i.e., two or three lines of text) in displaying their results. Google creates document extracts using query-biased techniques. Query words appearing within the document are output together with some of their context; i.e., with leading and trailing non-query terms.

Tombros and Sanderson investigate the advantages of query-biased summaries in information retrieval [56]. In that study, surface-level sentence extraction techniques, such as title, location and query features, are used for summarization. It is shown that the use of query-biased summaries significantly improves both the accuracy and the speed of users in identifying relevant documents. Another related work is WebDocSum which is a retrieval interface providing summaries much longer than those of the traditional search engines to improve the search experience of Web users [22]. Instead of displaying the long summaries under the search results, which may result in a cluttered view, only the link titles are listed and the corresponding summaries are presented in a separate window when the mouse is moved on a particular link. The summarization system was shown to be more effective than the summaries of Google and AltaVista on a task-based evaluation. WebDocSum uses a query-biased and surface-level technique for summarization. In [57], a structure-based and query-specific summarization technique was proposed. This method tries to add structure to a document by dividing it into text fragments (e.g. paragraphs) and connecting related fragments as a document graph rather than making use of the explicit document structure such as the sectional hierarchy. In that study, the summary is formed by a graph search algorithm.

There is few work on summarization based on explicit document structure such as the document sectional hierarchy. In one of the studies, the system builds a “table of content”-

like hierarchy of sections and subsections for each document using heuristics on HMTL tags present in the documents and incorporates this structural information in the output summaries [58]. In fractal summarization method [59], summaries are created based on the hierarchical structure of a long document, including chapters, sections and subsections. These studies focus on general-purpose summaries, not tailored for particular user queries or Web search tasks. To the best of our knowledge, there is no related work on summarization combining explicit structure of Web documents and query-biased techniques.

There exist some studies on summarization of XML documents which are inherently structured. In one of the studies, query-biased summarization was used as an aid for searching XML documents [60]. In another study, a machine learning approach was proposed for summarization of XML documents based on structure and content [61].

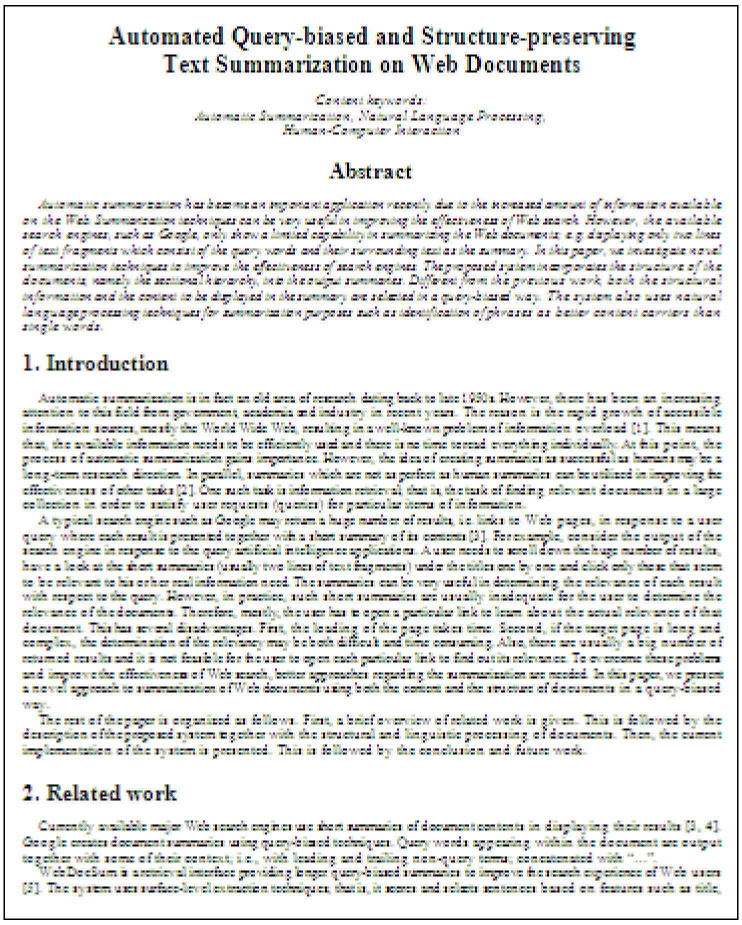
3. QUERY-BIASED AND STRUCTURE-PRESERVING SUMMARIZATION

In this chapter, first, the general approach and the architecture of the proposed system are presented. Then, the implementation of the system and the data collection are overviewed. A detailed description of the most important system modules will be given in the following chapters.

3.1. The Approach

In this thesis, we develop a novel summarization approach in order to improve the effectiveness of Web search. The proposed approach is intended to be used together with a search engine, such as Google and AltaVista. The aim is to enhance the user experience of such search engines which usually show only a limited capability in summarizing Web documents; e.g. by displaying only two or three lines of document fragments which consist of the query words and their surrounding text as the summary. Our main aim is to improve the effectiveness with respect to the information retrieval task; i.e., the task of locating documents which are relevant to a particular search query.

The proposed summarization approach is based mainly on two aspects: document structure and query-biased techniques. First, we consider the fact that Web documents are not flat texts, but they usually contain a structure. Structured documents may have sections and subsections with different topics and formatting, and corresponding headings and subheadings (see Figure 3.1 for an example structured document). Traditional approaches in summarization usually ignore the document structure and treat a document as a linear sequence of sentences. That is, they select the sentences or document fragments to be included in the summaries from this linear space of text, and their summary output is also usually unstructured. However, the use of document structure becomes important on the Web where documents usually have diverse content and formatting. Another aspect we consider is the user query, i.e., the information need of the user as entered in the Web search interface, which is also usually ignored in traditional approaches. However, there is evidence in the literature that a query-biased approach is more suitable to Web search than a generic approach.



Automated Query-biased and Structure-preserving Text Summarization on Web Documents

Content keywords:
Automatic Summarization, Natural Language Processing, Human-Computer Interaction

Abstract

Automatic summarization has become an important application recently due to the increased amount of information available on the Web. Summarization techniques can be very useful in improving the effectiveness of Web search. However, the available search engines, such as Google, only show a limited capability in summarizing the Web documents; e.g. displaying only two lines of text fragments which consist of the query words and their surrounding text as the summary. In this paper, we investigate novel summarization techniques to improve the effectiveness of search engines. The proposed system incorporates the structure of the documents, namely the sectional hierarchy, into the output summaries. Different from the previous work, both the structural information and the content to be displayed in the summary are selected in a query-biased way. The system also uses natural language processing techniques for summarization purposes such as identification of phrases as better content carriers than single words.

1. Introduction

Automatic summarization is in fact an old area of research dating back to late 1950s. However, there has been an increasing attention to this field from government, academia and industry in recent years. The reason is the rapid growth of accessible information sources, mostly the World Wide Web, resulting in a well-known problem of information overload [1]. This means that, the available information needs to be efficiently used and there is no time to read everything individually. At this point, the process of automatic summarization gains importance. However, the idea of creating summaries as successful as humans may be a long-term research direction. In parallel, summaries which are not as perfect as human summaries can be utilized in improving the effectiveness of other tasks [2]. One such task is information retrieval, that is, the task of finding relevant documents in a large collection in order to satisfy user requests (queries) for particular items of information.

A typical search engine such as Google may return a huge number of results, i.e. links to Web pages, in response to a user query where each result is presented together with a short summary of its contents [3]. For example, consider the output of the search engine in response to the query artificial intelligence applications. A user needs to scroll down the huge number of results, have a look at the short summaries (usually two lines of text fragments) under the titles one by one and click only those that seem to be relevant to his or her real information need. The summaries can be very useful in determining the relevance of each result with respect to the query. However, in practice, such short summaries are usually inadequate for the user to determine the relevance of the documents. Therefore, mostly, the user has to open a particular link to learn about the actual relevance of that document. This has several disadvantages. First, the loading of the page takes time. Second, if the target page is long and complex, the determination of the relevancy may be both difficult and time consuming. Also, there are usually a big number of returned results and it is not feasible for the user to open each particular link to find out its relevance. To overcome these problems and improve the effectiveness of Web search, better approaches regarding the summarization are needed. In this paper, we present a novel approach to summarization of Web documents using both the content and the structure of documents in a query-biased way.

The rest of the paper is organized as follows. First, a brief overview of related work is given. This is followed by the description of the proposed system together with the structural and linguistic processing of documents. Then, the current implementation of the system is presented. This is followed by the conclusion and future work.

2. Related work

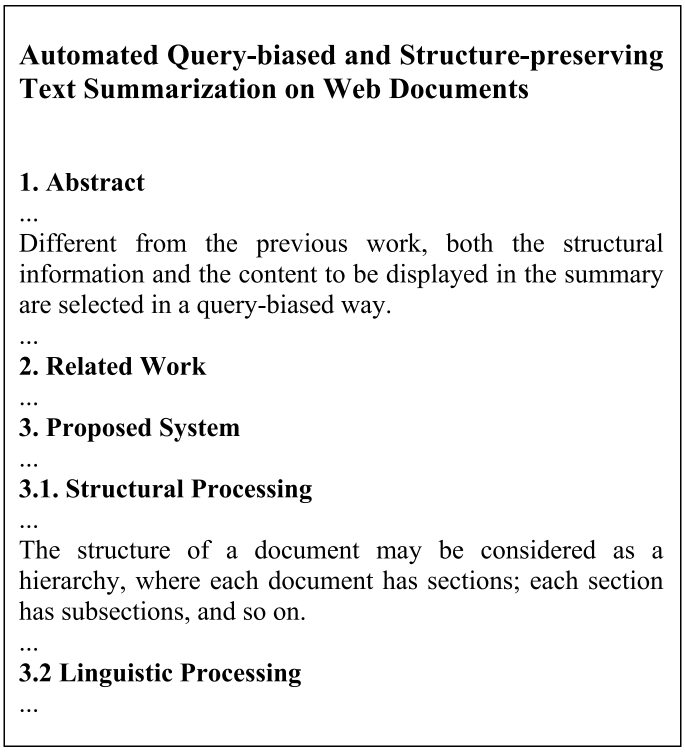
Currently available major Web search engines use short summaries of document contents in displaying their results [3, 4]. Google creates document summaries using query-biased techniques. Query words appearing within the document are output together with some of their context, i.e., with leading and trailing non-query terms, concatenated with "...".

WebDocSum is a retrieval interface providing longer query-biased summaries to improve the search experience of Web users [5]. The system uses surface-level extraction techniques, that is, it scores and selects sentences based on features such as title,

Figure 3.1. An example structured document

The structural information used in the proposed system is the document sectional hierarchy. It is incorporated into the summarization process in two different ways. First, it is used to determine important document sections and subsections based on the user query. Second, some structural information (i.e., headings and subheadings) is also displayed as a part of the output summary in order to provide the context of the text fragments selected as a part of the summary. In this way, the user is expected to judge the relevance of search results better. In Figure 3.2, an example structured summary is given. In the figure, the summary sentences are displayed in a structured way, and '…' is used to indicate the content not selected as a part of the summary.

In the proposed system, indicative summaries are created to direct users to relevant documents rather than informative summaries that can be used as a replacement of the original documents. We use the method of sentence extraction rather than sentence abstraction which involves rewriting. In this way, the structure of the original document and the context of the selected sentences can be preserved and thus the user can judge the relevancy of documents more precisely.



Automated Query-biased and Structure-preserving Text Summarization on Web Documents

1. Abstract

...

Different from the previous work, both the structural information and the content to be displayed in the summary are selected in a query-biased way.

...

2. Related Work

...

3. Proposed System

...

3.1. Structural Processing

...

The structure of a document may be considered as a hierarchy, where each document has sections; each section has subsections, and so on.

...

3.2 Linguistic Processing

...

Figure 3.2. An example structured summary

The summary length is also increased in the proposed system compared to the ones provided by search engines. However, if these longer summaries were again displayed under the corresponding titles in the search results, then the user would need to scroll too much to see the consecutive results. To prevent this problem, similar to a previous work in the literature [22], only the titles of the results are listed, and when the user moves the mouse on a particular link, the summary for that document is displayed in a separate frame. Although the summaries in this approach are much longer than the traditional approaches, they are still limited; e.g., to the size of the area of the screen that can be seen without scrolling.

3.2. System Architecture

The system architecture is given in Figure 3.3. There are two types of roles: the system and the user. In the system part, the HTML documents collected from the Internet are processed in order to obtain their structure; i.e. the document sectional hierarchy (structural processing). For each document, a hierarchical representation is obtained. Both the documents and user queries are processed linguistically; e.g. stemming and recognition of phrases. The user query and the hierarchical representations of the documents are used by the summarization engine to obtain the output summaries.

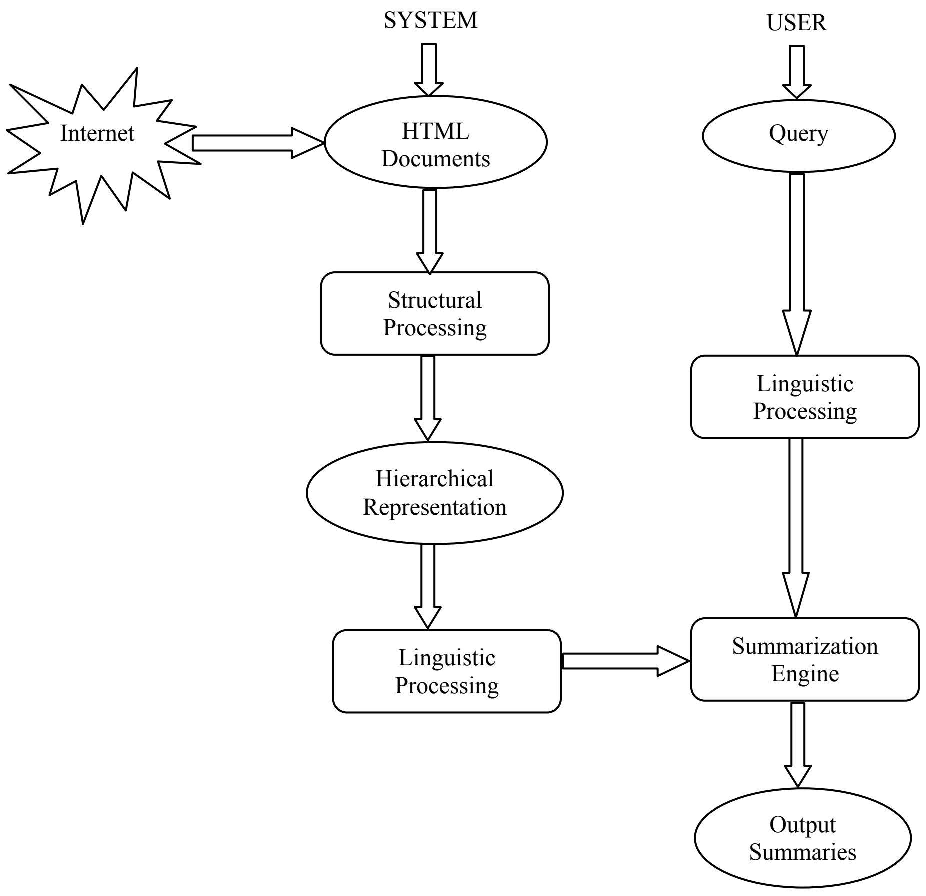


Figure 3.3. The system architecture

3.2.1. Structural Processing

The aim of structural processing is to identify heading-based sectional hierarchy for Web documents. In general, the structure of a document may be considered as a hierarchy where each document may have sections, each section may have subsections and so on, together with the corresponding headings and subheadings. The targeted document format is HTML because it is still the most frequently used format on the Web. The input to the proposed system is a Web document in HTML format. The output is a tree representing the sectional hierarchy of the document where headings and subheadings are at the intermediate nodes and other text units are at the leaves (see Figure 3.4). The root contains a dummy unit covering the whole document. As can be seen, headings in different levels can be identified as a hierarchy together with the sentences under the headings. In Figure 3.5, a part of the sectional hierarchy for an example document is given. The rule-based and machine-learning approaches we developed for structural processing are detailed in Chapters 4 and 5, respectively.

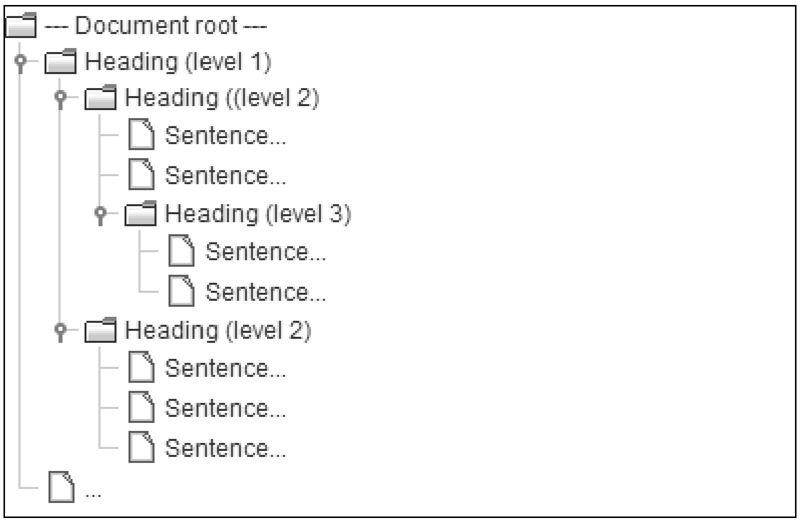


Figure 3.4. Output of structural processing

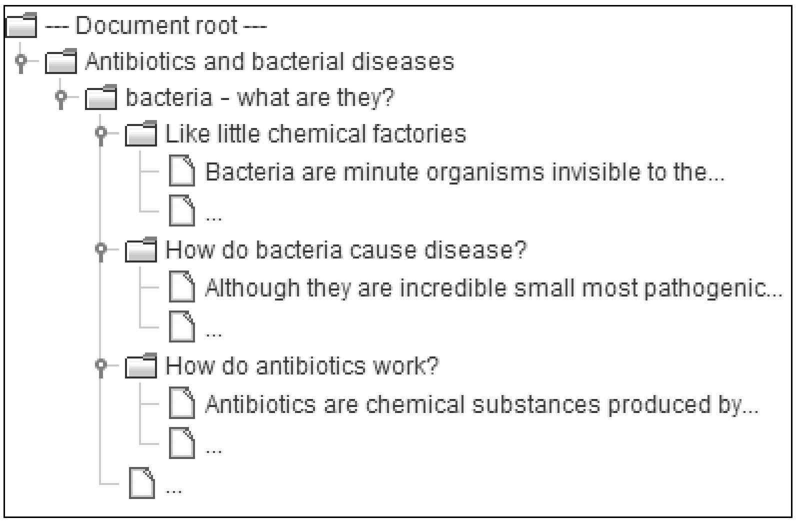

--- Document root ---
Antibiotics and bacterial diseases
bacteria - what are they?
Like little chemical factories
Bacteria are minute organisms invisible to the...
...
How do bacteria cause disease?
Although they are incredible small most pathogenic...
...
How do antibiotics work?
Antibiotics are chemical substances produced by...
...
...

Figure 3.5. Part of the sectional hierarchy for an example document

3.2.2. Summarization

The summarization algorithm is run after the structural processing phase is completed. The algorithm works on the document tree obtained in structural processing and it is based on an extractive approach. The structural properties of documents are utilized during the summarization process and in the output summaries. The proposed summarization approach is detailed in Chapter 6.

3.3. Implementation

The proposed system was implemented in Java programming language. We utilized two different frameworks: GATE Text Engineering Framework and the Cobra Toolkit. In the following subsections, we overview the basic properties of these frameworks and our implementations.

3.3.1. GATE Framework

GATE (General Architecture for Text Engineering) is a framework and development environment for human language technology modules and applications [62, 63]. It is an open source project using component-based technology in Java and is used by several academic and commercial projects. Using such a framework has several advantages because it includes commonly used natural language functionalities such as sentence splitting, part-of-speech tagging and noun phrase identification. Also, it is a modular environment into which new components can be easily added.

GATE distinguishes between data, algorithms and the methods of their visualization. That is, each component in the GATE system is one of the three types: Language resources, processing resources and visual resources. Language resources represent entities such as documents and corpora. Processing resources correspond to algorithmic entities such as parsers and generators. Finally, visual resources are related to the visualization and editing of the components in the graphical user interface. The processing resources work on the language resources, and several processing resources can be run sequentially as a pipeline. The data flow between different processing resources is achieved by the annotation of documents.

GATE supports parsing of HTML documents and it can identify HTML tags in the documents as annotations marked on the documents; however, it does not build the explicit DOM tree. We modified its algorithm in order to extract the DOM tree. In the proposed system, after an HTML document is loaded to the system, the following processing resources (modules) are applied to it in the indicated order and the final summary is generated:

- *Tokeniser* - splits the text into tokens, such as words, numbers and punctuation marks.
- *Sentence Splitter* - splits the text into sentences.
- *Stemmer* - applies stemming to individual words.
- *Part-of-Speech Tagger* - produces a part-of-speech tag on each token
- *Noun Phrase Chunker* - identifies noun phrases
- *HTML Document Structure Analyzer* - applies the proposed structural processing algorithm on the document.
- *Summarization Engine* - runs the proposed summarization method on the document.

The tokeniser and sentence splitter were taken from ANNIE, a GATE implementation of an information extraction system. The stemmer used is a GATE plugin and it is based on the commonly-used Porter stemming algorithm [64]. We used English and Turkish versions of the stemmer. We implemented two new processing resources as plugins to GATE: HTML Document Structure Analyzer and Summarization Engine.

3.3.2. Cobra Toolkit

In the standalone implementation of the system, we also utilized Cobra Java HTML Renderer and Parser Toolkit for parsing the HTML documents and building the DOM trees [65]. Cobra is an open source project in Java and supports HTML 4, Javascript and CSS (Cascading Style Sheets). Cobra provides an API for the navigation of the DOM tree. One advantage of Cobra is the support of Cascading Style Sheets which have been started to be widely used in HTML documents to separate the presentation and content. Cobra can also be used to obtain the visual positions (such as the *x* and *y* coordinates) of individual text units in the rendered Web page.

3.4. Data Collection

In order to evaluate the proposed structural processing and summarization approaches, a sufficiently large and representative corpus of Web documents is needed. There may be several ways to create such a corpus, such as using a list of popular search queries from a search engine, defining queries that reflect current search interests of users, or using standard queries and collections such as TREC (Text Retrieval Conference) [39, 66, 67].

Analyses in the literature about user behavior in forming queries have shown that users do not put much effort into formulating a query and they mostly use very simple Boolean types of query [21, 68]. It was reported that 80% of the queries are formed as a sequence of words without any Boolean operator in between and the average query length is 2.35 words [18]. Another study gives the same average length and estimates the average number of operators as 0.41 operators per query [19]. In addition, it was found that 25% of the users use a single keyword. Chowdhury states that only about 8% of the queries contain

Boolean operators and only 9% use some advanced features [17]. All these results indicate the poor nature of end-user searches. We follow the same approach in simulating user behavior in this work by using Boolean queries having a length of 2-3 words. We selected queries by considering current search interests of users in various domains.

We created three different document collections for the experiments. The first one (*English Collection*) includes English Web documents collected from the results of Google in response to 10 different queries from TREC-2004 Robust Test Set (see queries 1-10 in Table 3.1). For each query, a set of 10 documents were randomly collected from the top 50 results returned by Google, corresponding to a total of 100 out of 500 documents. The second collection (*Turkish Collection*) includes Turkish Web documents collected from the results of Google using TREC-like queries defined for Turkish [69] (Table 3.2). The collection includes 50 documents randomly selected out of 250 documents. For the machine learning algorithms, a larger document collection was needed. For this purpose, we created the third collection (*Extended English Collection*) which is an extension of the original English collection and includes all the 20 queries in Table 3.1. The documents for the collection were collected from the top results of Google in response to each query; i.e., top 25 HTML documents for each query, corresponding to a total of 500 documents. The average document length is about 1566 words in the English collection, 900 words in the Turkish collection and 1340 words in the extended English collection. These document collections were used in the evaluation of both structural processing and summarization methods.

Table 3.1. Queries used for building the English collections

Query ID	**Query Keywords**
1	Hubble telescope achievements
2	best retirement country
3	literary/journalistic plagiarism
4	Mexican air pollution
5	antibiotics bacteria disease
6	abuses of e-mail
7	declining birth rates
8	human genetic code
9	mental illness drugs
10	literacy rates Africa
11	robotic technology
12	creativity
13	tourism, increase
14	newspapers electronic media
15	wildlife extinction
16	R&D drug prices
17	Amazon rain forest
18	Osteoporosis
19	alternative medicine
20	health and computer terminals

Table 3.2. Queries used for building the Turkish collection

Query ID	**Query Keywords**
1	*tsunami* (tsunami)
2	*ekonomik kriz* (economic crisis)
3	*Türkiye'de meydana gelen depremler* (earthquakes in Turkey)
4	*sanat ödülleri* (art awards)
5	*bilişim eğitimi ve projeleri* (IT education and projects)

4. RULE-BASED APPROACH FOR STRUCTURAL PROCESSING

In this chapter, we present the rule-based approach we developed for Web document structure analysis. It is a heuristic approach based on DOM tree processing. This is followed by the experiments to measure the effectiveness of the proposed method.

4.1. The Method

Structural analysis of Web documents is a nontrivial task because the underlying HTML format is not intended for a semantic representation of data. Also, most of the documents on the Web show a cluttered and complex organization with diverse formatting and topics. In this section, we overview the proposed structural processing method to identify the sectional hierarchy for a given Web document; i.e. sections and subsections together with the corresponding headings and subheadings. The method involves three steps which are detailed in the following subsections:

- DOM tree processing: The DOM tree of the document is analyzed to find out the structural properties of the document and internal relationships between text fragments.
- Heading identification: The headings in the document are determined based on heuristics.
- Hierarchy restructuring: The heading information is used to restructure the tree obtained in the first step and to identify the actual hierarchical relationships between sections and headings.

4.1.1. DOM Tree Processing

The DOM tree is a hierarchical representation of an HTML document. However, it primarily concerns the presentation of the document contents and usually does not correspond directly to a semantic hierarchy. Nevertheless, this hierarchical representation may be partly utilized to obtain the semantic organization of a document. In Figure 4.1, a part of the DOM tree corresponding to the example document in Figure 1.2 is shown. The fragments identified with circles were marked for correspondence.

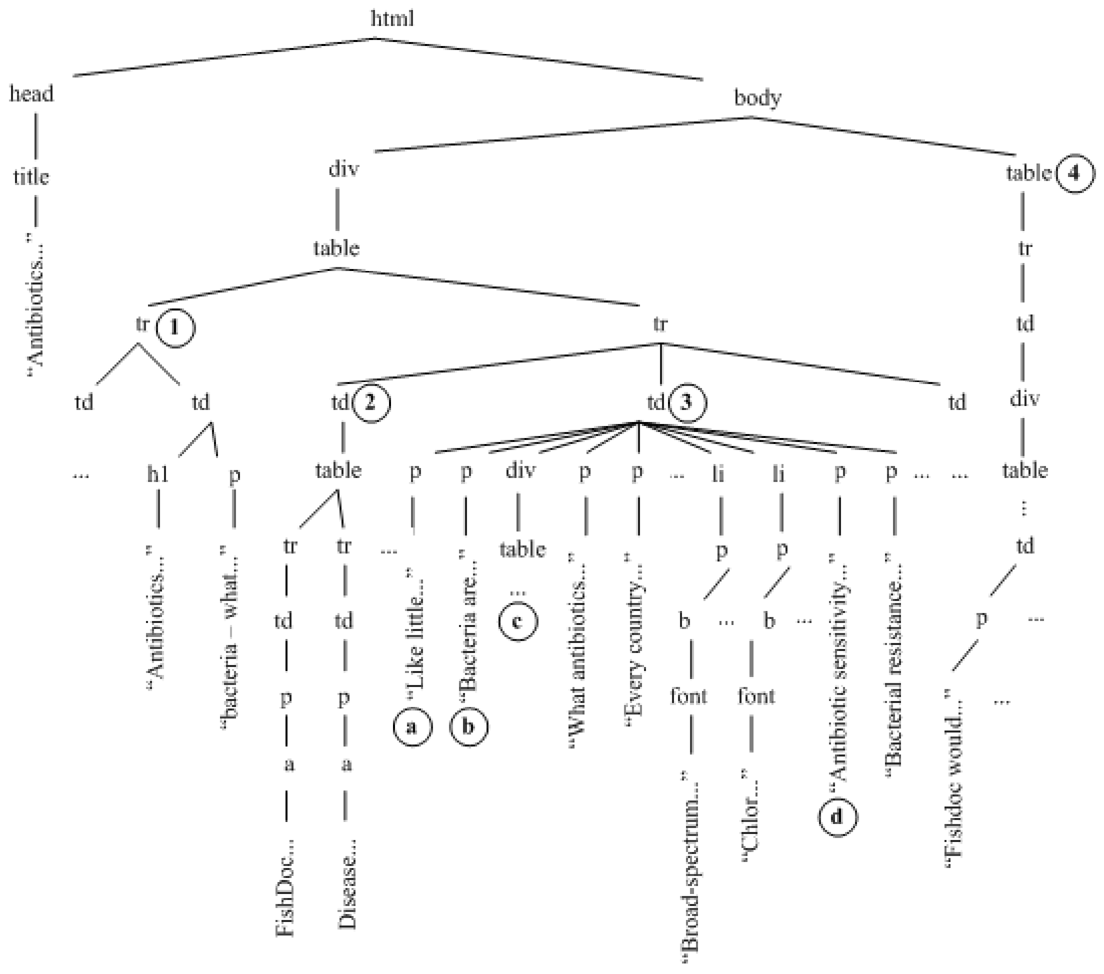


Figure 4.1. Part of the DOM tree for an example Web document

As can be seen in the example tree, semantically related parts like (1), (2), (3), and (4) are grouped under certain container tags such as <*table*>, <*td*>, and <*tr*>. In general, the DOM tree usually has a complex organization with nested container and format tags. The depth of the tree may be quite large; it is common to find DOM trees having 20-30 levels. Also, the DOM tree usually contains tags that do not correspond to textual content. Therefore, the hierarchy that is closer to the sectional hierarchy (usually with much smaller depths) should be distilled from this tree.

The intuition behind DOM tree processing step is that semantically related parts of an HTML document usually occur in the same or neighboring container tags in the hierarchy. The approach we take in the proposed system is to convert the DOM tree into a simplified version of it with only containment relationships (i.e. container tags). In the converted tree, sentence boundaries are also taken into account. The tree is restructured such that each leaf corresponds to exactly one sentence whereas in the original DOM tree, each leaf may correspond to part of a sentence or more than one sentence. The processing involves the following steps:

- Prune nodes that do not contain text in the leaves beneath them and nodes which will not be used in summarization, such as forms or drop-down menus.
- Split/merge leaf nodes such that each leaf node corresponds to exactly one sentence.
- Simplify the tree to obtain the containment hierarchy.

The algorithm to simplify the tree is given in Figure 4.2. It works in a breadth-first fashion starting from the root node. Nodes that have only one child or nodes with format tags (such as <*bold*>, <*font*>) are removed in order to simplify the tree. For this purpose, their children are percolated up and the format tags are passed as features to them. The output of this step for the example DOM tree in Figure 4.1 is given in Figure 4.3. In the resulting tree, document parts are grouped under *block* elements. Also, the main title of the document is identified using the <*title*> tag and it is percolated to the root of the document tree such that all the sections and subsections of the document are stored under the main document title.

```
Algorithm SimplifyTree
Input
    root: root node of the document tree
begin
1:  Insert root into queue
2:  while (queue not empty)
3:      Get the next node n from queue
4:      if ( (the node has only one child)
            or (the node has a format tag, such as <bold>))
5:          Remove the node
6:          Percolate its child(ren) up together with corresponding features
            such as boldness
7:      end if
8:      Insert child(ren) to the queue
9:  end while
end
```

Figure 4.2. Document tree simplification algorithm

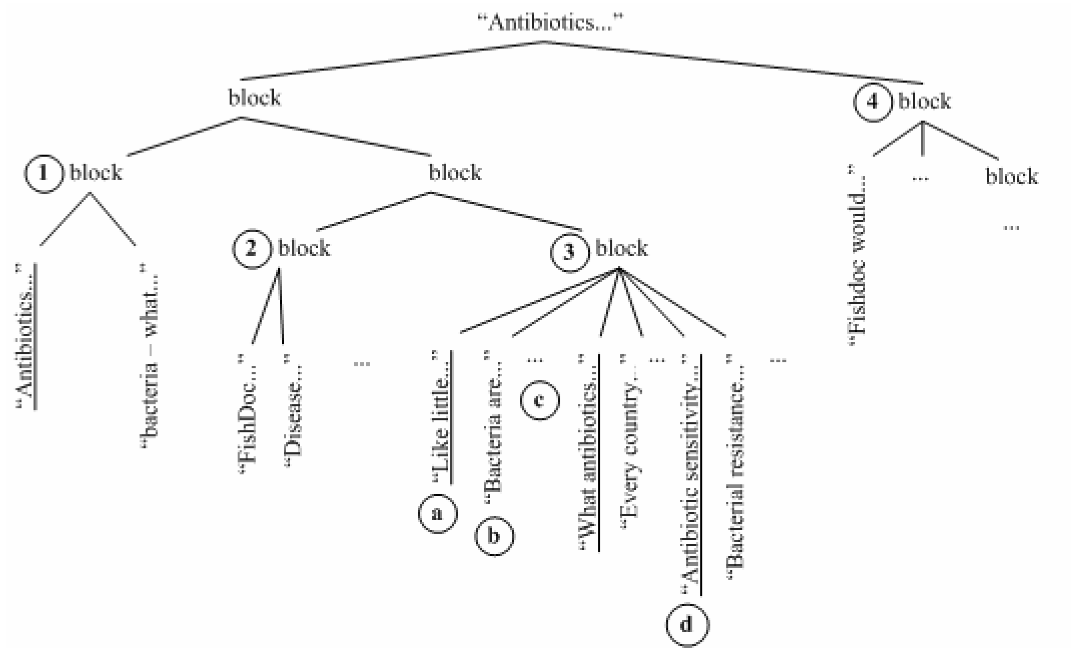


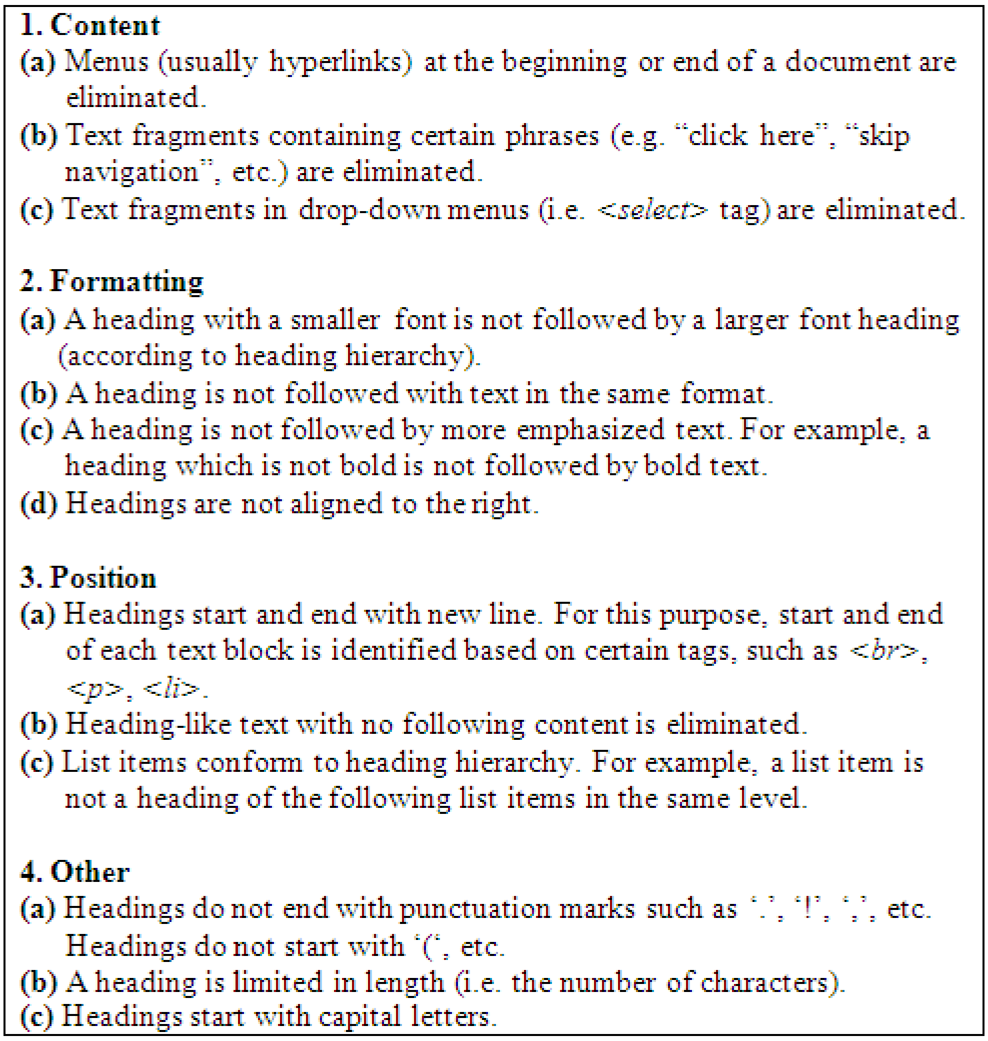
Figure 4.3. The document tree obtained after DOM tree processing

4.1.2. Heading Identification

The aim of this step is to identify all the headings and subheadings in a given HTML document. This is a nontrivial task because of the cluttered structure found in most Web documents. There are usually multiple columns and blocks of content with different formatting styles. Actually, there are six different heading tags (<*h1*> through <*h6*>) in HTML to format different levels of heading. However, these tags are rarely used by Web authors for this purpose. Sometimes, they are even used for formatting non-heading text.

We examined several Web pages (English and Turkish) in order to find out the characteristics that distinguish headings from non-heading text. In most of the documents, the headings are formed by formatting them differently from their surrounding text (e.g. font size, color, boldness, etc.). That is, headings usually have higher emphasis than the text following them in terms of formatting. For instance, the headings (1), (a), and (d) in Figure 1.2 are formed in this way. Also, headings usually do not end with punctuation marks. However, menus or links (e.g. (2) in Figure 1.2) which also do not end with punctuation marks should not be identified as headings. For this purpose, their content and surrounding text should be examined.

In the implementation, we took a heuristic-based approach for heading identification. The heuristics employed are summarized in Figure 4.4 under different categories. These are encoded as *if-then* rules in the system. The headings obtained after the application of the heuristics on the example document are shown underlined in Figure 4.3.



1. Content
(a) Menus (usually hyperlinks) at the beginning or end of a document are eliminated.
(b) Text fragments containing certain phrases (e.g. "click here", "skip navigation", etc.) are eliminated.
(c) Text fragments in drop-down menus (i.e. *<select>* tag) are eliminated.

2. Formatting
(a) A heading with a smaller font is not followed by a larger font heading (according to heading hierarchy).
(b) A heading is not followed with text in the same format.
(c) A heading is not followed by more emphasized text. For example, a heading which is not bold is not followed by bold text.
(d) Headings are not aligned to the right.

3. Position
(a) Headings start and end with new line. For this purpose, start and end of each text block is identified based on certain tags, such as *
*, *<p>*, *<li>*.
(b) Heading-like text with no following content is eliminated.
(c) List items conform to heading hierarchy. For example, a list item is not a heading of the following list items in the same level.

4. Other
(a) Headings do not end with punctuation marks such as '.', '!', ',', etc. Headings do not start with '(', etc.
(b) A heading is limited in length (i.e. the number of characters).
(c) Headings start with capital letters.

Figure 4.4. Heuristics used for heading identification

4.1.3. Hierarchy Restructuring

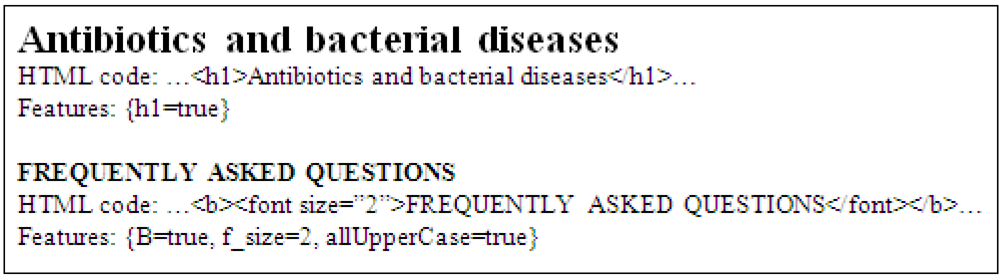
The aim of this step is to restructure the tree resulting from the previous steps to obtain a hierarchy that is closer to the actual sectional hierarchy. Since we chose to concentrate on headings in obtaining the hierarchy, the tree is rearranged making use of the heading information already discovered. In the first step (Section 4.1.1), each sentence (including headings) was identified with formatting features as given in Table 4.1. The features may have Boolean values such as whether or not a text fragment is annotated with heading tags (*<h1>*, *<h2>*, etc.), integer values such as the font size, and string values such as the font face or CSS (Cascading Style Sheets) class.

Table 4.1. Features used for identifying the format of the text

Feature	**Description**	**Data Type**
h1	*<h1>*, level-1 heading	Boolean
h2	*<h2>*, level-2 heading	Boolean
h3	*<h3>*, level-3 heading	Boolean
h4	*<h4>*, level-4 heading	Boolean
h5	*<h5>*, level-5 heading	Boolean
h6	*<h6>*, level-6 heading	Boolean
B	*<b>*, bold	Boolean
strong	*<strong>*, strong emphasis	Boolean
em	*<em>*, emphasis	Boolean
A	*<a>*, hyperlink	Boolean
U	*<u>*, underlined	Boolean
I	*<i>*, italic	Boolean
f_size	*<font size=…>*, font size	Integer
f_color	*<font color=…>*, font color	String
f_face	*<font face=…>*, font face	String
allUpperCase	all the letters of the words are in uppercase	Boolean
cssId	CSS id attribute if used	String
cssClass	CSS class attribute if used	String
alignment	align attribute	String
li	*<li>*, different levels of list elements	Integer

We use these features in differentiating between different levels of heading in the hierarchy. The idea is that in a semantic block of text, headings in the same level usually have the same features and the sectional hierarchy is achieved with distinct formatting for different levels of heading. In Figure 4.5, headings that correspond to two different levels in the hierarchy are given.

The feature set in Table 4.1 is used to rearrange the hierarchy based on headings in different levels. The strategy we employ works bottom-up in the document tree and first smaller blocks of text (deeper in the hierarchy) are restructured based on the headings. The algorithm to restructure a given block within the document tree is given in Figure 4.6.



Antibiotics and bacterial diseases
HTML code: ...<h1>Antibiotics and bacterial diseases</h1>...
Features: {h1=true}

FREQUENTLY ASKED QUESTIONS
HTML code: ...<b><font size="2">FREQUENTLY ASKED QUESTIONS</font></b>...
Features: {B=true, f_size=2, allUpperCase=true}

Figure 4.5. Headings corresponding to different levels in the document hierarchy

```
Algorithm RestructureTree(p)
Input
    p: parent node of the nodes constituting the block to be restructured
begin
1:  Remove all the children of p to a list L
2:  textAppendPoint = p
3:  headingAppendPoint = p
4:  for each node n in L
5:      if (n is not a heading)
6:          Append n as child to textAppendPoint
7:      else
8:          Check headingFormats list
9:          if (there is no entry for the format of the current node)
10:             Add the new heading format to headingFormats list as the
                next level in the hierarchy
11:         end if
12:         Update headingAppendPoint
13:         Append n as child to headingAppendPoint
14:         Update textAppendPoint
15:     end if
16: end for
end
```

Figure 4.6. Hierarchy restructuring algorithm

According to the algorithm, given a particular node in the tree, its children are considered one by one. Meanwhile, the formatting features of headings and their corresponding levels are stored (*headingFormats* list). If the considered node is not a heading, it is appended under the last heading node encountered (*textAppendPoint*). If the node is a heading, first, it is checked whether it belongs to a heading level previously used in the block. If no entry is found for that format, it is added as a new level. Then, the node is appended to the appropriate position in the hierarchy (*headingAppendPoint*). For example, consider block (3) in Figure 4.3. The heading “Like little…” is identified as belonging to the first level in that block. The sentences that follow are rearranged under that heading. Then, the heading “What antibiotics…” is placed to the same level as the former heading because it has the same features.

Figure 4.7 shows the tree obtained after the application of the restructuring step on the tree of Figure 4.3. In this tree, the root covers the entire document. The intermediate nodes contain section and subsection headings and the leaves contain the underlying sentences. As can be seen, most of the sentences are correctly identified under the corresponding headings. Also, most of the headings are in correct levels. There is, however, an error in the level of the heading “Antibiotics and bacterial diseases” in block (1). It should be at a higher level than the following content. The reason for the error is that (1) is considered as a separate block based on DOM tree processing and thus restructured accordingly.

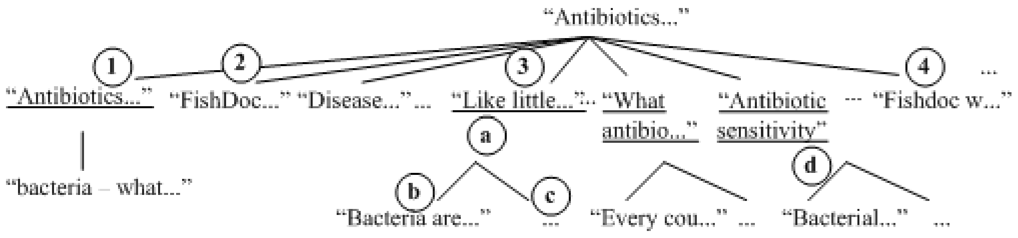


Figure 4.7. Part of the document tree after restructuring

4.2. Evaluation

4.2.1. Performance Measures

The accuracy of the proposed method for structural processing is evaluated according to two different criteria: the accuracy of heading extraction and the accuracy of hierarchy extraction. For this purpose, the outputs of the proposed method are compared with golden standard headings and document hierarchies which are determined manually for each document.

In order to evaluate the accuracy of heading extraction we adapted measures which are widely used in the field of information retrieval [19]. These are precision, recall and f-measure. For each text unit in a document, four different results can be identified by comparing the results of the proposed method with the golden standard: *TP* (true positive), *FP* (false positive), *FN* (false negative), and *TN* (true negative) as in Table 4.2. Based on these values, recall (*R*), precision (*P*), and f-measure (*F*) values for the heading extraction experiment are calculated as in 4.1, 4.2 and 4.3. Here, recall is computed as the ratio of the number of headings correctly identified to the number of actual headings. Precision is calculated as the ratio of the correctly identified headings to the number of headings identified. F-measure is a combined measure of recall and precision.

Table 4.2. Contingency table for the heading extraction experiment

		Golden Standard	
		Heading	**Non-heading**
Proposed Method	**Heading**	*TP*	*FP*
	Non-heading	*FN*	*TN*

$$R \equiv \frac{TP}{TP + FN} \tag{4.1}$$

$$P \equiv \frac{TP}{TP + FP} \tag{4.2}$$

$$F \equiv \frac{2 \times P \times R}{P + R} \tag{4.3}$$

The accuracy of hierarchy extraction should be determined based on the tree structure of the output. We especially focus on parent-child relationships in this tree structure because they correspond to *heading* - *subheading* and *heading* - *text* (i.e. heading and underlying sentence) relationships in the document. We define the accuracy as the ratio of the number of correctly identified parent-child relationships in the obtained hierarchy (as compared with the golden standard) over the total number of parent-child relationships. Formally, given a manually identified hierarchy and an automatically extracted hierarchy for a document *i*, if there exists an edge (i.e. parent-child relationship) between the node pair (*p*,*c*) in both of the hierarchies, we say *e*(*p*,*c*) = 1; otherwise, *e*(*p*,*c*) = 0. Given the set of parent-child node pairs PC_i in a manually identified document hierarchy *i*, the hierarchy accuracy is computed as in 4.4.

$$Hierarchy\_Accuracy(i) \equiv \frac{\sum_{p,c \in PC_i} e(p,c)}{|PC_i|} \tag{4.4}$$

4.2.2. Experiments

The structural processing algorithm was run on two different document collections (*English Collection* and *Turkish Collection*). The examination of the collections revealed that they include different levels of structured documents, ranging from flat documents to highly structured ones with an average sectional hierarchy depth of around four. In Figure 4.8, the distribution of hierarchy depths in both collections are given.

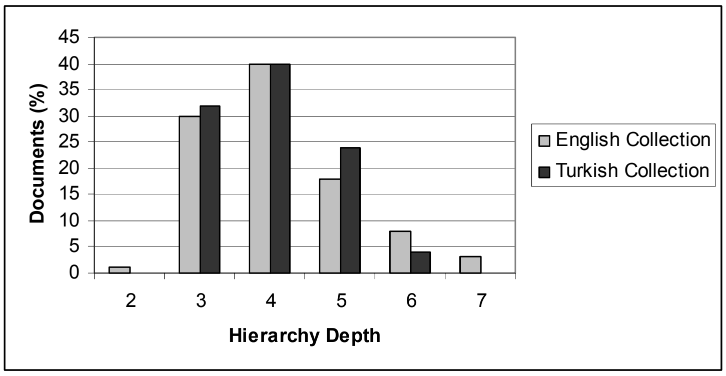


Figure 4.8. Distribution of hierarchy depths in the document collections

In order to evaluate the accuracy of the system, the hierarchy output of the algorithm was compared with the manually identified hierarchy as the golden standard. We also compared the results of the proposed system with a baseline (hierarchy formed using only heading tags <*h1*> through <*h6*> in HTML). In the following subsections, the results for English and Turkish collections are presented.

4.2.2.1. English Collection. The results are given for each document set (corresponding to each query) in English Collection separately followed by their averages for the whole collection. In Table 4.3, the average number of actual headings in the documents and the accuracy for the heading extraction are given for the proposed system and the baseline method. As seen, in the proposed system, a fairly high value was obtained for heading recall (88%). Compared with recall, the precision obtained for heading extraction is lower (64%). The reason is that although most of the headings are correctly identified, some text fragments which are not headings are also extracted as headings. This is due to the cluttered organization encountered in most of the Web documents. The f-measure of the proposed system is calculated as 71%. In the table, the heading recall for the baseline method is also given, which is 43%. The precision and f-measure values for the baseline method could not be computed, because they were undefined in some cases where the baseline method failed to identify any heading correctly in a document.

Table 4.3. Heading extraction results for English Collection

Document Set	**Actual Number**	**Proposed Sys. Recall**	**Proposed Sys. Precision**	**Proposed Sys. F-measure**	**Baseline Recall**
1	6.50	0.94	0.60	0.69	0.51
2	11.30	0.80	0.65	0.67	0.34
3	8.20	0.91	0.56	0.66	0.68
4	3.60	0.89	0.64	0.73	0.38
5	9.30	0.89	0.58	0.66	0.57
6	18.10	0.82	0.70	0.73	0.39
7	5.40	0.84	0.59	0.67	0.27
8	6.90	0.98	0.57	0.68	0.56
9	12.70	0.93	0.76	0.82	0.38
10	6.20	0.84	0.75	0.77	0.24
Average	8.82	0.88	0.64	0.71	0.43

Some statistics related to the hierarchy extraction experiment are given in Table 4.4. These include the average depth of document DOM trees before the processing, the average hierarchy depths obtained in the proposed and baseline methods and the average depths of the actual hierarchies. We observe that, on the average, the depth decreases from 15.21 to 6.54, which signals a significant improvement. As mentioned previously, the DOM tree has a cluttered structure and contains many superfluous levels. The structural processing algorithm eliminates this irrelevant information and leaves the tree with only the details relevant to the actual content of the document. As can be seen, the manually extracted hierarchies have an average depth of 4.11. Thus, the proposed method gives rise to hierarchies with a depth slightly larger than the actual depth. An analysis of individual documents explains this situation: users sometimes express the contents on the same level with different styles of writing, but these are identified incorrectly by the algorithm as belonging to different sections. On the other hand, the depth of the baseline hierarchy is less than that of the actual hierarchy. This is an expected result since the users usually do not use the heading tags for dividing the document into sections; instead this is achieved by changing the style in between the sections. Thus, the scarcity of heading tags in the documents results in smaller depths.

Table 4.4. Statistics related to the hierarchy depths

Document Set	**DOM Tree**	**Proposed Sys. Hierarchy**	**Baseline Hierarchy**	**Actual Hierarchy**
1	15.80	5.50	3.40	3.70
2	20.80	8.20	3.10	4.20
3	12.10	7.30	3.90	4.10
4	13.90	4.90	3.40	3.90
5	13.20	6.10	3.70	4.00
6	13.00	7.00	3.60	4.40
7	19.20	6.20	3.10	3.80
8	12.80	6.10	3.70	4.20
9	17.50	7.10	3.30	4.00
10	13.80	7.00	2.90	4.80
Average	15.21	6.54	3.41	4.11

Table 4.5 shows the accuracy results for hierarchy extraction. The average accuracy obtained in the proposed system is 71%, whereas it is 50% in the baseline method. This result indicates that half of the sectional relationships in the hierarchy extracted from an HTML document using heading tags are wrong. A significant improvement is possible via a heuristics-based analysis of the document structure.

In Figure 4.9, we also show the accuracies of the proposed system and the baseline method with respect to the percentage of documents. The figure indicates that nearly half of the documents have accuracy between 80% and 100% in the proposed system, while only about 25% of the documents achieve this rate in the baseline method. We observe that for majority of the documents, acceptable accuracy rates were obtained in the proposed method.

Table 4.5. Hierarchy accuracy results for English Collection

Document Set	Baseline (only h tags)	Proposed System
1	0.57	0.58
2	0.52	0.81
3	0.64	0.74
4	0.40	0.66
5	0.51	0.66
6	0.40	0.65
7	0.54	0.74
8	0.55	0.69
9	0.48	0.77
10	0.36	0.78
Average	0.50	0.71

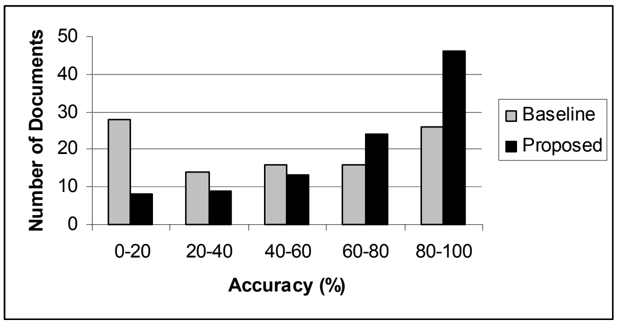


Figure 4.9. Distribution of accuracy results in hierarchy extraction

4.2.2.2. Turkish Collection. We adapted our rule-based approach for structural processing to Turkish Web documents. For this purpose, we modified some of the heuristics used in heading extraction. These include content-related heuristics to recognize cue phrases commonly encountered in Turkish Web documents; e.g. “burayı tıklayın” (“click here”), “favorilere ekle” (“add to favorites”), etc.

In Table 4.6, first, the average numbers of headings in document sets, as found in manual investigation, are given. Then, the performance of the automatic analysis is given in terms of recall, precision and f-measure. The results show 79% recall for heading extraction. Compared with recall, the precision obtained for heading extraction is lower

(57%). In the experiment, we obtained 70% average accuracy for hierarchy extraction (Table 4.7).

Table 4.6. Heading extraction results for Turkish Collection

Document Set	**Number of Headings**	**Recall**	**Precision**	**F-measure**
1	7.60	0.81	0.56	0.64
2	5.40	0.67	0.63	0.61
3	5.10	0.84	0.49	0.66
4	4.90	0.89	0.54	0.68
5	9.20	0.89	0.68	0.73
Average	5.40	0.79	0.57	0.65

Table 4.7. Hierarchy extraction results for Turkish Collection

Document Set	**DOM Tree Depth**	**Hierarchy Depth**	**Hierarchy Accuracy**
1	17.6	6.5	0.49
2	16.2	5.0	0.61
3	20.4	7.5	0.78
4	18.8	5.6	0.80
5	19.2	5.1	0.81
Average	17.2	6.1	0.70

The baseline method failed for Turkish collection because the particular <*h*> tags were not used in any of the documents. Instead, the sectional hierarchy was achieved using other features on the DOM tree such as format tags. In order to test whether the methods work on Turkish documents in general, we performed an additional analysis on documents of a Turkish university Web site (50 documents on *boun.edu.tr* domain) and obtained 71% accuracy using the proposed approach, which proves the robustness of the algorithm for Turkish Web pages.

5. MACHINE LEARNING APPROACH FOR STRUCTURAL PROCESSING

In the structural processing of Web documents, we first developed a rule-based approach for the extraction of headings and sectional hierarchies of documents in which we obtained acceptable results as presented in Chapter 4. However, a rule-based approach has also some disadvantages. It is less adaptive and less robust when compared with a machine learning approach. Although the heuristics employed may be suitable for most of the cases, they cannot model exceptions sufficiently. A machine learning approach can be more flexible; as an example, a classifier can combine different features to make a decision rather than using some predefined rules to filter headings. For this reason, we decided to investigate the structural processing also using machine learning techniques.

The problem we consider is the extraction of all the headings in a document and the underlying hierarchy which is a more complex problem than extracting a single title for a document which was previously investigated in the literature [39]. In the hierarchical analysis, a structure-based learning (i.e. tree-based learning) approach is needed rather than the simpler case of classification. In the literature, structure-based learning approaches have been applied to natural language processing tasks such as syntactic parsing whose output is also a tree structure. In fact, the document sectional hierarchy extraction problem we consider is analogous to the syntactic parsing of sentences. In the former, the input to be parsed is a document consisting of text fragments (sentences or paragraphs), whereas in the latter, the input is a single sentence consisting of words.

The main difficulty in developing a tree-based learning approach is the exponential search space encountered due to the nature of the problem. We considered several approaches to this problem proposed in the literature. In [70], the main approaches in statistical parsing are considered. One approach is nondeterministic parsing with the use of generative probabilistic models and dynamic programming. Such parsing techniques can be improved by using a discriminative model. Another approach is using discriminative models to search the complete space of possible parses. Finally, a different approach is using a greedy algorithm which makes a sequence of locally optimal choices to approximate a globally optimal solution. This methodology has been emerged as an alternative to more complex models especially in dependency parsing. This is also the approach we take in this part of the thesis.

In the proposed system, we aim to model the dependency relations between document fragments analogous to the dependency relations between word/phrase pairs in syntactic parsing. These include *heading - underlying text* and *heading – subheading* relations in the document hierarchy. Also, we use a discriminative machine learning approach rather than a generative approach, because it allows a large number of features as required by the problem we consider. To the best of our knowledge, this is a novel approach in the structural processing of Web documents. In order to develop such a machine learning approach, several issues need to be considered with respect to training and testing phases. These are overviewed in the following:

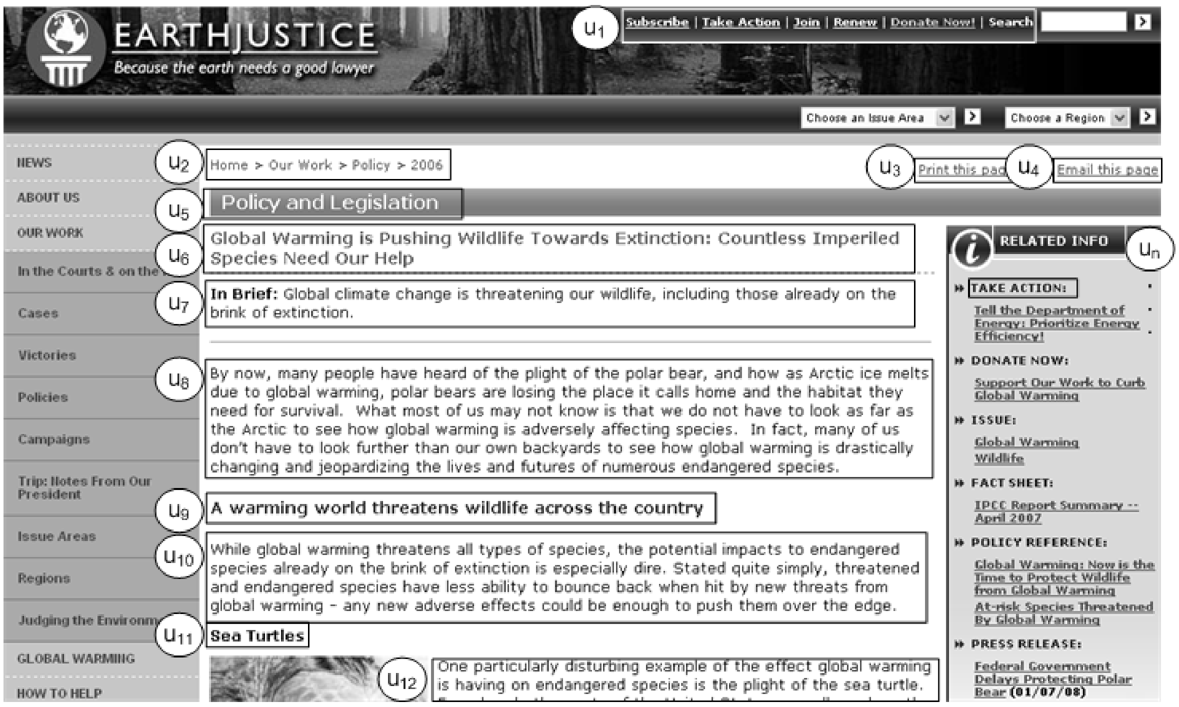
(1) <u>Training:</u> The training phase includes design decisions with respect to the machine learning models, algorithms and types of features:
   (a) <u>Machine learning models:</u> This includes developing a representation for Web documents and models suitable for the learning task. A classification approach can be appropriate for heading extraction task; whereas a structure-based approach is needed for hierarchy extraction.
   (b) <u>Machine learning algorithms:</u> Several algorithms such as support vector machines (SVMs) and perceptron can be adapted for the learning task.
   (c) <u>Types of features:</u> This includes developing appropriate feature representations with respect to the document content and structure as required by heading and hierarchy extraction tasks. Then, the effects of using combinations of different types of features can be evaluated.

(2) <u>Testing:</u> The performance of each method can be evaluated on a sufficiently large and representative test set using cross validation.

The rest of this chapter is organized as follows. We present the proposed machine learning models and feature sets in sections 5.1 and 5.2, respectively. These are followed by the incremental learning approach in Section 5.3 and variations of the testing approach in Section 5.4. Finally, we present the implementation (Section 5.5) and evaluation details (Section 5.6).

5.1. Machine Learning Models

In the proposed system, we model a Web document as a sequence of text units based on the order in the HTML source of the document. This is analogous to sentence parsing where each sentence is modeled as a sequence of words. We define a *text unit* u_i in the document as a text fragment delimited by a newline (i.e. paragraph) as illustrated by rectangles in Figure 5.1. We developed two main machine learning models: heading extraction model and hierarchy extraction model. These are detailed in the following subsections.


EARTHJUSTICE
Because the earth needs a good lawyer
u_1
Subscribe | Take Action | Join | Renew | Donate Now! | Search
Choose an Issue Area
Choose a Region
NEWS
ABOUT US
OUR WORK
In the Courts & on the
Cases
Victories
Policies
Campaigns
Trip: Notes From Our President
Issue Areas
Regions
Judging the Environm
GLOBAL WARMING
HOW TO HELP
u_2
u_3
Print this pag
u_4
Email this page
u_5
u_6
u_7
u_8
u_9
u_{10}
u_{11}
u_{12}
u_n
RELATED INFO
TAKE ACTION:
Tell the Department of Energy: Prioritize Energy Efficiency!
DONATE NOW:
Support Our Work to Curb Global Warming
ISSUE:
Global Warming
Wildlife
FACT SHEET:
IPCC Report Summary -- April 2007
POLICY REFERENCE:
Global Warming: Now is the Time to Protect Wildlife from Global Warming
At-risk Species Threatened By Global Warming
PRESS RELEASE:
Federal Government Delays Protecting Polar Bear (01/07/08)

Home > Our Work > Policy > 2006

Policy and Legislation

Global Warming is Pushing Wildlife Towards Extinction: Countless Imperiled Species Need Our Help

In Brief: Global climate change is threatening our wildlife, including those already on the brink of extinction.

By now, many people have heard of the plight of the polar bear, and how as Arctic ice melts due to global warming, polar bears are losing the place it calls home and the habitat they need for survival. What most of us may not know is that we do not have to look as far as the Arctic to see how global warming is adversely affecting species. In fact, many of us don't have to look further than our own backyards to see how global warming is drastically changing and jeopardizing the lives and futures of numerous endangered species.

A warming world threatens wildlife across the country

While global warming threatens all types of species, the potential impacts to endangered species already on the brink of extinction is especially dire. Stated quite simply, threatened and endangered species have less ability to bounce back when hit by new threats from global warming - any new adverse effects could be enough to push them over the edge.

Sea Turtles

One particularly disturbing example of the effect global warming is having on endangered species is the plight of the sea turtle.

Figure 5.1. Part of an example HTML document

5.1.1. Heading Extraction Model

In the heading extraction model, the Web document is considered as a flat sequence of text units and binary classification is performed. The training examples include (u_i, y_i) pairs for i = 1…t where u_i correspond to a text unit and y_i to its label. The label denotes whether the text unit is a heading or not. In Figure 5.2, the representation of an example document (consisting of n units) in the heading extraction model is given where x_{ij} for j =

1…k correspond to the features of a unit u_i. Then, the task is to learn the classification model in order to distinguish positive instances (headings) from negative instances (non-headings).

u_1:	$x_{11}\ x_{12}\ x_{13}$ …..…………….... x_{1k}	y_1 (heading)
u_2:	$x_{21}\ x_{22}\ x_{23}$ ……………….… x_{2k}	y_2 (heading)
u_3:	$x_{31}\ x_{32}\ x_{33}$ ……………….….x_{3k}	y_3 (non-heading)
…	…	…
…	…	…
…	…	…
$u_{n\text{-}1}$:	$x_{(n\text{-}1)1}\ x_{(n\text{-}1)2}\ x_{(n\text{-}1)3}$ ….…. $x_{(n\text{-}1)k}$	$y_{n\text{-}1}$ (heading)
u_n:	$x_{n1}\ x_{n2}\ x_{n3}$ …..………………. x_{nk}	y_n (non-heading)

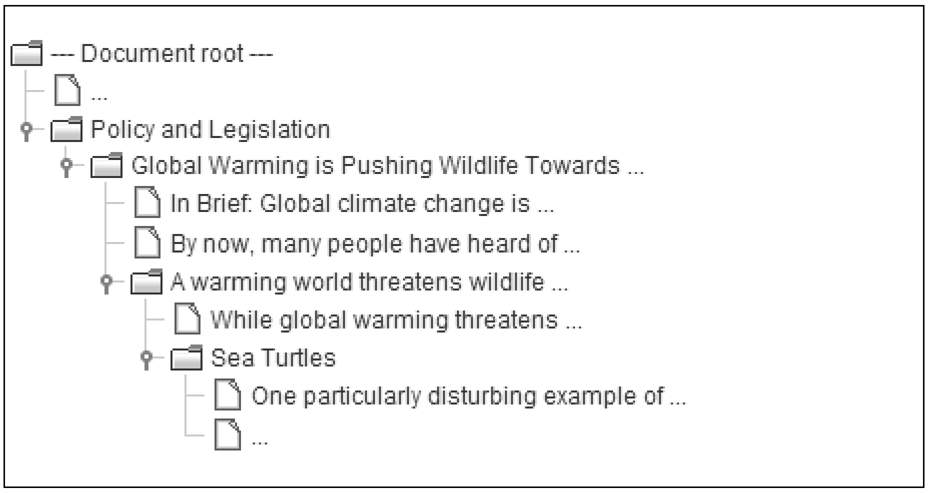
Figure 5.2. Representation of an example document in heading extraction model

5.1.2. Hierarchy Extraction Model

In hierarchy extraction, the general problem of learning a mapping from inputs $x \in X$ to outputs $y \in Y$ is considered. In the case of syntactic parsing, X is a set of sentences and Y is a set of possible parse trees [27]. Analogously, in the structural analysis of Web documents, we define X as a set of documents and Y as a set of possible sectional hierarchies of documents using the following framework:

- Training examples (x_i, y_i) for $i = 1…n$
- A function $GEN(x)$ which enumerates a set of possible outputs for an input x
- A representation Φ mapping each $(x_i, y_i) \in X \times Y$ to a feature vector $\Phi(x_i, y_i)$
- A parameter vector α

In the proposed system, the training set includes (x_i, y_i) pairs where x_i is a Web document and y_i is the golden standard tree corresponding to the document sectional hierarchy. In Figure 5.3, part of the sectional hierarchy for the example document in Figure 5.1 is given.



--- Document root ---
- ...
- Policy and Legislation
  - Global Warming is Pushing Wildlife Towards ...
    - In Brief: Global climate change is ...
    - By now, many people have heard of ...
    - A warming world threatens wildlife ...
      - While global warming threatens ...
      - Sea Turtles
        - One particularly disturbing example of ...
        - ...

Figure 5.3. Part of the sectional hierarchy for an example document

In this framework, the learning task is to estimate the parameter vector *α* using the training examples as evidence. The parameter vector α is estimated such that it will give the highest scores to correct outputs as in 5.1.

$$F(x) = \arg \max_{y \in GEN(x)} \Phi(x, y) \cdot \alpha \tag{5.1}$$

In document sectional hierarchy extraction, we work on trees (i.e. document sectional hierarchies) analogous to parse trees in syntactic parsing. In general, the main difficulty in developing a tree-based learning approach is the exponential search space encountered during the solution of the problem. That is, the set of candidate outputs for an input *x*, enumerated by *GEN*(*x*), can grow exponentially with the size of *x*, making the brute force enumeration of the set members intractable. One solution to this problem is to use a heuristic method, such as beam search, to reduce the search space. Such an approach has previously been successfully applied to other tasks in the literature, such as syntactic parsing and generating a table-of-contents for a general document [27, 28]. In this approach, the output tree is incrementally built by making a sequence of locally optimal choices in order to approximate a globally optimal solution, which is also the approach we take.

5.2. Features

We define the types of features used in the proposed machine learning models according to different levels of a document. The first one includes the features corresponding to the smallest unit (i.e. text unit) in a document. This is followed by the features based on the context of a unit; i.e. the neighboring units. Finally, global features are defined considering the document as a whole.

5.2.1. Unit Features

A *text unit* (delimited with a newline) is the smallest unit in our machine learning models and is associated with a set of features. We determine text units automatically using certain tags which are used to specify paragraphs in HTML, such as <*br*> and <*p*>. A text unit may correspond to one or more nodes in the HTML Document Object Model (DOM) tree because different parts of a text unit may be enclosed in different HTML tags. For example, a paragraph may contain one or more bold words, i.e. enclosed in <*b*> tags, whereas it may not be bold in the remaining parts. As a design decision, the formatting features used at the beginning of a text unit may be used for that unit.

In most of the Web documents, Cascading Style Sheets (CSS) rules are used to define the presentation of document contents. We use Cobra HTML Renderer and Parser which supports parsing of the DOM tree and CSS information [65]. After the parsing process, text units in the HTML document are automatically identified and associated with features. The unit features include formatting features, DOM tree features, content features and other types of features as outlined in the following subsections.

5.2.1.1. Formatting Features. These features are mostly based on HTML tags and attributes used for formatting the text unit, such as font size, boldness, color, etc. (Table 5.1). The features are similar to the formatting features used in the rule-based approach. However, we use the formatting information obtained after CSS information is incorporated. Therefore, we do not define additional features for CSS. The features have Boolean, integer or string values. The letter case information may have three different values: All letters in upper case (e.g. “RELATED WORK”), initial letters in uppercase (e.g. “Related Work”) and other (e.g. “Related work”). Finally, an emphasis score may be defined for a unit based on its formatting differences with a given unit.

Table 5.1. Formatting features of a text unit

Feature	**Description**	**Data Type**
h1	<*h1*>, level-1 heading	Boolean
h2	<*h2*>, level-2 heading	Boolean
h3	<*h3*>, level-3 heading	Boolean
h4	<*h4*>, level-4 heading	Boolean
h5	<*h5*>, level-5 heading	Boolean
h6	<*h6*>, level-6 heading	Boolean
B	<*b*>, bold	Boolean
strong	<*strong*>, strong emphasis	Boolean
em	<*em*>, emphasis	Boolean
A	<*a*>, hyperlink	Boolean
U	<*u*>, underlined	Boolean
I	<*i*>, italic	Boolean
f_size	<*font size=...*>, font size	Integer
f_color	<*font color=...*>, font color	String
f_face	<*font face=...*>, font face	String
b_color	Background color of the text unit	String
li	<*li*>, different levels in a list	Integer
lettercase	Letter case used in the text unit	Integer

5.2.1.2. DOM Tree Features. These features are related to the DOM tree parse of the document. Although the DOM tree is mostly concerned with the presentation of a document, it can also contain valuable information about the structural organization of the document. In most of the HTML documents, the organization is achieved by using nested tables (<*table*>) and divisions (<*div*>).

We can use the *DOM address* of a unit in order to incorporate structural information [40]. For this purpose, starting from <*html*> node (i.e. the root), the children of each node are numbered consecutively starting from 0 (see Figure 5.4). Then, the DOM address of a unit may be determined by following the path from the root node to the leaves covering the unit. For example, the unit with the text content “Introduction” in the figure has the DOM address “0.1.0.0.0.1.0”. The DOM path of a unit (e.g. “*html.body.div.table.tr.td.b*”) can also be used as another feature. An alternative feature is the position of a text unit within the innermost table or division according to the DOM tree. This information may be especially useful in heading extraction because headings are often found at the first position within a table or division.

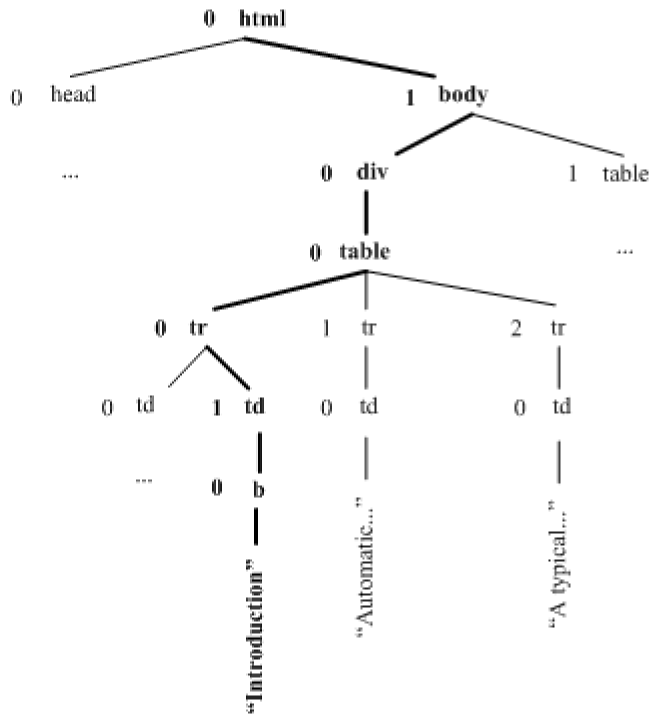


Figure 5.4. Illustration of DOM addresses on an example DOM tree

5.2.1.3. Content Features. These features are related to the textual content of a unit, such as features to specify whether a unit contains certain cue words or phrases (e.g. “back to top”, “login”, etc.). Other content related features include the number of characters in the text unit (e.g. 0-50, 51-100, >100), the number of sentences the text unit contains, and the punctuation mark at the end of the text unit (‘.’, ‘,’, ‘;’, ‘:’, ‘!’, no punctuation mark, etc.). Such features are especially important in the heading extraction task; in general, headings are limited in length, consist of no more than one sentence and contain no punctuation marks at the end. Content features may have binary and integer values.

5.2.1.4. Other Features. Several other features may be defined for a text unit. Some of these are related to the visual position of the unit in a rendered Web document; i.e., as it is displayed in a browser (see Figure 5.5). For this purpose, the *x* and *y* coordinates of the text units are computed using the Cobra HTML Parser and Renderer. This information may be useful both for heading and hierarchy extraction tasks and can be incorporated to the machine learning models as the visual position difference between two given units in the document. Other features include a feature designating whether a unit is the document root or not, and the usage of a horizontal line (<*hr*> tag in HTML) in a portion of the document to separate content.

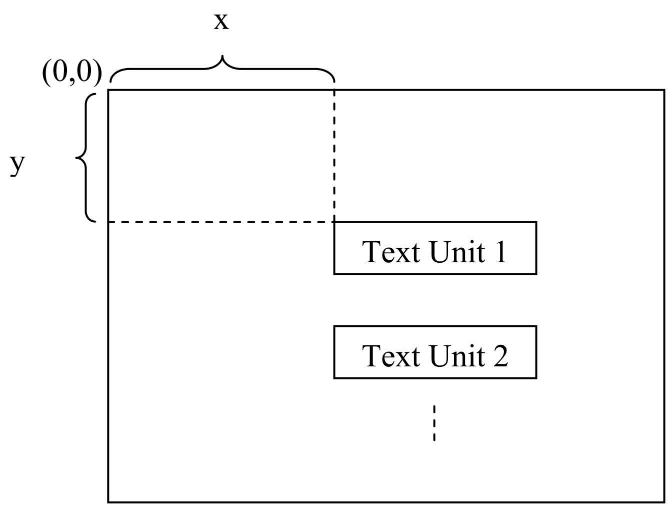


Figure 5.5. Visual coordinates of a text unit in a rendered Web document

5.2.2. Contextual Features

In the proposed machine learning models, the contextual information of a text unit is utilized based on the ordering of units in a document and the document sectional hierarchy. In heading extraction, the context of a unit is investigated based on the preceding and succeeding units in the document. In order to incorporate hierarchical information, the context of a unit in the tree corresponding to the document sectional hierarchy is considered. In Figure 5.6, the potential attachment of a unit u, to the unit u_{10} in a partial tree is given. We use u_{ij} to denote the unit i levels above a unit u, and j units to its left similar to a syntactic parsing study [27]. For example, u_{10} denotes the parent unit of the unit u, u_{20} denotes the grandparent unit, and u_{01} denotes the preceding sibling unit.

We define *composite features* of two units u and u_{ij} as F_{ij} in order to incorporate the contextual information into the machine learning models. For this purpose, we mainly utilize the difference and distance of two units in context because it can provide useful information in determining the headings and the sectional hierarchy. Intuitively, a heading unit (i.e. parent unit) is usually more emphasized than the underlying text unit or subheadings in terms of formatting. Similarly, units under the same heading, i.e. sibling units, generally have similar formatting features.

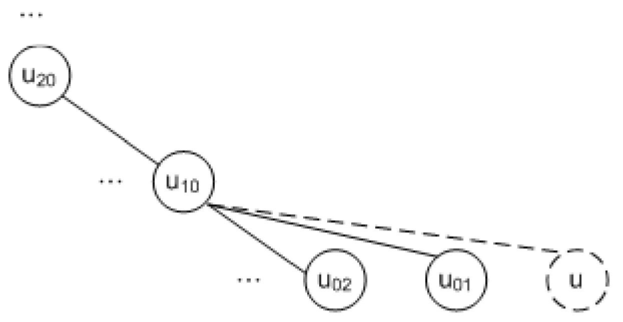


Figure 5.6. Contextual information in document sectional hierarchy

Composite features of two units are defined for formatting, DOM tree and visual features in our system. For binary and integer formatting features, these are defined as the difference of the corresponding values. For example, to determine the composite value of *h1* feature for two units *m* and *n*, we calculate *m.h1-n.h1*. If *h1* value of *m* is 1 and *h1* value of *n* is 0, their difference is 1. Similarly, if *m* has a font size 12 and *n* has font size 14, their difference can be calculated as -2.

In calculating the composite feature for the DOM addresses of two units, we consider the similarity of their addresses starting from the root of the DOM tree. In Figure 5.7, some units of an example document (based on the order of their appearance in the document) are given together with their DOM addresses. The units that are headings are shown as bold. As previously noted in Chapter 4, semantically related parts in a document show spatial locality in the DOM tree. As a result, related parts usually have similar DOM addresses. We define the composite DOM address feature for two units as the length of the path common to their DOM addresses starting from the root. This value can be normalized by dividing it to the depth of the overall DOM tree for that document in order to overcome DOM path length differences across different documents. In Figure 5.7, the length of common paths for given unit pairs from Figure 5.8 are computed. As seen, the DOM address of a section heading is more similar to the DOM address of the following content compared to a unit in another section.

Text Unit:	DOM tree address:
…	…
“Medline”	0.20.1.0.3.1.3.0.3.3.1.1.1.1.3.3.3.1.1.0.0
“Congress Resource Centre”	0.20.1.0.3.1.3.0.3.3.1.1.1.1.3.3.5.1.1.0.0
“EXPLORE :”	**0.20.1.0.3.1.9.1.1.0.2.0.0**
Most Read News	0.20.1.0.3.1.9.1.1.0.2.5.1.0.0
All News	0.20.1.0.3.1.9.1.1.0.2.5.5.1.1.0
…	…
“Osteoporosis”	**0.20.1.1.16.1.0**
“The latest medical news and…”	0.20.1.1.18.1.1.1.0.0
“Medical News and Alerts”	0.20.1.1.18.1.1.6.1.3.0.0.0
…	…

Figure 5.7. DOM addresses of text units in an example document

Example 1:	
“EXPLORE :”	**0.20.1.0.3.1.9.1.1.0.2**.0.0
“Most Read News”	**0.20.1.0.3.1.9.1.1.0.2**.5.1.0.0
	Common path length:11
Example 2:	
“EXPLORE :”	**0.20.1**.0.3.1.9.1.1.0.2.0.0
“Osteoporosis”	**0.20.1**.1.16.1.0
	Common path length: 3

Figure 5.8. Calculation of DOM address similarity

We define composite features for visual positions of two units *m* and *n* by considering the visual *x* and *y* coordinate differences between the units. The difference of *x* coordinates and the difference of *y* coordinates can be considered separately as positive, negative or zero. As an example, if the difference of *y* coordinates is positive, this means that *m* comes below *n* in the visual display of the Web document; therefore, *m* cannot be a heading of *n*. The difference of *x* and *y* coordinates can also be compared to a threshold value. As an example, if the visual *y* position difference between two consecutive units *m* and *n* is very large, *n* cannot be a heading of *m*.

5.2.3. Global Features

In addition to the features of a single unit and the contextual features, we can also define *global features* by considering the document or the document sectional hierarchy as a whole. These features may be incorporated to the process of building the document sectional hierarchy. One such feature we use is the depth of the document sectional hierarchy at a given step in training or testing.

5.3. The Incremental Learning Approach

In document sectional hierarchy extraction, we take an incremental approach based on one or more machine learning models. In Figure 5.9, part of a graph representing the sectional hierarchy of a Web document is given. In the graph, the root node is a dummy node covering the whole document and each of the other nodes corresponds to a text unit in the document. The nodes are arranged from left to right according to their order of appearance in the document. The actual dependency relations between node pairs (i.e., parent-child relationships like heading-underlying text or heading-subheading) are shown as regular lines. These correspond to positive examples for the learning process. The negative examples are the potential dependency relations which are not realized in the golden standard hierarchy (e.g. the dashed lines in Figure 5.9).

The main training algorithm for constructing the document sectional hierarchy is given in Figure 5.10. The input to the algorithm is the training set consisting of Web documents and corresponding golden standard hierarchies. For each document in the training set, the algorithm works on the units one by one starting from the first unit, and considers the attachment of a unit to its parent unit as a positive example and other potential attachments of the unit as negative examples. In this process, two constraints due to the document flow are applied. First, a unit cannot be attached to a heading unit coming after it in the document order. Second, the connections cannot cross each other according to the projectivity rule as in dependency parsing [71]. The projectivity rule states that if a unit u_j depends on the unit u_{j-k}, (i.e., u_{j-k} is the parent of u_j), then all the units between u_{j-k} and u_j must also be descendants of u_{j-k} in the hierarchy. This rule can be implemented as in the following definition (lines 5-11 in Figure 5.10):

Projectivity: "When searching for the parent of a unit u_j, consider only the previous unit (u_{j-1}), the parent of u_{j-1}, that unit's parent, and so on to the root of the tree.

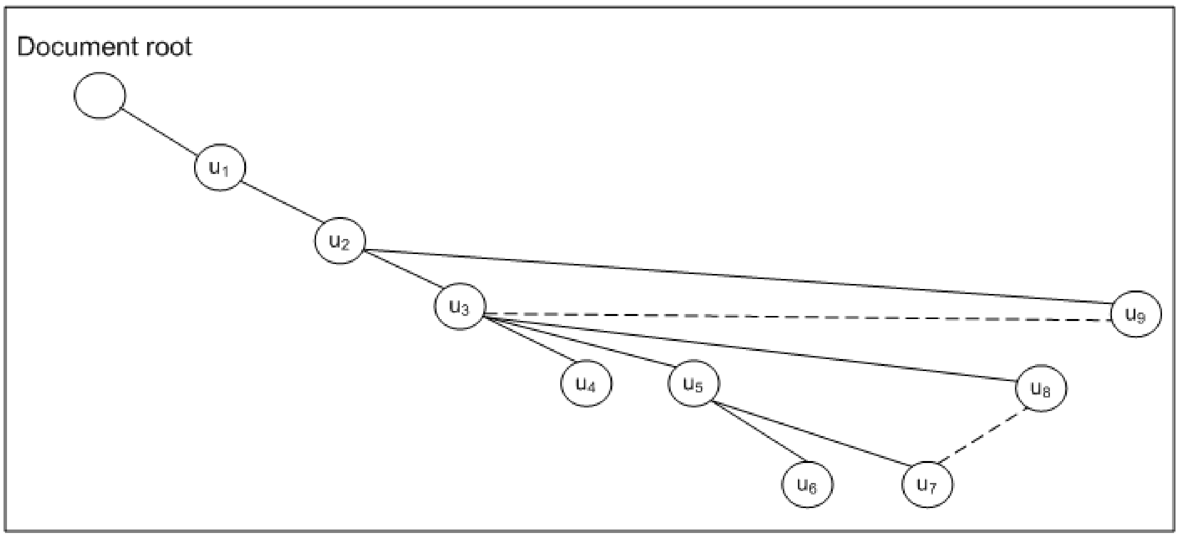


Figure 5.9. Part of an example document graph

```
Algorithm Train_Hierarchy_Extraction_Model
Input
    Training set ($x_i$, $y_i$)
begin
1:  for each document $x_i$ in the training set
2:      for each unit $u_j$ in $x_i$
3:          $p$ = parent($u_j$)
4:          Set ($p$, $u_j$) as positive_example
5:          $prev = u_{j-1}$
6:          while ($prev$ != null)
7:              if ($prev$ != $p$)
8:                  Set ($prev$, $u_j$) as negative_example
9:              end if
10:             $prev$ = parent($prev$)
11:         end while
12:     end for
13: end for
14: Build machine learning model
End
```

Figure 5.10. The training algorithm for document sectional hierarchy extraction

5.4. Variations of the Testing Approach

In document sectional hierarchy extraction, the testing phase includes incrementally building the hierarchy for a previously unseen document. The document is automatically decomposed into text units each of which is associated with features. The testing algorithm operates on each text unit sequentially based on the order in the document; at each step, alternative partial solutions (trees) are generated. We take the approach of maintaining only the most likely partially generated solutions at each step using the heuristic approach of beam search. Beam search algorithm is based on breadth-first search. At each level, all successors of the states in the current level are generated; however, only a predetermined number of states are stored which is called the *beam width*. For this purpose, we maintain a set of partial analyses (i.e. partial trees) of the given document. The set is initially empty and is updated at each step of the algorithm.

We adapt two operations similar to the previous work in syntactic parsing [27, 28]: ADV (i.e. advance) and FILTER. Whenever the next text unit in the document is processed, the ADV operation is applied. The potential attachments of the current unit to the partial trees are considered and the set is updated to include new partial trees. In Figure 5.11, potential attachments of a unit (u_9) to an example partial tree are shown with dashed lines. During this process, restrictions on the search space due to the document flow are also applied (direction of attachments and projection principle). To prevent the exponential growth of the set of partial trees, the FILTER operation is introduced. For this purpose, the score for each partial tree is computed using the machine learning models and the partial trees with lower scores are eliminated. We take the approach of maintaining only the top k (i.e. beam width) highest scored partially generated trees at each step.

In order to improve the accuracy of hierarchy extraction, we also utilize the output of heading extraction. For this purpose, we define a preprocessing step to the incremental approach which is based on the idea that a heading unit is always immediately followed by a child unit in the hierarchy. That is, by definition, a heading in a document has always at least one text unit (heading or non-heading) as the underlying content immediately following it. We use the binary classification model in Section 5.1.1 to determine the heading units in a given document. Based on the output of the heading extraction model, if a unit u_j is a heading, we attach u_{j+1} to u_j. In Figure 5.12, the output of the preprocessing step is shown on the units of an example document. The heading units are shown with bold circles in the figure. Assuming that the heading units are correctly identified, each heading unit is connected with its immediately following unit in the preprocessing.

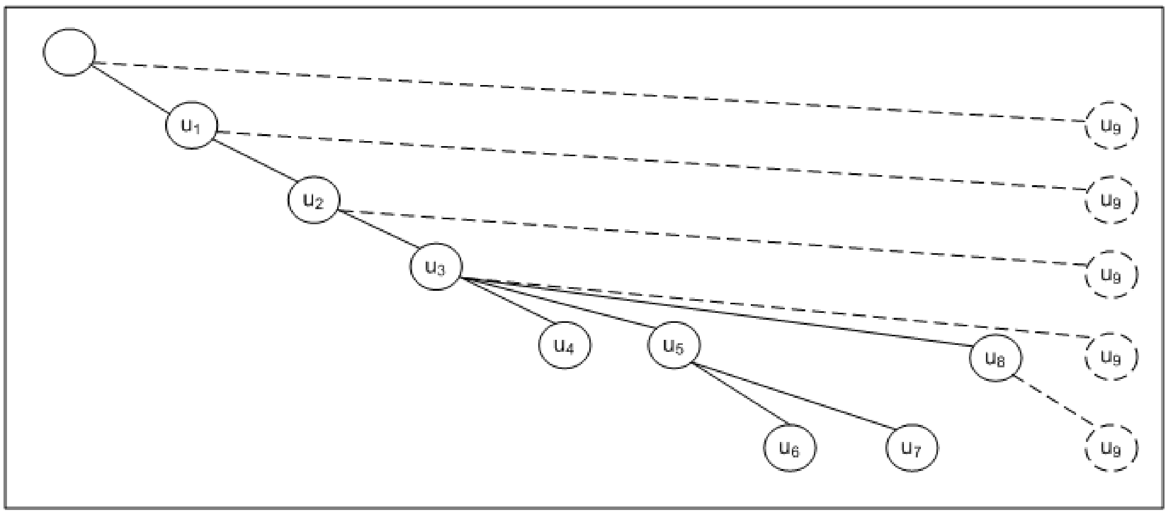


Figure 5.11. Potential attachments of a unit to an example partial tree

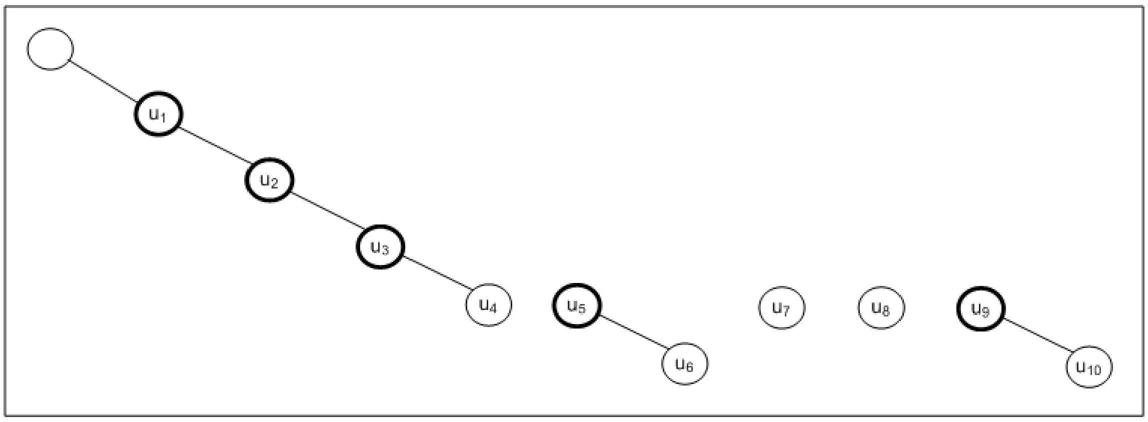


Figure 5.12. Preprocessing for heading units

The main algorithm to build the sectional hierarchies for previously unseen documents (i.e., the test set) is given in Figure 5.13. The algorithm works on each unit of a document sequentially. The first unit of the document is attached to the document root. This corresponds to the initial tree stored in the set of partial trees (*PartialTrees*). The potential attachments of a unit are considered if the unit was not already attached in the preprocessing step (lines 8-14 in Figure 5.13). Another restriction used in the algorithm is to allow a unit to attach only to a heading unit. In this way, the potential attachments of a unit to a non-heading unit are eliminated.

```
Algorithm Test_Hierarchy_Extraction_Model
Input
    Test set ($x_i$, $y_i$)
    $k$: beam width
begin
1:  for each document $x_i$ in the test set
2:      PartialTrees = {}
3:      Attach $u_1$ to document_root, add the tree to PartialTrees
4:      for each unit $u_j$ in $x_i$  ($j$ = 2 to $n$)
5:          if unit $u_j$ not already attached in preprocessing
6:              for each tree T in PartialTrees
7:                  Remove T from PartialTrees
8:                  prev = $u_{j-1}$
9:                  while prev != null
10:                     if prev is a heading
11:                         Attach $u_j$ to prev, add the tree to PartialTrees (ADV)
12:                         prev = parent(prev)
13:                     end if
14:                 end while
15:             end for
16:             Run hierarchy extraction model on all alternative attachments
17:             Keep only top $k$ highest scored trees in PartialTrees (FILTER)
18:         end if
19:     end for
20: end for
end
```

Figure 5.13. The testing algorithm for hierarchy extraction

We developed several modifications to the main testing algorithm to investigate their effect on the accuracy of sectional hierarchy extraction. These are detailed in the following:

- Modification 1: Instead of using the score obtained from the machine learning model for hierarchy extraction directly, it is possible to convert it into a probability value. The conversion of a machine learning model output (e.g. Support Vector Machines) into a probability value has been investigated in previous studies [72, 73]. In this conversion, a sigmoid function can be used (see 5.1). The sigmoid function parameters A and B can be estimated using an iterative method [72]. Alternatively, the sigmoid function can be used with fixed parameters; e.g. A=-2, B=0 [73], which is also the approach we take. Given a sequence of units in a document as u_1 to u_n, we define the sequence of parent-child dependencies as *parent*(1) to *parent*(*n*) where *parent*(*i*) = *j* means that the unit *j* is the parent of unit *i* in document sectional hierarchy. We make a simplifying assumption that the probabilities of such dependencies are mutually independent in a document; i.e., the attachment of unit *i* to unit *j* is independent from the other attachments in the hierarchy. Then, the probability of building a document hierarchy with *n* units can be defined using the multiplication rule of probability for independent events as in 5.2.

$$f(x) = 1 / (1 + exp\,(Ax + B)) \tag{5.1}$$

$$\prod_{i=1}^{n} P(parent(i) = j) \tag{5.2}$$

The application of the algorithms M_0 (the main testing algorithm in Figure 5.13) and M_1 (Modification 1) on an example document for units *i*=5 and *i*=7 are illustrated in Figure 5.14 and Figure 5.15, respectively. The dashed lines in the figures show the alternative partial trees obtained in those steps together with the scores obtained in the alternative methods. In M_0, at each step, the scores output by the machine learning algorithm are directly used. In M_1, the scores output by the algorithm are converted to probabilities. Then, the probabilities at each step are multiplied to form the current score.

- Modification 2: An alternative implementation is to run the testing algorithm in two levels. In the first level the algorithm is applied on only heading units; i.e. only heading units are connected in order to obtain the overall heading hierarchy of the document. Then, in the second level, the algorithm continues with the output of the first level and non-heading units are attached to the correct positions in the hierarchy.

- Modification 3: In this implementation, integer ranks are used instead of using the scores output by the machine learning models directly. The partial trees are given ranks starting from “1” which is given to the best scored tree(s). During the filtering, the times a partial tree has obtained rank “1” are summed to obtain its score. The trees with higher scores are favored.

- Modification 4: The partial trees are given integer ranks similar to Modification 3. Then, the ranks at each step are summed to determine the score of a given partial tree. The trees with smaller scores are favored.

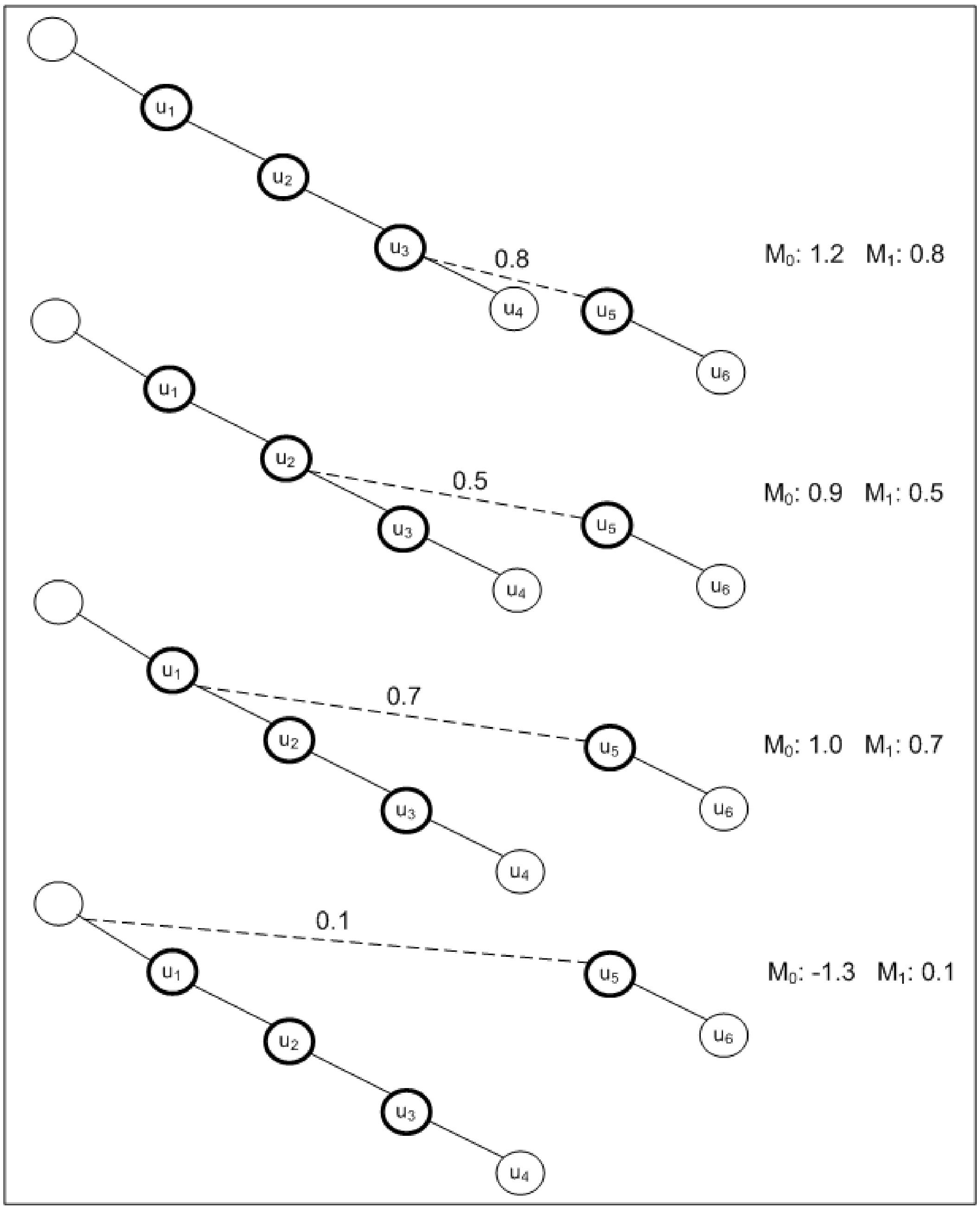
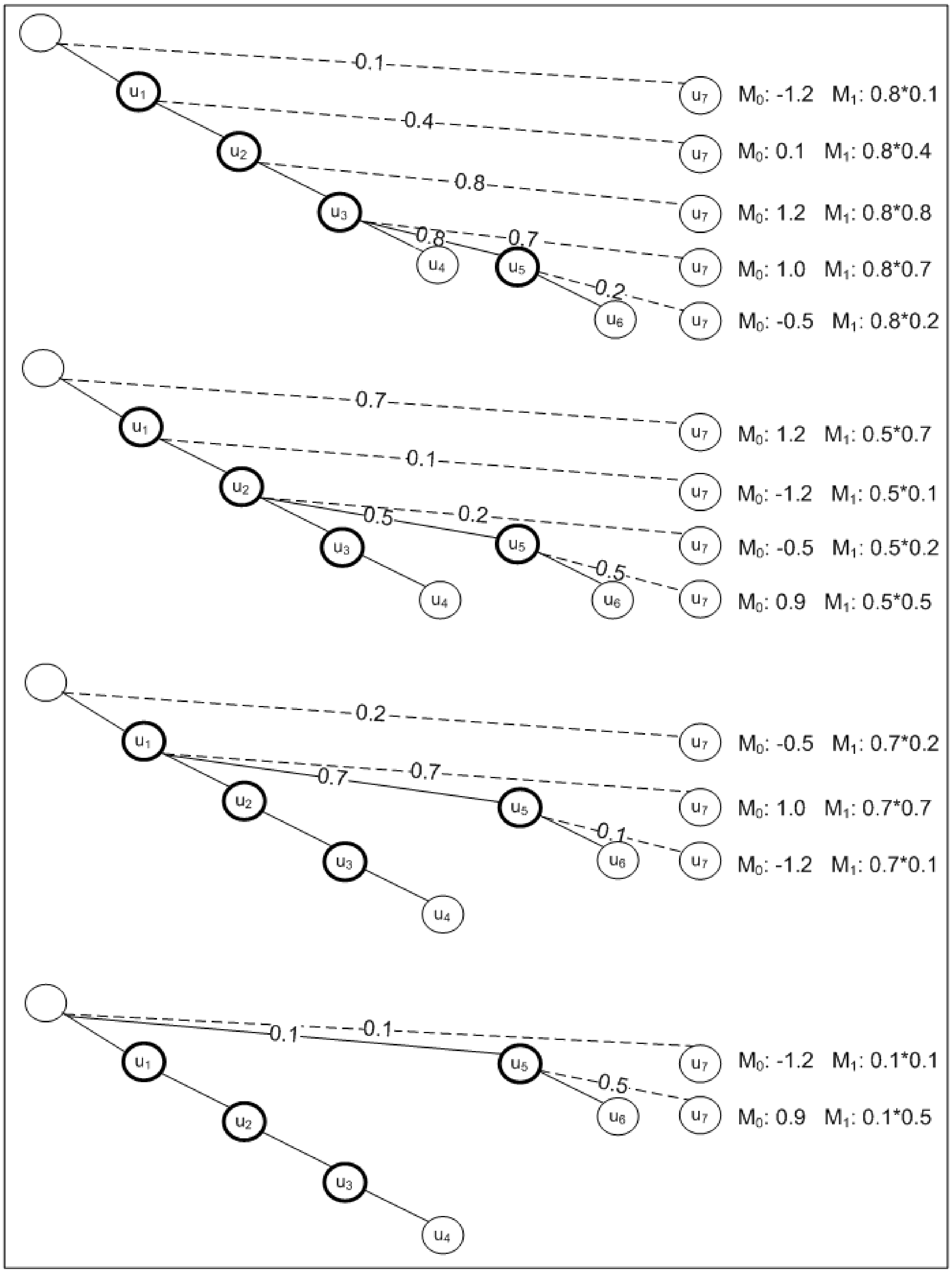


Figure 5.14. Alternative partial trees at *i* = 5

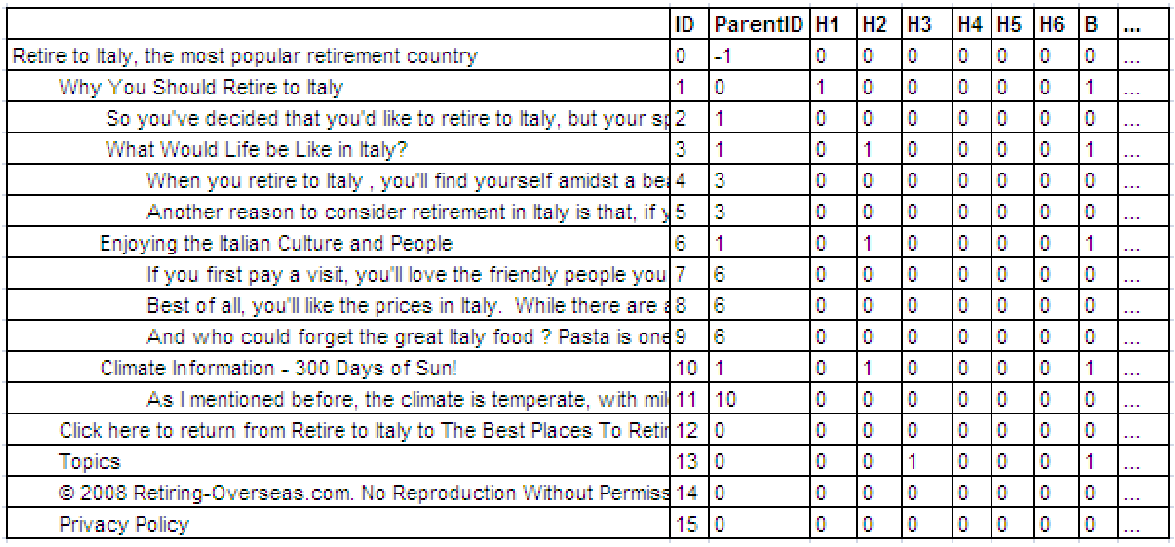


0.1
u_1
u_7
M_0: -1.2 M_1: 0.8*0.1
0.4
u_2
u_7
M_0: 0.1 M_1: 0.8*0.4
0.8
u_3
u_7
M_0: 1.2 M_1: 0.8*0.8
0.8
0.7
u_4
u_5
u_7
M_0: 1.0 M_1: 0.8*0.7
0.2
u_6
u_7
M_0: -0.5 M_1: 0.8*0.2
0.7
u_1
u_7
M_0: 1.2 M_1: 0.5*0.7
0.1
u_2
u_7
M_0: -1.2 M_1: 0.5*0.1
0.5
0.2
u_3
u_5
u_7
M_0: -0.5 M_1: 0.5*0.2
0.5
u_4
u_6
u_7
M_0: 0.9 M_1: 0.5*0.5
0.2
u_1
u_7
M_0: -0.5 M_1: 0.7*0.2
0.7
0.7
u_2
u_5
u_7
M_0: 1.0 M_1: 0.7*0.7
0.1
u_3
u_6
u_7
M_0: -1.2 M_1: 0.7*0.1
u_4
0.1
0.1
u_1
u_5
u_7
M_0: -1.2 M_1: 0.1*0.1
0.5
u_2
u_6
u_7
M_0: 0.9 M_1: 0.1*0.5
u_3
u_4

Figure 5.15. Alternative partial trees at i = 7

5.5. Implementation

The proposed machine learning approaches have been implemented as a standalone application in Java. The program runs on each document in the collection and automatically identifies the text units and their features by utilizing Cobra HTML Parser and Renderer. The units of each document are stored automatically in MS Excel files separately for each document to allow manual annotation of document sectional hierarchies (see Figure 5.16).



	ID	ParentID	H1	H2	H3	H4	H5	H6	B	...
Retire to Italy, the most popular retirement country	0	-1	0	0	0	0	0	0	0	...
Why You Should Retire to Italy	1	0	1	0	0	0	0	0	1	...
So you've decided that you'd like to retire to Italy, but your s	2	1	0	0	0	0	0	0	0	...
What Would Life be Like in Italy?	3	1	0	1	0	0	0	0	1	...
When you retire to Italy , you'll find yourself amidst a be	4	3	0	0	0	0	0	0	0	...
Another reason to consider retirement in Italy is that, if y	5	3	0	0	0	0	0	0	0	...
Enjoying the Italian Culture and People	6	1	0	1	0	0	0	0	1	...
If you first pay a visit, you'll love the friendly people you	7	6	0	0	0	0	0	0	0	...
Best of all, you'll like the prices in Italy. While there are a	8	6	0	0	0	0	0	0	0	...
And who could forget the great Italy food ? Pasta is one	9	6	0	0	0	0	0	0	0	...
Climate Information - 300 Days of Sun!	10	1	0	1	0	0	0	0	1	...
As I mentioned before, the climate is temperate, with mil	11	10	0	0	0	0	0	0	0	...
Click here to return from Retire to Italy to The Best Places To Reti	12	0	0	0	0	0	0	0	0	...
Topics	13	0	0	0	1	0	0	0	1	...
© 2008 Retiring-Overseas.com. No Reproduction Without Permiss	14	0	0	0	0	0	0	0	0	...
Privacy Policy	15	0	0	0	0	0	0	0	0	...

Figure 5.16. The internal representation of a document in implementation

Each unit in a document is associated with a unique identification number (ID) based on its order of appearance in the document. The document root contains the main title of the document as enclosed in <*title*> tags in HTML, and its ID is 0. For each unit, the ID of its parent unit (ParentID) is also stored to keep track of the hierarchical structure (i.e. tree). In golden standard hierarchies, the parent IDs are manually annotated based on the original document. In the testing process, the parent IDs are determined automatically based on the machine learning models. Figure 5.16 also shows a part of the features associated with each unit and the corresponding values. The training and testing data for heading and hierarchy extraction models are obtained by utilizing this hierarchical information and features.

The proposed heading and hierarchy extraction models were implemented using two different machine learning algorithms: support vector machines and perceptron. In the following, the algorithms and their usage are overviewed.

5.5.1. Support Vector Machines-Based Approach

Support vector machines (SVMs) are machine learning methods commonly used in classification. In this approach, given two classes C_1 and C_2 and a training set $X = \{x^t, r^t\}$ where r^t is +1 if $x^t \in C_1$ and r^t is -1 if $x^t \in C_2$, the aim is to find $\boldsymbol{w}$ and w_0 such that the conditions in 5.3 and 5.4 are satisfied [74]. These two conditions can be rewritten as in 5.5. The aim is to separate the instances in two classes using a hyperplane and to have them with some distance away from each other. The distance from the hyperplane to the closest instances on either side of it is called the *margin*. The *optimal separating hyperplane* is defined as the hyperplane maximizing the margin.

$$\boldsymbol{w}^T \boldsymbol{x}^t + w_0 \geq +1 \text{ for } r^t = +1 \tag{5.3}$$

$$\boldsymbol{w}^T \boldsymbol{x}^t + w_0 \leq -1 \text{ for } r^t = -1 \tag{5.4}$$

$$r^t\,(\boldsymbol{w}^T \boldsymbol{x}^t + w_0) \geq +1 \tag{5.5}$$

The distance of $\boldsymbol{x}^t$ to the hyperplane can be written as in 5.6. To maximize the margin, ||w|| is minimized by solving the optimization problem in 5.7. This problem can be rewritten in dual form using Lagrange multipliers α^t as in 5.8. The set of $\boldsymbol{x}^t$ whose $\alpha^t > 0$ are *support vectors*. The obtained discriminant function is called the *support vector machine* (SVM). During testing, $g(x) = \boldsymbol{w}^T \boldsymbol{x} + w_0$ is calculated. If $g(x) > 0$, C_1 is chosen; otherwise, C_2 is chosen.

$$r^t\,(\boldsymbol{w}^T \boldsymbol{x}^t + w_0) \,/\, ||\boldsymbol{w}|| \tag{5.6}$$

$$\text{Minimize } \frac{1}{2}\,||\boldsymbol{w}||^2 \text{ subject to } r^t\,(\boldsymbol{w}^T \boldsymbol{x}^t + w_0) \geq +1,\ \forall t \tag{5.7}$$

$$\text{Maximize } -\frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s\,(\boldsymbol{x}^t)^T\boldsymbol{x}^s + \sum_t \alpha^t \text{ subject to } \sum_t \alpha^t r^t = 0 \text{ and } \alpha^t \geq 0,\ \forall t \tag{5.8}$$

If the training data are not linearly separable, a solution with the least error can be calculated (*soft margin hyperplane*). In the case of nonlinear problems, it is also possible to map the problem to a new space by a nonlinear transformation using appropriate functions (*kernel functions*) and then use a linear model in this new space. Commonly used kernel functions include polynomials of degree d (see 5.9) and radial basis functions where $\boldsymbol{\sigma}$ defines the radius (see 5.10).

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = (\boldsymbol{x}^T \boldsymbol{x}^t + 1)^d \tag{5.9}$$

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp(-\ ||\boldsymbol{x}^t - \boldsymbol{x}||^2 / \sigma^2) \tag{5.10}$$

In the implementation, we utilized SVM-light support vector machine implementation [75]. It supports classification, regression, preference ranking, and several kernel functions including linear, polynomial and radial basis functions. In heading extraction, SVM classification is performed. In hierarchy extraction, positive and negative examples for the learning process are created based on the incremental approach.

5.5.2. Perceptron-Based Approach

Perceptron is a type of artificial neural network. It performs classification by mapping an input x to an output y, basically as a weighted sum as in 5.11 [74]. The weights w_j are called connection weights associated with each dimension of the input (x_j for j = 1…d), and w_0 is used to make the model more general as a bias value. In this way, the perceptron defines a hyperplane to divide the input space into two: positive and negative examples. In training the perceptron, generally *online learning* is used where the training instances are given one by one and the parameters are updated after each instance. Initially, it starts with random initial weights and adapts itself slowly.

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \tag{5.11}$$

In general, the update rule of perceptron has the form in 5.12 [74]. At any step, if the actual output is equal to the desired output, no update is performed. If they are different, the magnitude of the update is proportional to the difference between the desired output and the actual output. The magnitude of the update also depends on the *learning factor*. If the learning factor is large, the updates depend much on recent instances; i.e., as if the system has short memory.

$$\text{Update} = \text{LearningFactor} \cdot (\text{DesiredOutput} - \text{ActualOutput}) \cdot \text{Input} \qquad (5.12)$$

In heading extraction, we implemented a perceptron-based classifier. For hierarchy extraction, we developed an incremental approach based on perceptron. In Figure 5.17, a variant of the perceptron-based training algorithm used in the hierarchy extraction task is given. In the algorithm, the weight vector α is updated if the highest scored tree is not the golden standard parse at the end of the processing of a document (lines 9-12 in Figure 5.17). The learning may be performed over several iterations. The average of the weight vectors obtained during the iterations may be used in the testing phase.

```
Algorithm Train_Perceptron_Based_Hierarchy_Extraction_Model
Input
    Training set ($x_i$, $y_i$)
begin
1:  Initialize weight vector $\boldsymbol{\alpha}$ to 0
2:  for each learning iteration $t$
3:      for each document $x_i$ in the training set
4:          PartialTrees = {}
5:          Attach $u_1$ to document_root, add the tree to PartialTrees
6:          for each unit $u_j$ in $x_i$  ($j$ = 2 to $n$)
7:              FILTER(ADV(PartialTrees))
8:          end for
9:          Calculate $z_i = \arg\max_{z \in GEN(x_i)} \Phi(x_i, z) \cdot \boldsymbol{\alpha}$
10:         if ($z_i$ != $y_i$)
11:             $\boldsymbol{\alpha} = \boldsymbol{\alpha} + LearningFactor \cdot (\Phi(x_i, y_i) - \Phi(x_i, z_i))$
12:         end if
13:     end for
14: end for
end
```

Figure 5.17. The perceptron-based training algorithm

Other variants of the perceptron-based algorithm may also be defined. As a variant, the update may be performed earlier (*early update* as in [27]), if the set of partial trees no longer contains the golden standard partial tree during the processing of a document. Then, the set may be updated to include only the golden standard partial parse. As another variant, the weight vector may be updated after the processing of each document unit rather than at the end of the processing of a document.

5.6. Evaluation

5.6.1. Corpus

For the experiments, we used the Extended English Collection presented in Chapter 3 which contains a total of 500 documents. We used a specification for manually annotating the headings in the documents (Figure 5.18). Document sectional hierarchies were manually marked based on the identified headings and the document organization. The agreement between two different annotators was measured as 70%.

1. Number

- In addition to the main document title (enclosed in <*title*> tags), an HTML document may have zero or more section headings.

2. Form

- A section heading consists of one line and is separated from the surrounding text with one or more line breaks.
- Section headings are more emphasized than the surrounding text in terms of formatting (e.g. font family, font weight, font color, font style, alignment, and background color).

3. Content

- Section headings cannot be too long.
- Section headings mostly do not end with punctuation marks. Sometimes they end with a punctuation mark such as ":".

4. Other

- Text contents in images are not considered.

Figure 5.18. The specification for manual annotation

Some statistics for the document collection are given in Table 5.2. The average sizes of documents are given as the number of text units (i.e. paragraphs separated by line breaks) in the documents. As seen, on the average, a Web document contains about 110 text units. The average depth of document sectional hierarchies and the average number of headings in the documents were determined based on manual marking. The statistics show that the Web documents have an average sectional hierarchy depth of about four and they contain around 10 headings on the average.

Table 5.2. Statistics for the document collection

Number of documents	500
Avg. number of text units	110.7
Avg. hierarchy depth	4.1
Avg. number of headings	10.6

5.6.2. Experiments

In the experiments, the output of heading and hierarchy extraction models were compared against golden standard headings and hierarchies in order to determine the performance of the proposed system. The performance measures defined in Chapter 4 were used which include recall, precision and f-measure for heading extraction, and accuracy for hierarchy extraction. The experiments were also performed for our previously developed rule-based approach. For this purpose, the sentence-based approach developed in Chapter 4 was converted to a paragraph-based implementation. Different from Chapter 4, the document sectional hierarchies were created based on all the units in a document. In the previous approach, the secondary parts such as menus were not considered in this process.

All the experiments were performed using cross-validation. Generally, in *K-fold cross-validation*, the dataset is divided randomly into *K* parts with equal sizes [74]. Then, one of the *K* parts is kept as the validation set and the remaining *K*-1 parts are used as the training set. This process is repeated *K* times; each time for a different one of the *K* parts. One advantage of this method is that all observations are used for training and validation, and each observation is used for validation exactly once. In the experiments, we performed 5-fold cross-validation. We divided the test collection randomly into *K*=5 different parts with the restriction that all the parts contain an equal proportion of documents for each query in the set. The tests were performed on five rounds and the averages of the results were taken.

5.6.2.1. Heading Extraction. We used the binary classification model described in Section 5.1.1 in order to classify text units as heading or not. In this model, the document is considered as a flat sequence of text units. The intuitive idea is that, heading and non-heading units can be determined based on their context. For this purpose, we define features considering the units that immediately follow and precede the current unit. We used five different feature sets (see Table 5.3). These include different combinations of features; i.e., features of the current unit (F_n), composite features of the current unit with immediately following two units ($F_{n(n+1)}$, $F_{n(n+2)}$) and immediately preceding two units ($F_{n(n-1)}$, $F_{n(n-2)}$). The number of features in each case is also shown in the table.

Table 5.3. Feature sets used in heading extraction

Feature Set	**Features**	**Number of Features**
Φ_1	F_n, $F_{n(n+1)}$	58
Φ_2	F_n, $F_{n(n+1)}$, $F_{n(n-1)}$	86
Φ_3	F_n, $F_{n(n+1)}$, $F_{n(n+2)}$	82
Φ_4	F_n, $F_{n(n+1)}$, $F_{n(n+2)}$, $F_{n(n-1)}$	110
Φ_5	F_n, $F_{n(n+1)}$, $F_{n(n+2)}$, $F_{n(n-1)}$, $F_{n(n-2)}$	134

We evaluated the performance of heading extraction using support vector machines and perceptron with different feature sets. In the support vector machine case, we also experimented with different cost factors. The cost factor adjusts the cost of training errors on positive examples (false positives) versus the cost of errors on negative examples (false negatives). In this way, it is possible to achieve different levels of precision and recall. The cost factor becomes especially important when there is an imbalance in the number of positive and negative examples in a training set. In heading extraction task, the number of negative examples (non-heading text units) is much larger than positive examples (i.e. headings) resulting in relatively low recall rates. To overcome this problem, a cost factor greater than 1 may be used. In the experiments, we obtained accurate results when a cost factor of 2 was used. We also experimented with different kernel types: linear, polynomial and radial basis function (RBF). In polynomial kernel, we used power 2 (i.e., d=2) because it provides accurate results.

Table 5.4 shows the precision, recall and f-measure obtained for heading extraction. The best results for SVM were obtained for polynomial and RBF kernels. The most accurate results were obtained for the feature set Φ_5 where all the mentioned features were included. The effect of different feature sets is less obvious for linear SVM. In the case of perceptron, the increased use of contextual information in features generally improved f-measure rates. In Table 5.5, the most accurate results obtained in SVM and perceptron algorithms are given together with the results of the rule-based approach. The results show that machine learning approaches provide dramatic increase in the accuracy of heading extraction compared to the rule-based approach. The improvements become especially noticeable with the use of a nonlinear technique (i.e. polynomial kernel) in SVM.

Table 5.4. Performance results of machine learning methods in heading extraction

Method	**Feature Set**	**Recall**	**Precision**	**F-measure**
SVM – Linear	Φ_1	**0.85**	**0.78**	**0.81**
	Φ_2	0.83	0.78	0.80
	Φ_3	0.81	0.77	0.79
	Φ_4	0.83	0.78	0.80
	Φ_5	0.83	0.78	0.80
SVM – Polynomial	Φ_1	0.87	0.80	0.83
	Φ_2	0.85	0.80	0.82
	Φ_3	0.87	0.82	0.84
	Φ_4	0.85	0.80	0.82
	Φ_5	**0.87**	**0.84**	**0.85**
SVM – RBF	Φ_1	0.84	0.76	0.80
	Φ_2	0.84	0.79	0.81
	Φ_3	0.87	0.81	0.84
	Φ_4	**0.88**	**0.83**	**0.85**
	Φ_5	**0.87**	**0.83**	**0.85**
Perceptron	Φ_1	0.71	0.77	0.74
	Φ_2	0.70	0.78	0.74
	Φ_3	0.71	0.84	0.77
	Φ_4	**0.78**	**0.82**	**0.80**
	Φ_5	0.77	0.81	0.79

In the literature, machine learning techniques (such as SVM) have been applied to the extraction of the main title (i.e. a single heading) from HTML documents where a maximum f-measure of 0.80 was obtained [39]. Compared with that study, the investigated problem, i.e. extracting all the headings in a given HTML document, is a more general and challenging problem where we obtained an f-measure of 0.85.

Table 5.5. Overview of performance results in heading extraction

Method	**Recall**	**Precision**	**F-measure**
SVM	0.87	0.84	0.85
Perceptron	0.78	0.82	0.80
Rule-based Approach	0.72	0.64	0.68

5.6.2.2. Hierarchy Extraction. The second model we evaluated is the tree-based learning approach for sectional hierarchy extraction. In hierarchy extraction, contextual features are defined based on the document sectional hierarchy. Whenever a new unit is added to the partial hierarchy in the incremental learning approach, the features of the units in its context are considered. We experimented with different feature sets including the composite features of the current unit with its candidate parent unit (F_{10}), candidate sibling units (F_{01}, F_{20}) and candidate grandparent unit (F_{02}) as given in Table 5.6. Based on the initial experimental results, we decided to use only features related to the distance and difference between two text units (i.e. relative features such as the font size difference) and eliminated the features with absolute values, such as the font size of the current unit.

Table 5.6: Feature sets used in hierarchy extraction

Feature Set	**Features**	**Number of Features**
Φ_1	F_{10}	17
Φ_2	F_{10}, F_{01}	40
Φ_3	F_{10}, F_{01}, F_{20}	57
Φ_4	$F_{10}, F_{01}, F_{20}, F_{02}$	73

We investigated the effect of using heading information in the task of hierarchy extraction. The basic idea is that a heading is always the parent for the immediately following unit. Based on the output of the heading extraction model, each heading is connected with its immediately following unit (preprocessing step). The initial experiments showed that the use of such information improves the accuracy. Therefore, we utilized heading information in the following experiments. In all the experiments, SVM with polynomial kernel was used with all the features for heading extraction because this was the case best accuracies were obtained (i.e., 0.85 for f-measure).

The accuracy of hierarchy extraction is given in Table 5.7 for the proposed approaches based on SVM and perceptron with different combinations of features and a beam width of 100. The results show that the SVM-based approach performs better than the perceptron-based one. The use of increased number of features usually resulted in improved accuracy in SVM. The best results (67%) were obtained when all the features, i.e. the composite features of the current unit with parent unit, sibling units and grandparent unit, were used together. However, this was not the case in perceptron which is based on a linear approach. SVM performed the best results for polynomial and RBF kernels rather than the linear kernel. We also experimented with different beam widths. In Table 5.8, the accuracies for SVM polynomial and RBF kernels are given for different beam widths ranging from 1 to 100. As seen in the table, there is not much change in the accuracies when the beam widths are varied.

Table 5.7. Performances of SVM and perceptron for hierarchy extraction

Learning Algorithm	**Feature Set**			
	Φ_1	**Φ_2**	**Φ_3**	**Φ_4**
SVM – Linear	0.42	0.61	0.61	0.61
SVM – Polynomial	0.57	0.63	0.63	0.65
SVM – RBF	0.58	0.66	0.67	0.67
Perceptron	0.51	0.46	0.46	0.46

Table 5.8. Effects of different beam widths in hierarchy extraction

Learning Algorithm	**Beam width**				
	1	**10**	**20**	**50**	**100**
SVM – Polynomial	0.64	0.65	0.65	0.65	0.65
SVM – RBF	0.66	0.66	0.66	0.66	0.67

In Table 5.9, some experiment results for the main decoding algorithm (M_0) and the variations (M_1 to M_4) detailed in Section 5.4 are given. We evaluated the methods using the feature set Φ_4 and two different kernels. The tests were performed using a beam width of 100. The best results were obtained for the variation M_4 with polynomial kernel.

We compared the effects of using headings extracted with the machine learning model and manually extracted (golden standard) headings. The tests were performed using the feature set Φ_4 and a beam width of 100 for perceptron and SVM. The best accuracy for hierarchy extraction was obtained when manually extracted headings were used (0.82). When using the headings output by the heading extraction model (Model 1), which corresponds to a fully automatic approach, the highest accuracy obtained was 0.68. In this case, SVM-based approach is superior to the rule-based approach. The performance difference between the fully automatic approach and the approach using manually identified headings stems from imperfect recall and precision rates in the automatic extraction of headings.

Table 5.9. Results for alternative methods in hierarchy extraction

Learning Algorithm	**Method**				
	M_0	**M_1**	**M_2**	**M_3**	**M_4**
SVM – Polynomial	0.65	0.67	0.59	0.64	0.68
SVM – RBF	0.67	0.67	0.59	0.67	0.66

Table 5.10. Overview of performance results in hierarchy extraction

Method	**Model 1 headings**	**Manual headings**
Rule-based Approach	0.61	0.81
Perceptron	0.51	0.82
SVM	0.68	0.79

We conducted an error analysis on the results obtained using the machine learning models and identified the main causes of inaccuracies. One source of inaccuracy in hierarchy extraction is errors in heading extraction. Although we have obtained a high performance in heading extraction (85% f-measure), the remaining inaccuracies (15%) result in a 10-15% decrease of accuracy in hierarchy extraction. False negatives of headings result in loss of structure, whereas false positives result in creation of structure which does not actually exist. Another source of inaccuracy in hierarchy extraction is related to the heuristic-based incremental approach. In order to prevent the exponential growth of partial trees, at each step, only a certain number of partial trees are kept. In some cases, correct partial trees are eliminated later in the processing which results in some errors. Part of the imperfect results in sectional hierarchy extraction result from connecting text units in unrelated document parts. There are cluttered Web documents with complex table-based layouts. Also, we have not eliminated secondary parts such as menus from documents and performed hierarchy extraction on the document as a whole. As a result, some text content and headings may be connected to wrong heading levels. Finally, there are sources of inaccuracies over which we do not have any control. These include errors made by Web document authors; e.g., ambiguous or wrong usage of tags and styles. Nevertheless, we obtained acceptable results as a full automatic approach for sectional hierarchy extraction which is a much more challenging task than identifying only the headings in a document.

6. SUMMARY EXTRACTION

In this chapter, the proposed summarization method, i.e. structure-preserving and query-biased summarization, is detailed. Then, task-based evaluations on different document collections and the discussion are presented.

6.1. The Method

In this thesis, we focus on summarization in the context of Web search. Our main aims are: (1) To enable users to make more accurate relevance judgments as compared to search engine extracts; i.e., by preventing them missing relevant results or spending time with irrelevant items; (2) To have reasonable times for relevance judgment using the summaries as compared to judging the relevance of the original document.

We developed a novel summarization approach for Web search combining two main ideas distinguishing it from the traditional approaches. First, it is based on the idea that utilizing the explicit document structure can help the users more accurately judge the relevance of documents on the Web, where documents have high diversity in structure and complex internal organizations. Second, it is a query-biased method suitable to Web search.

In the proposed system, indicative summaries are created using the method of sentence extraction. The summarization algorithm is run after the structural processing phase described in previous chapters. The structural information is utilized in two different ways in the system. First, it is used in determining the importance of sentences and document sections. Second, the structure, i.e. the context of extracted sentences, is also preserved in the output summaries.

The summaries of the proposed system are created using two levels of scoring. In the first level, individual sentences of a document are scored. In the second level, the sections and subsections are scored in order to determine important ones. Then, based on these two types of scores, the final summaries are formed. The scoring methods are explored in the following subsections.

6.1.1. Sentence Scoring

The text content of the Web document is split into sentences and a variant of the basic sentence scoring metrics in the literature is employed. This includes four different types of sentence scoring methods: Heading, location, term frequency and query methods. We adapted these methods such that they utilize the output of the structural processing step.

- Heading method: The intuition behind this approach is that headings in a document usually contain key words related to the content of the document. Therefore, the sentences containing such words may be important. This approach has been investigated in various studies [22, 44, 56, 59]. However, most of the previous studies either use only the main title of the document, or they utilize the headings without investigating the automatic extraction of them which poses a challenge for Web documents. This study targets all the headings in a Web document based on the output of the structural processing step. Words in the headings are stored in a heading word list after stop words are eliminated and stemming is applied. A heading score is assigned to each sentence as the number of heading words it contains.

- Location method: This method is based on the idea that sentences located at certain positions of the document usually convey salient (i.e., important) information [22, 44, 56, 59]; e.g., sentences occurring near the start or end of the document or its paragraphs. We modified this approach to incorporate sectional information. Sentences are given a positive score if they are the first sentence of a section or subsection as found by the structural processing step.

- Term frequency method: The motivation for this method comes from the idea that terms occurring frequently within a document usually convey important information and sentences with higher number of such words are important sentences [44, 56, 59]. In the proposed system, each sentence is given a term frequency score by summing the frequencies of the constituting words. In finding the term frequencies, stop words are eliminated and stemming is applied.

• Query method: In information retrieval context, biasing summaries towards queries becomes important [22, 56]. By heavily selecting sentences containing query words, it is expected that the users can judge the relevance of the search results better. Therefore, in the proposed system, each sentence is given a query score as the number of query words it contains after stemming is applied.

The scores obtained in each of these methods are normalized by dividing them to the maximum score obtained for that method in a given document; in this way, the normalized scores are in the interval [0, 1]. The overall sentence score is calculated as the weighted sum of these four types of scores as in 6.1 where *s*'s represent method scores and *w*'s correspond to method weights. Table 6.1 shows the score calculation for the following example sentence.

$$s_{sentence} = s_{heading} \times w_{\text{heading}} + s_{location} \times w_{\text{location}} + s_{tf} \times w_{\text{tf}} + s_{query} \times w_{\text{query}} \tag{6.1}$$

Example:

Query: antibiotics bacteria disease

Sentence: "These are the bacteria that are usually involved with bacterial disease such as ulcers, fin rot, acute septicaemia and bacterial gill disease."

In Table 6.1, the application of the four methods on the example sentence is shown together with the applicable terms and the scores. In the example, heading, location and term frequency methods are given equal weights whereas query method is given three times more weight; i.e., $w_{heading} = w_{location} = w_{tf} = 1$ and $w_{query} = 3$. The stems *bacteria*, *bacteri*, and *diseas* appear in the headings throughout the document. Each occurrence of these words in the sentence increases its heading score by one. The location score is either 1 or 0, depending on whether the sentence is the first sentence of any section or subsection. In the example, it is 0. The term frequency score is calculated by summing the frequency of each term in the sentence multiplied by its frequency in the whole document (given with parentheses in the table). Finally, the query score applies to two query terms occurring in the sentence (*bacteria*, *diseas*). In the table, the maximum score in the whole document for each method is also indicated. These values are used in the normalization of the scores. The sentence score is obtained as the weighted sum of the normalized scores.

Table 6.1 Application of scoring methods on an example sentence

Method	**Applicable terms**	**Sentence score**	**Maximum score in document**	**Normalized sentence score**
Heading	bacteria, bacteri, diseas	5	6	0.83
Location	—	0	1	0.00
Term Frequency	bacteria(17), involv(1), bacteri(11), diseas(7), ulcer(5), fin(1), rot(1), acut(1), septicaemia(1), gill(1)	64	261	0.25
Query	bacteria, diseas	2	2	1.00
Overall				4.08

6.1.2. Section Scoring

Traditional summarization approaches usually create summaries by considering the document as a linear sequence of sentences. Some of them score sentences by considering some structural information; e.g. heading and location information. However, during the extraction phase, most of the approaches still select the sentences from an unstructured space of sentences, and also form unstructured summary outputs. In the proposed approach, we consider that different sections and subsections of a document may have different importance values and should be represented at different extents in the summary depending also on the user query. For instance, a section whose content is closely related to the information need of the user may be represented with much more sentences in the final summary than a section with a similar length but less relevant material. Moreover, in the proposed system, structured summaries, which include context of sentences in the form of headings and subheadings, are created rather than flat text summaries.

In the system, each section and subsection of a document is given a section score as a measure of its importance. Sections with higher scores are represented with more sentences in the output summary. The *section score* is calculated as the sum of scores of sentences in a given section. Also, in a hierarchical way, the score of a section can be calculated as the sum of the scores of constituting subsections. Each section or subsection is assigned a *quota* based on its section score. The quota determines the number of sentences with which that section will be represented in the output summary. The quota for the whole document is selected as 25 which is the approximate number of sentences in the output summary. Then, hierarchically, this quota is divided among the sections and subsections as in 6.2 where *s*'s represent section and subsection scores. The summarization algorithm runs on the document tree (sectional hierarchy) obtained in structural processing. A variant of the algorithm is given in Figure 6.1.

$$quota_{subsection} \equiv quota_{section} \times \frac{s_{subsection}}{s_{section}} \tag{6.2}$$

The process starts at the root node of the tree which covers the entire document. The root is given a quota which also corresponds to the maximum summary size as the number of sentences (e.g. 25). The document tree nodes are visited in a breadth-first fashion. If the quota of a node is greater than a predetermined threshold (e.g. 3) and the node has non-leaf nodes (i.e. it contains other subsections rather than only sentences), its quota is shared among the subsections based on the scores of the subsections. When the quota of a section or subsection reaches a certain threshold or the section has no more subsections, the highest scored sentences are selected from that section one by one to be included in the summary together with the heading of that section. Also the predecessor headings in the hierarchy, all the way to the main heading, are selected as a part of the summary if not already included. The summarization continues until the summary quota for the whole document is reached. In Figure 6.2, the summarization process is illustrated on an example document tree containing heading nodes (h) and non-heading nodes (t). The initial quota is 25. One is reserved for the root (containing the document title) and the remaining quota (i.e. 24) is divided among child nodes. This process continues in a hierarchical way. Some of the nodes selected as part of the summary are shown as bold. As seen, whenever a node is selected, its predecessor heading nodes are also incorporated to the summary.

```
Algorithm Summarize
Input
    root: root node of the document tree
begin
1:  Set a threshold for section quotas
2:  Insert root into queue with a quota
3:  while (queue not empty)
4:      summarize = false
5:      Get the next node x from queue
6:      sentenceList = {}
7:      if ($quota_x > threshold$)
8:          sentenceList = {x} if x != root
9:          for each child node c of x
10:             if (c is leaf)
11:                 summarize = true
12:                 insert c into sentenceList
13:             else
14:                 $quota_c = quota_x * sectionScore_c / sectionScore_x$
15:                 Insert c into queue
16:             end if
17:         end for
18:     else
19:         summarize = true
20:         Insert all the sentences under x into sentenceList
21:     end if
22:     if(summarize)
23:         Get quota for sentenceList
24:         while (quota and summary size limit not exceeded)
25:             Mark next highest scored sentence s from sentenceList
                for inclusion in summary
26:             while (summary size limit not exceeded)
27:                 Mark ancestors of s (i.e. headings) in hierarchy
                    for inclusion in summary
28:             end while
29:         end while
30:     end if
31: end while
end
```

Figure 6.1. A variant of the summarization algorithm

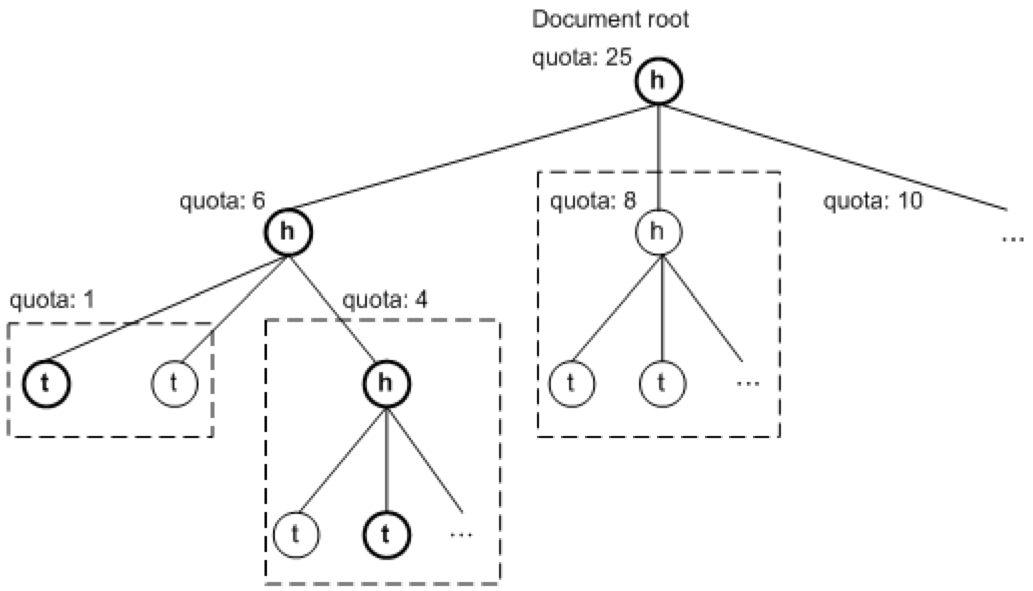


Figure 6.2. Illustration of the summarization process

6.2. Evaluation

6.2.1. Experiment Setup

We conducted a task-based (extrinsic) evaluation for the summaries of the proposed system. In this type of evaluation, the quality of a summary is evaluated based on how it effects the completion of another task. Here, the task is information retrieval where the summaries are planned to be actually used. That is, the summaries are judged according to their usefulness in a search engine. We preferred extrinsic evaluation since it is more suitable for cases where the summarizer is embedded within another system (e.g. a search engine) [42], which is the case in this research.

We used the TREC queries (topics) and documents which were also used in heading and hierarchy extraction experiments. Four types of summaries (document surrogates) were used for comparison. These are:

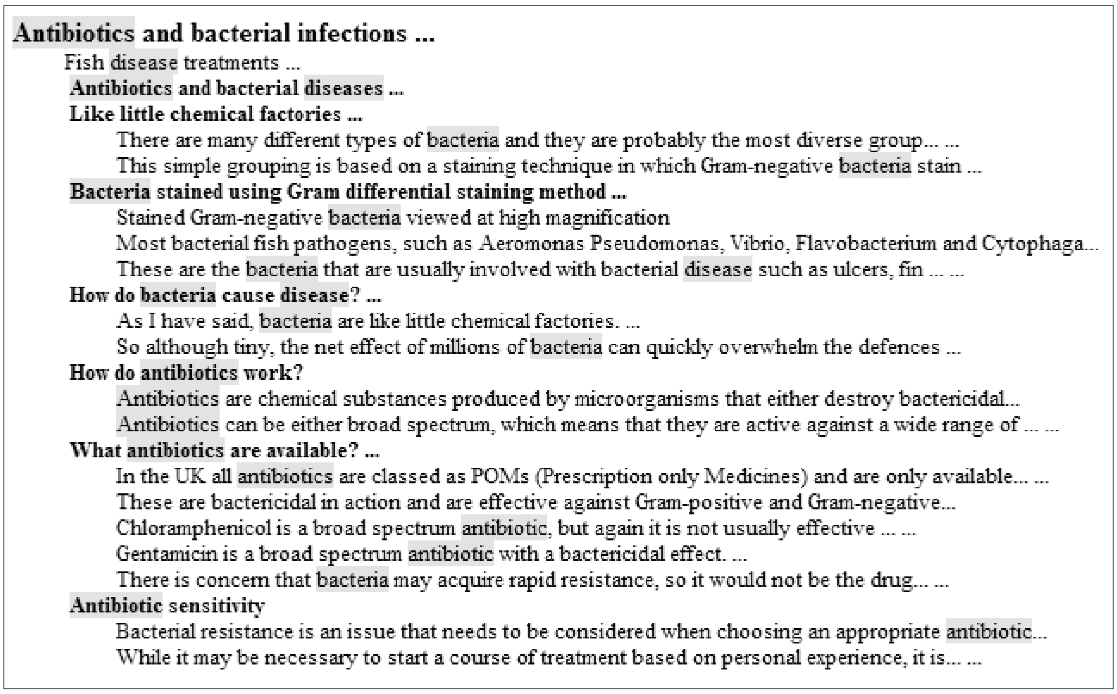
- *Google* – Query-biased extracts provided by Google for the given document.
- *Unstructured* – Query-biased summary without the use of structural information (including extraction of menu information).

- *Structured1* – Structure-preserving and query-biased summary created by the proposed system using the output of the structural processing step.
- *Structured2* – Structure-preserving and query-biased summary created by the proposed system using manually identified structure.

All the summaries except Google extracts are longer summaries with about the same size (about 25 sentences) to make them comparable with each other. When calculating the sentence scores for the summaries, we have experimented with different weight values for Equation 6.1. We observed that giving similar weights to heading, location, and term frequency metrics and assigning a weight to the query metric a few times more than the others give rise to the best performances. This indicates that query terms provide the most important information, but the other sources of information also have an effect and should not be disregarded. The results shown in this section were obtained using the weighting factors $w_{heading} = w_{location} = w_{tf} = 1$ and $w_{query} = 3$.

In creating the unstructured summaries, structural information, i.e. heading and location methods, is not used. Instead, those summaries are formed using only term frequency and query methods. Structured summaries are based on the document sectional hierarchy. Navigation menus in the Web documents (e.g. links) which cause a cluttered view were identified using heuristics and eliminated in structured summaries. An example summary output of the proposed system and an example unstructured summary for the query *Antibiotics Bacteria Disease* are given in Figure 6.3 and Figure 6.4, respectively. In the summaries, query keywords are highlighted. Each summary sentence is output as a single line; for this purpose, the end parts of longer sentences are replaced by '…' (allowing a maximum of about 100 characters for each sentence). The structured summaries are displayed in a hierarchical way in accordance with the sectional hierarchy obtained in the structural processing step. Also, headings and subheadings are given as bold.

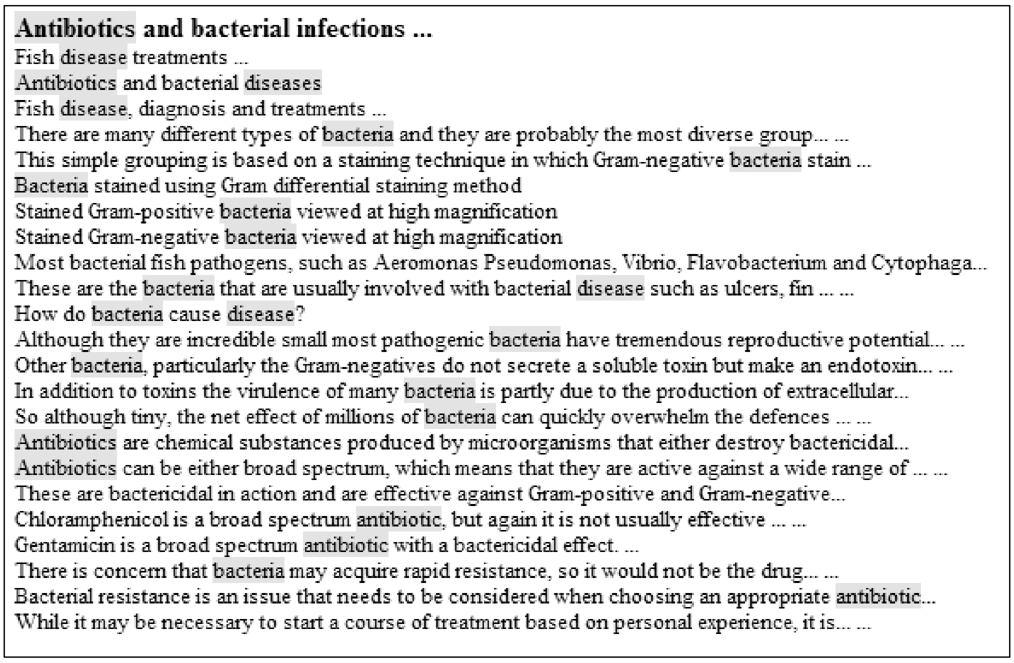
The experiments were performed on human subjects. We considered several experimental methodologies proposed in the literature. Generally, in experiments performed on humans, the differences in the responses of different people to the same treatment may be very large due to individual differences, such as differences in experience, training or background. As a result, it may be difficult to detect real differences between treatments or systems. In order to control this variability between different subjects, a *repeated measures* (i.e. *within-subjects*) design may be used [76]. In this experimental design, each of the *n* treatments is used in each person (i.e. subject). In this way, the effects of differences among subjects are minimized. Therefore, we decided to use a repeated measures design. In the proposed system, each of the summary types for a document (corresponding to different system treatments; e.g. *Google*, *Unstructured*, *Structured1*, etc.) are presented to the user. Also, the summary type and documents are presented in a random order to reduce carryover effects and the original full-text document is not displayed until all the summaries for that document are displayed. In the experiments, each query and corresponding summary sets were evaluated by more than one subject (between 4 and 10 subjects).



Antibiotics and bacterial infections ...
Fish disease treatments ...
Antibiotics and bacterial diseases ...
Like little chemical factories ...
There are many different types of bacteria and they are probably the most diverse group... ...
This simple grouping is based on a staining technique in which Gram-negative bacteria stain ...
Bacteria stained using Gram differential staining method ...
Stained Gram-negative bacteria viewed at high magnification
Most bacterial fish pathogens, such as Aeromonas Pseudomonas, Vibrio, Flavobacterium and Cytophaga...
These are the bacteria that are usually involved with bacterial disease such as ulcers, fin ... ...
How do bacteria cause disease? ...
As I have said, bacteria are like little chemical factories. ...
So although tiny, the net effect of millions of bacteria can quickly overwhelm the defences ...
How do antibiotics work?
Antibiotics are chemical substances produced by microorganisms that either destroy bactericidal...
Antibiotics can be either broad spectrum, which means that they are active against a wide range of ... ...
What antibiotics are available? ...
In the UK all antibiotics are classed as POMs (Prescription only Medicines) and are only available... ...
These are bactericidal in action and are effective against Gram-positive and Gram-negative...
Chloramphenicol is a broad spectrum antibiotic, but again it is not usually effective ... ...
Gentamicin is a broad spectrum antibiotic with a bactericidal effect. ...
There is concern that bacteria may acquire rapid resistance, so it would not be the drug... ...
Antibiotic sensitivity
Bacterial resistance is an issue that needs to be considered when choosing an appropriate antibiotic...
While it may be necessary to start a course of treatment based on personal experience, it is... ...

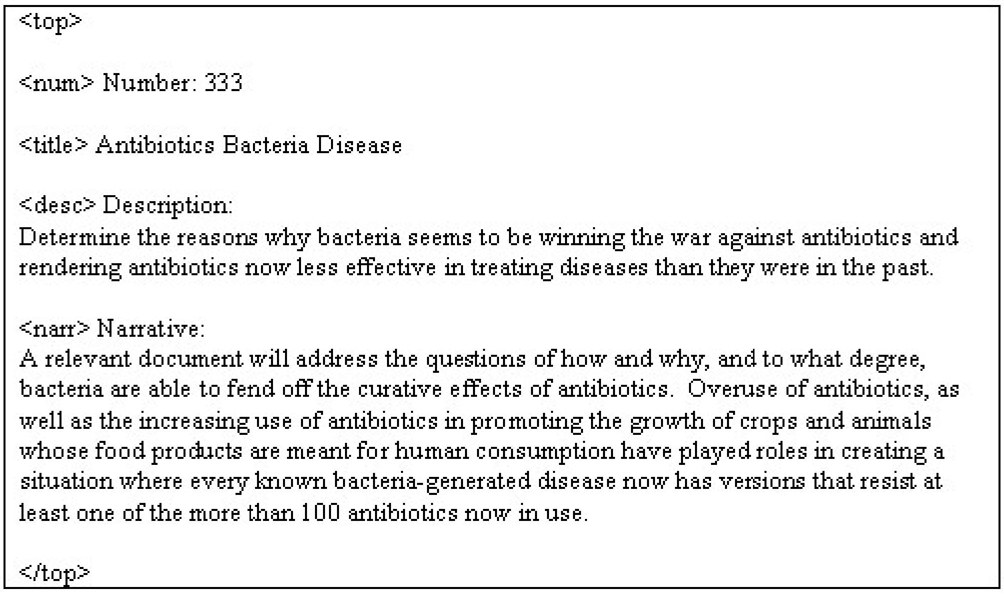
Figure 6.3. An example summary output of the proposed system

In the experiments, the subjects were given the necessary instructions and the queries one by one, and were asked to determine the relevance of the documents with respect to the given query (either relevant or not) based on the summaries. An example query is given in Figure 6.5. As seen in the figure, each query is presented with a title (query terms), a description, and a narrative part. The description states what is intended by the query terms and the narrative part provides a guide for deciding on the relevancy of a document. The subjects performed the experiment tasks on a web-based interface which is overviewed in the next section.



Antibiotics and bacterial infections ...
Fish disease treatments ...
Antibiotics and bacterial diseases
Fish disease, diagnosis and treatments ...
There are many different types of bacteria and they are probably the most diverse group... ...
This simple grouping is based on a staining technique in which Gram-negative bacteria stain ...
Bacteria stained using Gram differential staining method
Stained Gram-positive bacteria viewed at high magnification
Stained Gram-negative bacteria viewed at high magnification
Most bacterial fish pathogens, such as Aeromonas Pseudomonas, Vibrio, Flavobacterium and Cytophaga...
These are the bacteria that are usually involved with bacterial disease such as ulcers, fin ... ...
How do bacteria cause disease?
Although they are incredible small most pathogenic bacteria have tremendous reproductive potential... ...
Other bacteria, particularly the Gram-negatives do not secrete a soluble toxin but make an endotoxin... ...
In addition to toxins the virulence of many bacteria is partly due to the production of extracellular...
So although tiny, the net effect of millions of bacteria can quickly overwhelm the defences ... ...
Antibiotics are chemical substances produced by microorganisms that either destroy bactericidal...
Antibiotics can be either broad spectrum, which means that they are active against a wide range of ... ...
These are bactericidal in action and are effective against Gram-positive and Gram-negative...
Chloramphenicol is a broad spectrum antibiotic, but again it is not usually effective ... ...
Gentamicin is a broad spectrum antibiotic with a bactericidal effect. ...
There is concern that bacteria may acquire rapid resistance, so it would not be the drug... ...
Bacterial resistance is an issue that needs to be considered when choosing an appropriate antibiotic...
While it may be necessary to start a course of treatment based on personal experience, it is... ...

Figure 6.4. An example unstructured summary

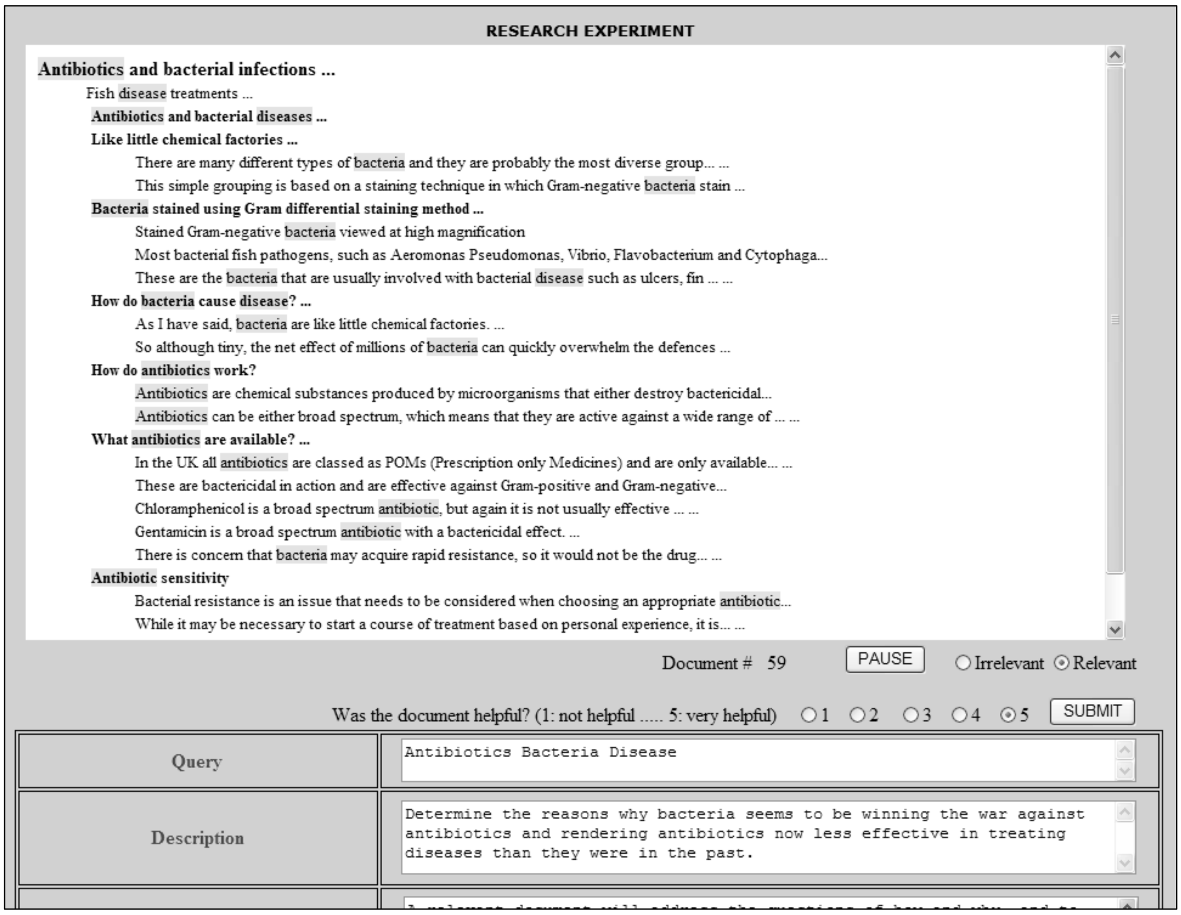


```
<top>

<num> Number: 333

<title> Antibiotics Bacteria Disease

<desc> Description:
Determine the reasons why bacteria seems to be winning the war against antibiotics and
rendering antibiotics now less effective in treating diseases than they were in the past.

<narr> Narrative:
A relevant document will address the questions of how and why, and to what degree,
bacteria are able to fend off the curative effects of antibiotics. Overuse of antibiotics, as
well as the increasing use of antibiotics in promoting the growth of crops and animals
whose food products are meant for human consumption have played roles in creating a
situation where every known bacteria-generated disease now has versions that resist at
least one of the more than 100 antibiotics now in use.

</top>
```

Figure 6.5. An example TREC query used in the experiments

6.2.2. Experiment Interface

The Web-based interface for the task-based evaluation has been developed using ASP.NET framework and SQL Server 2005 database. The users enter the system using predefined user ID and passwords. Then, the instructions for the experiment and the details of the current query are displayed. After the instructions, the user is presented with the actual evaluation screen. In Figure 6.6, a screenshot of the interface is given. It contains a frame in which the summaries are displayed one by one in a predetermined random order. The summaries (including Google ones) are presented in a consistent formatting style in order to prevent the effect of recognition of one particular system. The user has to evaluate a given summary/document as either relevant or irrelevant. The decision times of the users are also recorded automatically. During the experiment, the details of the current query are always presented at the bottom of the page.


RESEARCH EXPERIMENT

Antibiotics and bacterial infections ...
Fish disease treatments ...
Antibiotics and bacterial diseases ...
Like little chemical factories ...
There are many different types of bacteria and they are probably the most diverse group... ...
This simple grouping is based on a staining technique in which Gram-negative bacteria stain ...
Bacteria stained using Gram differential staining method ...
Stained Gram-negative bacteria viewed at high magnification
Most bacterial fish pathogens, such as Aeromonas Pseudomonas, Vibrio, Flavobacterium and Cytophaga...
These are the bacteria that are usually involved with bacterial disease such as ulcers, fin ... ...
How do bacteria cause disease? ...
As I have said, bacteria are like little chemical factories. ...
So although tiny, the net effect of millions of bacteria can quickly overwhelm the defences ...
How do antibiotics work?
Antibiotics are chemical substances produced by microorganisms that either destroy bactericidal...
Antibiotics can be either broad spectrum, which means that they are active against a wide range of ... ...
What antibiotics are available? ...
In the UK all antibiotics are classed as POMs (Prescription only Medicines) and are only available... ...
These are bactericidal in action and are effective against Gram-positive and Gram-negative...
Chloramphenicol is a broad spectrum antibiotic, but again it is not usually effective ... ...
Gentamicin is a broad spectrum antibiotic with a bactericidal effect. ...
There is concern that bacteria may acquire rapid resistance, so it would not be the drug... ...
Antibiotic sensitivity
Bacterial resistance is an issue that needs to be considered when choosing an appropriate antibiotic...
While it may be necessary to start a course of treatment based on personal experience, it is... ...

Document # 59 PAUSE Irrelevant [✓ Relevant]

Was the document helpful? (1: not helpful ..... 5: very helpful) 1 2 3 4 [✓ 5] SUBMIT

Query	Antibiotics Bacteria Disease
Description	Determine the reasons why bacteria seems to be winning the war against antibiotics and rendering antibiotics now less effective in treating diseases than they were in the past.

Figure 6.6. A screenshot of the Web-based experiment interface

In the recent experiments, we also added a user poll to rate the helpfulness of each summary. For this purpose, a Likert scale was used with ratings between 1 and 5 (i.e., 1 as not helpful and 5 as very helpful). Such questionnaires have also been used in previous task-based evaluation studies for summarization systems [22, 56].

6.2.3. Performance Measures

In the task-based evaluation, our aim is to determine how well the proposed system summaries help the users judge the relevancy of documents. In most previous summarization studies, the user judgments for the summaries are compared with golden standard judgments for documents (determined by annotators). That is, each document is strictly marked as relevant or irrelevant beforehand. In *relevance prediction*, which is a more recent approach, the subject's judgment on a summary is compared with his or her own judgment on the original full-text document. This is an intuitive approach and parallels also with what a user does in a real-world task using a search engine; i.e., the user judges the relevancy of the original document based on the summary and decides whether or not to open the document in the browser. If the user judges that the document is relevant, he or she opens it in the browser and investigates whether their decision was correct.

The relevance prediction approach has been found to be more reliable than the gold standard approach [77]. As the performance measure, we also used relevance prediction instead of relying on a golden standard. In this way, we expect to reduce the effect of differences in human subjects and obtain more reliable information on the utility of each summarization method with better agreement levels. As another advantage, this method eliminates the need for defining golden standard relevance judgments.

For each summarization method, four different types of results can be identified by comparing the relevance judgments for the summaries and the original documents: *TP* (true positive), *FP* (false positive), *FN* (false negative) and *TN* (true negative) as in Table 6.2. According to these values, accuracy (*A*), recall (*R*), precision (*P*) and f-measure (*F*) values for the method can be calculated. *Accuracy* is calculated as the number of cases the user makes consistent judgment for the summary and the original document (i.e. both relevant or both irrelevant) divided by the number of judgments as in 6.3. *Recall* is calculated as the ratio of the number of documents identified as relevant using both the summaries and the original documents to the number of relevant judged original documents (see 4.1). *Precision* is calculated as the ratio of the documents identified as relevant using both the original documents and summaries to the number of summaries identified as relevant (see 4.2). *F-measure* is a combined measure of recall and precision (as defined in 4.3). Additional performance measures may also be defined. *False negative rate* is calculated as the number of false negatives over total number of actual positive instances as in 6.4. *False positive rate* is defined as the number of false positives over total number of actual negative instances as in 6.5. False negative rate corresponds to the cases relevant documents are missed by the user due to inadequate summaries. False positive rate corresponds to the cases where the users spend time by viewing irrelevant results although the summary seems relevant; this includes the time to loading the document in the browser and viewing the document. For all the performance measures, the results of each query set completed by a user are averaged.

Table 6.2. Contingency table for the summarization experiment

		Original document judgment	
		Relevant	**Irrelevant**
Summary judgment	**Relevant**	*TP*	*FP*
	Irrelevant	*FN*	*TN*

$$A \equiv \frac{TP + TN}{TP + TN + FP + FN} \tag{6.3}$$

$$FNR \equiv \frac{FN}{FN + TP} \tag{6.4}$$

$$FPR \equiv \frac{FP}{FP + TN} \tag{6.5}$$

6.2.4. Results

The summarization approach has been tested on the queries and document collections described in Chapter 3 (English Collection, Turkish Collection and Extended English Collection). In the following subsections, the results for different collections are presented.

6.2.4.1. English Collection. The queries used in this experiment cover different types of information need (see Q1-Q10 in Appendix A.1), including search for a number of items (7 queries), decision search (1 query) and background search (2 queries). The effectiveness of each method is shown in Table 6.3. A total of 300 judgments were made for each summarization method. In the table, the numbers of true positive, false positive, false negative and true negative judgments are given together with the average values of accuracy, precision, recall, and f-measure. First, we see that all three methods involving longer summaries (*Structured1*, *Structured2*, and *Unstructured*) perform significantly better than Google. Second, the structured summaries (*Structured1* and *Structured2*) are superior to unstructured ones. Third, we observe that *Structured2* (i.e., summaries using manually identified structure) performs the best under all performance measures, and the performance of the proposed fully automated method (*Structured1*) is quite similar to that of *Structured2*. Another point of view for the effectiveness of the methods is the false negative and false positive rates (Table 6.4). We see that structured summaries significantly reduce the amount of missed relevant results compared to Google and unstructured summaries (false negative rate). Also, structured summaries significantly reduce the lost time viewing irrelevant results (false positive rate).

Table 6.5 shows the performance improvement provided by the proposed method (*Structured1*) over *Google* and *Unstructured* methods. Especially when compared with a state-of-the-art search engine (Google), a significant increase in performance reveals itself (26.98% in f-measure). The performance increase is not so high when compared with unstructured summaries. However, the effect of structure identification is explicit. We also performed suitable statistical tests, i.e. repeated measures ANOVA (analysis of variance), on the results using SPSS Toolkit. The statistical tests verify that *Structured1* method yields significantly better results than both Google and unstructured summaries with p<0.001 for f-measure. We also measured the effect of summary sizes on decision times. Table 6.6 shows the average time of making judgments together with the average size of the summaries for each method and the original document. The values show that the proposed system has acceptable judgment times despite the much longer summary size.

Table 6.3. Results of the summarization experiment (English Collection)

System	**TP**	**FP**	**FN**	**TN**	**A**	**P**	**R**	**F**
Google	107	38	60	95	0.67	0.73	0.62	0.63
Unstructured	131	28	36	105	0.79	0.82	0.76	0.77
Structured1	137	25	30	108	0.82	0.85	0.80	0.80
Structured2	138	23	29	110	0.83	0.85	0.83	0.82

Table 6.4. False negative and false positive rates (English Collection)

System	**FNR**	**FPR**
Google	0.36	0.29
Unstructured	0.22	0.21
Structured1	0.18	0.19
Structured2	0.17	0.17

Table 6.5. Improvement of proposed system over other methods (English Collection)

System	**A**	**P**	**R**	**F**	**FNR**	**FPR**
Google	+22.39%	+16.44%	+29.03%	+26.98%	-50%	-34.48%
Unstructured	+3.80%	+3.66%	+5.26%	+3.90%	-18.18%	-9.52%

Table 6.6. Average judgment times versus average summary/document sizes (English Collection)

System	**Time (seconds)**	**Size (words)**
Google	14.58	41
Unstructured	27.24	278
Structured1	27.60	264
Structured2	28.58	253
Original	41.43	1566

6.2.4.2. Turkish Collection. The Turkish queries used in this experiment cover different types of information need (see Appendix A.2), including search for a number of items (2 queries) and background search (3 queries). The effectiveness of each method is shown in

Table 6.7. A total of 150 judgments were made for each summarization method. In the table, the numbers of true positive, false positive, false negative and true negative judgments are given together with the average values of accuracy, precision, recall, and f-measure. The results show that structured summaries (*Structured1* and *Structured2*) are superior to unstructured ones and Google snippets. It is interesting to note that *Structured1* performed better than *Structured2*. The effectiveness of each method is also given in Table 6.8 as the false negative and false positive rates. We see that structured summaries significantly reduce the amount of missed relevant results and lost time viewing irrelevant results compared to Google and unstructured summaries.

Table 6.7. Results of the summarization experiment (Turkish Collection)

System	**TP**	**FP**	**FN**	**TN**	**A**	**P**	**R**	**F**
Google	45	20	10	75	0.80	0.69	0.82	0.75
Unstructured	43	13	12	82	0.83	0.77	0.78	0.77
Structured 1	49	8	6	87	0.91	0.86	0.89	0.88
Structured 2	47	10	8	85	0.88	0.82	0.85	0.84

Table 6.8. False negative and false positive rates (Turkish Collection)

System	**FNR**	**FPR**
Google	0.18	0.21
Unstructured	0.22	0.14
Structured 1	0.11	0.08
Structured 2	0.15	0.11

Table 6.9 shows the performance improvement provided by the proposed method (*Structured1*) over *Google* and *Unstructured* methods. The proposed system has 17.33% improvement over Google and 14.29% improvement over unstructured summaries in terms of f-measure. The statistical tests (repeated measures ANOVA) we performed on the results verify that *Structured1* method yields significantly better results than both Google and unstructured summaries with $p<0.05$ for f-measure. Table 6.10 shows the average judgment times of the users and average sizes for each method and original documents. Although *Structured1*, *Structured2* and *Unstructured* methods provide summaries much longer than Google snippets, we see that the time increase in response time is moderate.

Table 6.9. Improvement of proposed system over other methods (Turkish Collection)

System	**A**	**P**	**R**	**F**	**FNR**	**FPR**
Google	+13.75%	+24.64%	+8.54%	+17.33%	-38.89%	-61.90%
Unstructured	+9.64%	+11.69%	+14.10%	+14.29%	-50%	-42.86%

Table 6.10. Average judgment times versus average summary/document sizes (Turkish Collection)

System	**Time (seconds)**	**Size (words)**
Google	11.04	30
Unstructured	19.96	216
Structured1	19.96	230
Structured2	19.71	235
Original	24.53	900

6.2.4.3. Extended English Collection. The queries used in this experiment cover different types of information need (see Q1-Q20 in Appendix A.1), including search for a number of items (12 queries), decision search (2 queries) and background search (6 queries). The structured summaries of the proposed system were created using the output of the machine learning approach for document structure analysis. Two different types of unstructured summaries were defined. In *Unstructured1*, the secondary parts (e.g. menus) of the document are not eliminated. In *Unstructured2*, this information is also eliminated as in structured summaries.

The effectiveness of each method is shown in Table 6.11 as the numbers of true positive, false positive, false negative, true negative judgments and the average values of accuracy, precision, recall, and f-measure. A total of 400 judgments were made for each summarization method. This experiment also verifies that structured summaries are superior to both Google extracts and unstructured summaries. The average values of false negative and false positive rates are given in Table 6.12. The results show that structured summaries significantly reduce the number missed results in the searches. The results of this experiment are not explicit for the false positive rate which corresponds to the time spent with irrelevant items.

Table 6.11. Results of the summarization experiment (Extended English Collection)

System	**TP**	**FP**	**FN**	**TN**	**A**	**P**	**R**	**F**
Google	118	36	120	126	0.57	0.72	0.47	0.52
Unstructured1	179	54	59	108	0.72	0.77	0.75	0.73
Unstructured2	176	53	62	109	0.72	0.77	0.73	0.72
Structured1	185	50	53	112	0.74	0.78	0.77	0.76
Structured2	183	40	55	122	0.75	0.82	0.76	0.77

Table 6.12. False negative and false positive rates (Extended English Collection)

System	**FNR**	**FPR**
Google	0.50	0.23
Unstructured1	0.23	0.32
Unstructured2	0.24	0.30
Structured1	0.20	0.30
Structured2	0.22	0.24

Table 6.13 shows the performance improvement provided by the proposed method (*Structured1*) over *Google*, *Unstructured1* and *Unstructured2* methods. As seen, the proposed system has significant improvement over Google extracts and unstructured summaries in terms of f-measure. The repeated measures ANOVA test also verify that the results are significant with $p<0.05$ for f-measure. Table 6.14 shows the average time of making judgments together with the average size of the summaries for each method and the original document. In this experiment, we also included an additional measure for rating the helpfulness of the summaries between 1 and 5. The average results of the ratings performed by the subjects are also given in Table 6.14. The results show that the structured summaries have very high ratings compared to Google extracts and unstructured summaries and values close to original documents.

Table 6.13. Improvement of proposed system over other methods (Extended English Collection)

System	**A**	**P**	**R**	**F**	**FNR**	**FPR**
Google	+30.68%	+9.66%	+63.88%	+44.97%	-59.65%	+29.80%
Unstructured1	+3.60%	+1.31%	+2.98%	+3.35%	-9.90%	-4.91%
Unstructured2	+3.14%	+1.79%	+5.42%	+4.90%	-16.31%	-0.30%

Table 6.14. Average judgment times, summary/document sizes and ratings (Extended English Collection)

System	**Time (seconds)**	**Size (words)**	**Rating**
Google	10.20	30	2.60
Unstructured1	17.70	298	2.77
Unstructured2	18.44	306	2.77
Structured1	17.51	277	3.03
Structured2	17.02	274	3.12
Original	23.59	1340	3.10

6.3. Discussion

The importance of summarization in information retrieval tasks was recognized in several studies related to human cognition. One of the studies compares the response time and accuracy of relevancy assessment for original documents and their summaries, and shows that the time decreases more or less linearly with the length while accuracy decreases only logarithmically [15]. This implies that we can gain from time substantially without a significant loss in accuracy when using summaries rather than original documents. This hypothesis was also supported by Marcu, who reports an experiment on an information retrieval task [51]. The time when summaries are used was found to be about 80% of the time required to perform the same task using the original documents, with recall and precision remaining approximately the same.

Current search engines use short extracts for displaying the results to the user. Such extracts focus only on the query words and thus miss the parts of the documents actually intended by the user. We have shown that a significant performance improvement is possible by using summaries much longer than the extracts generated by traditional search engines (e.g. Google). Longer summaries can show the parts of a document relevant to the user query explicitly even if those parts do not contain any of the query terms. We have also shown the importance of maintaining document structure in the summaries. Structured summaries increased the performance of the system significantly when compared with unstructured summaries of the same size. A structured summary provides an overview of the document and makes it for the user much easier to focus on the relevant parts which can be considered as some sort of semantic information for the user.

In current search engines, a major limitation for the user is the size of the display screen which constrains the number of results and the extracts. To optimize the number of results reviewed per screen, most of the search engines display a few lines of the document that include the query terms. However, this does not seem to be a suitable choice and it is argued that human cognition does not conform to this style of displays [78]. In this work, we combined the objective of displaying as many results as possible on a page with the objective of giving a detailed view for each result by using a dynamic summary window.

The high success rates of the *Structured1* and *Structured2* methods indicate that combining structural processing with summary extraction is a convenient approach. The size of a summary prepared with these methods ranges about between 15-25% of the corresponding document on the average. By looking at a summary of this size, the users are able to determine the relevancy with about 75-90% correctness. When we compare these two methods with each other, we see that they give similar success rates. *Structured2* may be expected to yield a better performance since it is based on manually identified hierarchical structures of documents. This is indeed the case for both English collections. However, interestingly, *Structured1* shows a little better performance than *Structured2* in the Turkish collection. We conjecture this result to the two-stage nature of the process in the sense that the summarization component can work on an imperfect structure and humans are good at coping with vagueness. Based on the high performance of the proposed method, *Structured1*, we can conclude that it is a fully automatic method that can be incorporated into a search engine.

An analysis of user response times (Table 6.6, Table 6.10 and Table 6.14) shows that summaries, although they are 6-9 times longer than Google extracts, cause only less than two times increase in response time. This indicates that people just look at the related parts of the summaries and then arrive at a decision. Thus we see that the proposed system has acceptable user response times despite the much longer summary sizes.

In this research, the main aim is not only to reduce the searching time, but to balance the time spent viewing the document summaries with the accuracy obtained in the given task. In this context, false negative rates of the summarization methods become important. The experiments show that Google extracts result in high false negative rates which correspond to the cases relevant documents are incorrectly identified as irrelevant; thus, the users miss useful items of information. In fact, the importance of false negative rates depends on the type of the query. This rate becomes especially important in the case of background search where more than one result may be necessary (e.g. the query *newspapers electronic media* where the aim is to find the effects of the electronic media on the newspaper industry) as well as specific and complex queries (e.g. the query *creativity* where the aim is to find ways of measuring creativity).

Another point of view for the results is the false positive rate of a summarization method which corresponds to the cases irrelevant documents are identified as relevant and as a result, users spend time examining them unnecessarily. That is, when the user clicks on the link of an irrelevant document, he/she will spend some time during page loading and to understand that the document is in fact irrelevant. We can define the time overhead of a summarization method as in 6.6 where $T_{summary}$ is the average time to view the summary of a particular system, $T_{document}$ is the average time to examine the original document and T_{page_load} is the time to load the original document into the browser (depending on the speed of the Internet connection). The time overhead is calculated as the sum of time spent viewing the summaries and the time spent with irrelevant items (i.e., false positives).

$$\text{Time Overhead} = \text{Number of Results Viewed} \cdot T_{summary} + FP \cdot (T_{page\_load} + T_{document}) \qquad (6.6)$$

By substituting the average values obtained in the experiments into the formula, we see that the proposed system summaries (*Structured1*) result in a 55-60% increase of time compared to Google extracts. However, the number of relevant documents missed using the Google extracts reaches about two times the ones using the proposed system summaries. Thus, we see that there is a tradeoff between the time spent with the summaries and the accuracy obtained. Our justification regarding the superiority of the proposed system summaries is as follows. In the case of common-place queries (e.g. *the population of Germany*), the users can locate the relevant information just by viewing a few of the top results of the search engine; as a result, the total time spent and the overhead of the summarization method is less important. In the case of more complex queries and background search, the accuracy provided by the summarization method becomes more important. The proposed system is preferable because it results in a reduced number of missed items (false negative rates) compared with the other systems. Also, using the proposed system can usually be less tedious for users in these tasks because of low false positive rates. That is, in the proposed system, the users usually spent less time examining the irrelevant documents. Finally, the proposed system has also received very high user ratings compared with Google and unstructured summaries. This verifies that the structured summaries have also been found more helpful and preferable by users.

An analysis of the time complexity of a search engine built on the proposed techniques shows that it is linear in document length. Document structural processing step (detailed in chapters 4 and 5) is independent of the user query and needs to be performed once for each document. This process can be done offline similar to the indexing phase of search engines. Summary extraction step (detailed in this chapter) has a linear time complexity. Given a document hierarchy with n nodes (sentences), the extraction algorithm (Figure 6.1) operates on at most n nodes in a top-down fashion. At each node, the quota of the node is calculated and the sentences are selected based on the quota if a threshold is reached, both of which require constant time. As a result, the time complexity of the whole process is $O(n)$.

7. CONCLUSION

There is a drastic increase of information sources on the World Wide Web. Search engines provide a means for Internet users to locate documents on the Web via queries. However, the users still have to complete the sifting process by themselves; that is, to decide on the relevance of the returned documents according to their actual needs. During Web search, one aid of users is the short document summaries (extracts) provided in the search results. However, the summaries provided by current search engines have limitations in directing users to relevant documents. As a result, the users often miss relevant results or spend time with irrelevant ones.

In this thesis, we developed a novel summarization approach, i.e. structure-preserving and query-biased summarization, to improve the effectiveness of Web search. It is a query-biased method utilizing document structure both during the summarization process and in the output summaries. In this way, the context of searched terms is preserved in the output summaries. To the best of our knowledge, it is the first approach using both explicit document structure and query-biased techniques in Web search context.

The proposed system has been developed in Java by utilizing GATE Text Engineering Framework which is an open source framework widely used in both academic and commercial projects of human language technology, and open source Cobra HTML Parser and Renderer Toolkit. We created English and Turkish document collections from the results of Google in response to several search queries that reflect current search interests of users in various domains.

The proposed approach is composed of two stages: structural processing and summary extraction. In the first stage, we considered the rather unexplored problem of heading-based sectional hierarchy extraction for unrestricted domain of Web documents. We represented a document as an ordered tree in which headings and subheadings are at intermediate nodes, and other text units are at the leaves. We first developed a rule-based approach for structural processing based on heuristics and HTML DOM tree processing.

In the progress, we developed a machine learning approach for structural processing of Web documents because it can be more flexible than a rule-based approach. For sectional hierarchy extraction, a tree-based learning approach was needed rather than the simpler case of classification. To overcome the exponential search space encountered in the solution, we developed an incremental learning approach based on beam search. We defined several features for learning, including features to represent the context of a unit in a document tree. We developed several variations of the approach using SVM and perceptron algorithms, and evaluated them using cross-validation. For extraction of headings, we obtained an f-measure of 85%. For hierarchy extraction, an accuracy of 82% was obtained with manually identified headings, whereas 68% accuracy was obtained using the output of heading extraction. The machine learning approach, as a fully automatic approach, performed better than the rule-based approach on the same document set. There are not many studies in the literature on heading-based analysis of Web documents. One related work is about the identification of the main title of an HTML document in which a maximum f-measure of 80% was obtained [39]. In the proposed system, we investigated a more challenging and general task; that is, the identification of all the headings in a document together with their levels in the sectional hierarchy.

The second stage of the proposed system is summary extraction where the documents are summarized with respect to the user queries and the structural information obtained in the first stage. We adapted basic statistical techniques and created indicative summaries based on two levels of processing: sentence scoring and section scoring. The structure is preserved in the output summaries by providing the context of extracted sentences in the form of headings and subheadings. The effectiveness of the proposed system was compared with Google extracts and unstructured summaries of the same size on task-based evaluations using a Web-based interface. We used a within subjects design and the recent relevance prediction approach. The approach has been applied to Turkish document collections and queries as well as English ones. This makes it the first automatic summarization study of Turkish targeting Web search.

The overall performance of the proposed summarization approach was measured about 75-90%. This corresponds to a significant performance improvement compared with a state-of-the-art search engine, Google. Also, there was a statistically significant performance gain when the document structures were preserved as compared to unstructured summaries. The accuracy obtained in the structural processing stage was also proved to be an acceptable performance for the summarization phase: the results of using the proposed structural processing are very close to the results where manually identified hierarchical structures of documents are used. This indicates that the errors of hierarchy extraction step can well be tolerated during later processing. Thus, we conclude that structure-preserving and query-biased summarization, as a fully automatic method, greatly influences the accuracy of Web search tasks.

The experiments also showed that the proposed system has acceptable user response times despite the much longer summary sizes compared to search engine extracts. Thus, we have reached our main goals for summarization as defined in Chapter 6; that is, to enable users to make more accurate relevance judgments as compared to search engine extracts in reasonable times. The proposed system resulted in great reduction in false negative rates; i.e., reducing the cases where the users miss relevant results. Also, the system usually resulted in reduced false positive rates, causing users to spend less time with irrelevant items. Finally, the user ratings also showed that the proposed system summaries are found to be more helpful than other system summaries. Time complexity analysis for the proposed system revealed that it is a practical approach which can be incorporated into a search engine. The structural processing stage can be performed offline and once; the summarization stage has linear time complexity.

In addition to the domain of search engines, the proposed approach can also be utilized in several other fields related to document processing. Information systems where large amounts of document need to be analyzed such as library systems, law, and medicine are the typical candidates. The methods proposed in this study allow browsing large documents with the help of the structural information. The structural information may also be used for classification and indexing of documents. Also, the summaries provided by the system can be taken as an outline in creating manual summaries. The proposed system has applications including display of Web content on small-screen devices such as PDAs.

There may be two directions for future work: structural processing and summary extraction. An issue in structural processing is to correctly identify some document components commonly encountered in Web pages, such as identification of menus, references and advertisements. Here, we have a heuristic-based approach which can be extended using machine learning. Such information can also be used in the summarization process to eliminate irrelevant information and improve the summaries. Currently, phrases in queries and stemming are used as linguistic information in the summarization process. The summarization approach can be extended using linguistic information in syntactic and semantic levels; such as incorporation of syntactic phrases and WordNet [45]. As an example, query terms can be extended using synonyms, "is-a" and "part-of" relations based on WordNet. New query-biased methods may be developed for the scoring of sentences. The effects of different search tasks, such as searching for a particular fact or searching for background information about a subject, can be investigated to refine the system. Also, the effects of different document types (i.e. genre) can be considered for summarization. Finally, another issue is the automatic evaluation of the summaries.

REFERENCES

1. Mani, I. and M. T. Maybury (editors), *Advances in Automatic Text Summarization*, MIT Press, Cambridge, 1999.

2. Jansen, B. J. and A. Spink, “An Analysis of Web Searching by European AlltheWeb.com Users”, *Information Processing and Management*, Vol. 41, No. 2, pp. 361-381, 2005.

3. Google, http://www.google.com, 2010.

4. AltaVista, http://www.altavista.com, 2010.

5. Text Retrieval Conference (TREC), http:// trec.nist.gov, 2010.

6. Luhn, H. P., “The Automatic Creation of Literature Abstracts”, *IBM Journal of Research and Development*, Vol. 2, No. 2, pp. 159-165, 1958.

7. Sparck Jones, K., “Automatic Summarizing: Factors and Directions”, in I. Mani and M. T. Maybury (eds.), *Advances in Automatic Text Summarization*, pp. 1-12, MIT Press, Cambridge, 1999.

8. Mao, S., A. Rosenfeld and T. Kanungo, “Document Structure Analysis Algorithms: A Literature Survey”, *Proceedings of SPIE Electronic Imaging*, pp.197-207, 2003.

9. Pembe, F. C. and T. Güngör, “Automated Query-biased and Structure-preserving Text Summarization on Web Documents”, *Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications*, İstanbul, 2007.

10. Pembe, F. C. and T. Güngör, “Heading-Based Sectional Hierarchy Identification for HTML Documents”, *Proceedings of the 22nd International Symposium on Computer and Information Sciences*, Ankara, IEEE Xplore, 2007.

11. Pembe, F. C. and T. Güngör, “A Tree Learning Approach to Web Document Sectional Hierarchy Extraction”, *Proceedings of 2nd International Conference on Agents and Artificial Intelligence*, Valencia, 2010.

12. Pembe, F. C. and T. Güngör, “Towards a New Summarization Approach for Search Engine Results: An Application for Turkish”, *Proceedings of the 23rd International Symposium on Computer and Information Sciences*, Istanbul, IEEE Xplore, 2008.

13. Pembe, F. C. and T. Güngör, “An Evaluation of Structure-Preserving and Query-Biased Summaries in Web Search Tasks”, *Computer Engineering*, Vol. 1, No. 2, p.62-66, 2007.

14. Pembe, F. C. and T. Güngör, “Structure-Preserving and Query-Biased Document Summarization for Web Search”, *Online Information Review*, Vol. 33, No. 4, p.696-719, 2009.

15. Jackson, P. and I. Moulinier, *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*, John Benjamins Pub., Amsterdam, 2007.

16. Kowalski, G.J. and M. T. Maybury, *Information Storage and Retrieval Systems: Theory and Implementation*, Kluwer Academic, Boston, MA, 2002.

17. Chowdhury, G. G., *Introduction to Modern Information Retrieval*, Facet Pub., London. 2006.

18. Broder, A. and M. Henzinger, “Algorithmic Aspects of Information Retrieval on the Web”, in J. Abello, P. M. Pardalos and M. G. C. Resende (eds.), *Handbook of Massive Data Sets*, pp. 3-23, Kluwer Academic, Dordrecht, 2002.

19. Baeza-Yates, R. and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, New York, NY, 1999.

20. Berry, M. W. and M. Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, SIAM, Philadelphia, 1999.

21. Ingwersen, P. and K. Järvelin, *The Turn: Integration of Information Seeking and Retrieval in Context*, Springer, Dordrecht, 2005.

22. White, R. W., J. M. Jose and I. Ruthven, "A Task-oriented Study on the Influencing Effects of Query-biased Summarization in Web Searching", *Information Processing and Management*, Vol. 39, No. 5, pp. 707-733, 2003.

23. Grossman, D. A. and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, Springer, Dordrecht, 2004.

24. Curran, J. R. and R. K. Wong, "Transformation-based Learning for Automatic Translation from HTML to XML", *Proceedings of the Fourth Australasian Document Computing Symposium*, 1999.

25. Brugger, R., A. Zramdini and R. Ingold, "Modeling Documents for Structure Recognition Using Generalized N-grams", *Proceedings of International Conference on Document Analysis and Recognition*, pp. 56–60, 1997.

26. Shilman, M., P. Liang, and P. Viola, "Learning Non-generative Grammatical Models for Document Analysis", *Proceedings of the Tenth IEEE International Conference on Computer Vision*, 2005.

27. Collins, M. and B. Roark, "Incremental Parsing with the Perceptron Algorithm", *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, 2004.

28. Branavan, S. R. K., P. Deshpande and R. Barzilay, "Generating a Table-of-Contents", *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

29. HTML 4.01 Specification, http://www.w3.org/TR/html401/, 1999.

30. Chaudhuri, B. B., *Digital Document Processing: Major Directions and Recent Advances*, Springer, London, 2006.

31. Gupta, S., G. E. Kaiser, P. Grimm, M. F. Chiang and J. Starren, "Automating Content Extraction of HTML Documents", *World Wide Web*, Vol. 8, No. 2, pp. 179–224, 2005.

32. Chung, C. Y., M. Gertz and N. Sundaresan, "Reverse Engineering for Web Data: From Visual to Semantic Structures", *Proceedings of the 18th International Conference on Data Engineering*, 2002.

33. Yang, Y. and H. J. Zhang, "HTML Page Analysis Based on Visual Cues", *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, 2001.

34. Chen, Y., W. Ma and H. Zhang, "Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices", *Proceedings of the 12th International World Wide Web Conference*, 2003.

35. Buyukkokten, O., H. Garcia-Molina and A. Paepcke, "Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2001.

36. Mukherjee, S., G. Yang, W. Tan and I. V. Ramakrishnan, "Automatic Discovery of Semantic Structures in HTML Documents", *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003.

37. Document Object Model, http://www.w3.org/DOM/, 2005.

38. Vadrevu, S., F. Gelgi and H. Davulcu, "Information Extraction from Web Pages Using Presentation Regularities and Domain Knowledge", *World Wide Web*, Vol. 10 No. 2, pp. 157-179, 2007.

39. Xue, Y., Y. Hu, G. Xin, R. Song, S. Shi, Y. Cao, C. Y. Lin and H. Li, “Web Page Title Extraction and Its Application”, *Information Processing and Management*, Vol. 43, No. 5, pp. 1332-1347, 2007.

40. Feng, J., P. Haffner and M. Gilbert, “A Learning Approach to Discovering Web Page Semantic Structures”, *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pp. 1055-1059, 2005.

41. Song, R., H. Liu, J. Wen and W. Ma, “Learning Block Importance Models for Web Pages”, *Proceedings of the 13th International Conference on World Wide Web*, pp. 203-211, 2004.

42. Mani, I, *Automatic Summarization*, J. Benjamins Pub., Amsterdam, 2001.

43. Sparck Jones, K., “Automatic Summarising: The State of the Art”, *Information Processing and Management*, Vol. 43, No. 6, pp. 1449-1481, 2007.

44. Edmundsun, H. P., “New Methods in Automatic Extracting”, *Journal of the ACM*, Vol. 16, No. 2, pp. 265-285, 1969.

45. WordNet, http://wordnet.princeton.edu, 2010.

46. I. Mani, G. Klein, D. House, L. Hirschman, T. Firmin and B. Sundheim, "SUMMAC: A Text Summarization Evaluation", *Natural Language Engineering*, Vol. 8, No. 1, pp. 43-68, Cambridge University Press, 2002.

47. Mochizuki, H. and M. Okumura, “A Comparison of Summarization Methods Based on Task-based Evaluation”, *Proceedings of the Second International Conference on Language Resources and Evaluation*, pp. 633–639, 2000.

48. J. Kupiec, J. Pedersen and F. Chen, "A Trainable Document Summarizer", *Proceedings of the 18th ACM-SIGIR Conference*, pp. 68-73, 1995.

49. Barzilay, R. and M. Elhadad, "Using Lexical Chains for Text Summarization”, in I. Mani and M. T. Maybury (eds.), *Advances in Automatic Text Summarization*, pp. 111-121, MIT Press, Cambridge, 1999.

50. Marcu, D., "Discourse Trees Are Good Indicators of Importance in Text”, in I. Mani and M. T. Maybury (eds.), *Advances in Automatic Text Summarization*, pp. 123-136, MIT Press, Cambridge, 1999.

51. Marcu, D., *The Theory and Practice of Discourse Parsing and Summarization*, MIT Press, Cambridge, 2000.

52. Guo, Y. and G. Stylios, “An Intelligent Summarisation System Based on Cognitive Psychology”, *Information Sciences*, Vol. 174, No. 1-2, pp. 1–36, 2005.

53. Yeh, J. Y., H. R. Ke, W. P. Yang and I. H. Meng, “Text Summarization Using a Trainable Summarizer and Latent Semantic Analysis”, *Information Processing and Management*, Vol. 41, No. 1, pp. 75–95, 2005.

54. Otterbacher, J., D. Radev and O. Kareem, “Hierarchical Summarization for Delivering Information to Mobile Devices”, *Information Processing and Management*, Vol. 44, No. 2, pp. 931-947, 2008.

55. Liang, S.F., S. Devlin and J. Tait, “Investigating Sentence Weighting Components for Automatic Summarization”, *Information Processing and Management*, Vol. 43, No. 1, pp. 146-153, 2007.

56. Tombros, A. and M. Sanderson, “Advantages of Query Biased Summaries in Information Retrieval”, *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador*, pp. 2-10, 1998.

57. Varadarajan R. and V. Hristidis, “Structure-Based Query-Specific Document Summarization”, *Proceedings of the 14th ACM international conference on Information and Knowledge Management*, 2005.

58. Alam, H., A. Kumar, M. Nakamura, A. F. R. Rahman, Y. Tarnikova and C. Wilcox, “Structured and Unstructured Document Summarization: Design of a Commercial Summarizer Using Lexical Chains”, *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pp. 1147-1150, 2003.

59. Yang, C. C. and F. L. Wang, “Hierarchical Summarization of Large Documents”, *Journal of the American Society for Information Science and Technology*, Vol. 59, No. 6, pp. 887-902, 2008.

60. Szlávik, Z., A. Tombros and M. Lalmas, “Investigating the Use of Summarisation for Interactive XML Retrieval”, *Proceedings of the 2006 ACM Symposium on Applied Computing*, 2006.

61. Amini M. R., A. Tombros, N. Usunier and M. Lalmas, “Learning Based Summarization of XML Documents”, *Journal of Information Retrieval*, Vol. 10, No. 3, pp. 233-255, 2007.

62. GATE, A General Architecture for Text Engineering, http://gate.ac.uk/, 2010.

63. Maynard, D., K. Bontcheva, H. Saggion, H. Cunningham and O. Hamza, “Using a Text Engineering Framework to Build an Extendable and Portable IE-based Summarisation System”, *Proceedings of the ACL Workshop on Text Summarisation*, 2002.

64. Porter, M., “An Algorithm for Suffix Stripping”, *Program*, Vol. 14, No. 3, pp. 130-137, 1980.

65. Cobra: Java HTML Renderer & Parser, http://lobobrowser.org/cobra.jsp, 2010.

66. Ko, Y., H. An and J. Seo, “Pseudo-relevance Feedback and Statistical Query Expansion for Web Snippet Generation,” *Information Processing Letters*, Vol. 109, No. 1, pp.18-22, 2008.

67. Chen, L. and W. L. Chue, “Using Web Structure and Summarisation Techniques for Web Content Mining,” *Information Processing and Management*, Vol. 41, No. 5, pp. 1225-1242, 2005.

68. Markey, K., “Twenty-five Years of End-user Searching, Part 1: Research Findings”, *Journal of the American Society for Information Science and Technology*, Vol. 58, No. 8, pp. 1071–1081, 2007.

69. Can, F., S. Kocberber, E. Balcik, C. Kaynak, H. C. Ocalan and O. M. Vursavas, "Information Retrieval on Turkish Texts", *Journal of the American Society for Information Science and Technology*, Vol. 59, No. 3, pp. 407-421, 2008.

70. Hall, J., J. Nivre and J. Nilsson, “Discriminative Classifiers for Deterministic Dependency Parsing”, *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pp. 316-323, 2006.

71. Covington, M. A., “A Fundamental Algorithm for Dependency Parsing”, *Proceedings of ACM Southeast Conference*, 2001.

72. Platt, J. C., “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”, in A. Smola, P.Bartlett, B. Scholkopf and D. Schuurmans (eds.), *Advances in Large Margin Classifiers*, pp. 61-74, MIT Press, Cambridge, 1999.

73. Mayfield, J., P. McNamee, C. Piatko and C. Pearce, “Lattice-based Tagging Using Support Vector Machines”, *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pp. 303-308, 2003.

74. E. Alpaydın, *Introduction to Machine Learning*, MIT Press, Cambridge, 2004.

75. T. Joachims, “Making Large-Scale SVM Learning Practical”, in B. Schölkopf, C. Burges and A. Smola (eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999.

76. Montgomery, D. C., *Design and Analysis of Experiments*, John Wiley, New York, 2001.

77. Hobson, S. P., B. J. Dorr, C. Monz and R. Schwartz, “Task-Based Evaluation of Text Summarization Using Relevance Prediction”, *Information Processing and Management*, Vol. 43, No. 6, pp.1482-1499, 2007.

78. Van Oostendorp, H. V. and S. de Mul, *Cognitive Aspects of Electronic Text Processing*, Ablex Pub., New Jersey, 1996.

APPENDIX A: QUERIES USED IN THE EXPERIMENTS

A.1. Turkish Queries

Q1:

Title: Tsunami

Description: Güney Asya’yı 26 Aralık 2004’te vuran büyük Tsunami faciası ve bu facianın sonuçları.

Narrative: Güney Asya’da 26 Aralık 2004 tarihinde meydana gelen 8.9 büyüklüğündeki deprem ve bu deprem sonrasında oluşan dev dalgalar, bu facia sonrasında hangi ülkede kaç kişinin öldüğüne dair bilgiler, ve afetten sonra diğer ülkeler tarafından yapılan yardımları içeren bir doküman.

Q2:

Title: Ekonomik kriz

Description: Türkiye'de ekonomik krize neden olan olaylar.

Narrative: Türkiye'de son bir kaç yıl içinde olan ekonomik krizlerin nedenleri ve bunlara zemin hazırlayan olaylar.

Q3:

Title: Türkiye'de meydana gelen depremler

Description: Türkiye'de meydana gelen depremlerin insanlar üzerindeki etkileri ve bu depremlere karşı alınan önlemler.

Narrative: Türkiye'de meydana gelen depremlere karşı insanların aldığı eğitim ve önlemler. Depremlerin, meydana geldikten sonra insanlarda bıraktığı etkiler ve devletin depremlerden sonra aldığı önlemler.

Q4:

Title: Sanat ödülleri

Description: Türkiye'de edebiyat, müzik, resim, sinema gibi sanat dallarında verilmiş ödüller.

Narrative: Türkiye'de sanatın değişik dallarına ne gibi ödüller, hangi yıllarda kimlere verilmiş. Bu ödüllerin sebepleri, sonuçları, etkileri...

Q5:

Title: Bilişim eğitimi ve projeleri

Description: Türkiye'de yapılan bilişim eğitimi ve bilişim projeleri, bu eğitimin ve projelerin kaliteleri ve sanayiye katkıları

Narrative: Türkiye'de yapılan bilişim eğitimi ve projelerinin süreçleri, sorunları ve ülkeye sağladığı katkılar. Bu eğitimin ve projelerin yaygınlaştırılması konusunda görüş ve öneriler

A.2. English Queries

Q1:

Title: Hubble Telescope Achievements

Description: Identify positive accomplishments of the Hubble telescope since it was launched in 1991.

Narrative: Documents are relevant that show the Hubble telescope has produced new data, better quality data than previously available, data that has increased human knowledge of the universe, or data that has led to disproving previously existing theories or hypotheses. Documents limited to the shortcomings of the telescope would be irrelevant. Details of repairs or modifications to the telescope without reference to positive achievements would not be relevant.

Q2:

Title: Best Retirement Country

Description: Aside from the United States, which country offers the best living conditions and quality of life for a U.S. retiree?

Narrative: A relevant document will contain information describing the living conditions and/or costs in one or more foreign countries. It will provide information that a potential retiree could use in deciding where to establish a retirement home.

Q3:

Title: Literary/Journalistic Plagiarism

Description: Find instances of plagiarism in the literary and journalistic worlds.
Narrative: A relevant document will report any occasion or suspected instance of plagiarism in the areas of either literature or journalism. Relevant documents will also include such areas as doctorate and master's theses and will encompass writings as well as the ideas and concepts developed by some authors and taken or borrowed by others without attribution.

Q4:
Title: Mexican Air Pollution
Description: Mexico City has the worst air pollution in the world. Pertinent Documents would contain the specific steps Mexican authorities have taken to combat this deplorable situation.
Narrative: Relevant documents would discuss the steps the Mexican Government has taken to alleviate the air pollution in Mexico City. Steps such as reducing the number of automobiles in the city, encouraging the use of mass public transportation, and creating new mass transportation systems are relevant, among others. Mention of any new methods in the design stage would also be appropriate.

Q5:
Title: Antibiotics Bacteria Disease
Description: Determine the reasons why bacteria seems to be winning the war against antibiotics and rendering antibiotics now less effective in treating diseases than they were in the past.
Narrative: A relevant document will address the questions of how and why, and to what degree, bacteria are able to fend off the curative effects of antibiotics. Overuse of antibiotics, as well as the increasing use of antibiotics in promoting the growth of crops and animals whose food products are meant for human consumption have played roles in creating a situation where every known bacteria-generated disease now has versions that resist at least one of the more than 100 antibiotics now in use.

Q6:
Title: Abuses of E-Mail

Description: The availability of E-mail to many people through their job or school affiliation has allowed for many efficiencies in communications but also has provided the opportunity for abuses. What steps have been taken world-wide by those bearing the cost of E-mail to prevent excesses?

Narrative: To be relevant, a document will concern dissatisfaction by an entity paying for the cost of electronic mail. Particularly sought are items which relate to system users (such as employees) who abuse the system by engaging in communications of the type not related to the payer's desired use of the system.

Q7:

Title: declining birth rates

Description: Do any countries other than the U.S. and China have a declining birth rate?

Narrative: To be relevant, a document will name a country other than the U.S. or China in which the birth rate fell from the rate of the previous year. The decline need not have occurred in more than the one preceding year.

Q8:

Title: human genetic code

Description: What progress is being made in the effort to map and sequence the human genetic code?

Narrative: Documents must discuss specific progress in mapping the human genome. Documents that simply describe applications of the research, such as using DNA in criminal cases, using the genetic code to treat disease, or creating genetically engineered organisms are irrelevant.

Q9:

Title: mental illness drugs

Description: Identify drugs used in the treatment of mental illness.

Narrative: A relevant document will include the name of a specific or generic type of drug. Generalities are not relevant.

Q10:

Title: literacy rates Africa

Description: What are literacy rates in African countries?

Narrative: A relevant document will contain information about the literacy rate in an African country. General education levels that do not specifically include literacy rates are not relevant.

Q11:
Title: robotic technology
Description: What are the latest developments in robotic technology?
Narrative: A relevant document will contain information on current applications of robotic technology. Discussions of robotics research or simulations of robots are not relevant.

Q12:
Title: creativity
Description: Find ways of measuring creativity.
Narrative: Relevant items include definitions of creativity, descriptions of characteristics associated with creativity, and factors linked to creativity.

Q13:
Title: tourism increase
Description: What countries are experiencing an increase in tourism?
Narrative: A relevant document will name a country that has experienced an increase in tourism. The increase must represent the nation as a whole and tourism in general, not be restricted to only certain regions of the country or to some specific type of tourism (e.g., adventure travel). Documents discussing only projected increases are not relevant.

Q14:
Title: newspapers electronic media
Description: What has been the effect of the electronic media on the newspaper industry?
Narrative: Relevant documents must explicitly attribute effects to the electronic media: information about declining readership is irrelevant unless it attributes the cause to the electronic media.

Q15:
Title: wildlife extinction

Description: The spotted owl episode in America highlighted U.S. efforts to prevent the extinction of wildlife species. What is not well known is the effort of other countries to prevent the demise of species native to their countries. What other countries have begun efforts to prevent such declines?
Narrative: A relevant item will specify the country, the involved species, and steps taken to save the species.

Q16:
Title: R&D drug prices
Description: Identify documents that discuss the impact of the cost of research and development (R&D) on the price of drugs.
Narrative: Documents that describe how any aspect of the development of a drug affects its price are relevant. Documents that discuss other factors that affect drug prices, such as advertising, without also discussing R&D costs, are not relevant.

Q17:
Title: Amazon rain forest
Description: What measures are being taken by local South American authorities to preserve the Amazon tropical rain forest?
Narrative: Relevant documents may identify: the official organizations, institutions, and individuals of the countries included in the Amazon rain forest; the measures being taken by them to preserve the rain forest; and indications of degrees of success in these endeavors.

Q18:
Title: osteoporosis
Description: Find information on the effects of the dietary intakes of potassium, magnesium and fruits and vegetables as determinants of bone mineral density in elderly men and women thus preventing osteoporosis (bone decay).
Narrative: A relevant document may include one or more of the dietary intakes in the prevention of osteoporosis. Any discussion of the disturbance of nutrition and mineral metabolism that results in a decrease in bone mass is also relevant.

Q19:

Title: alternative medicine

Description: What forms of alternative medicine are being used in the treatment of illnesses or diseases and how successful are they?

Narrative: A relevant document should identify a form of alternative medicine which is being utilized in the treatment of a disease or illness, identify the illness or disease being treated, and provide an indication of the success of the procedure.

Q20:

Title: health and computer terminals

Description: Is it hazardous to the health of individuals to work with computer terminals on a daily basis?

Narrative: Relevant documents would contain any information that expands on any physical disorder/problems that may be associated with the daily working with computer terminals. Such things as carpel tunnel, cataracts, and fatigue have been said to be associated, but how widespread are these or other problems and what is being done to alleviate any health problems.