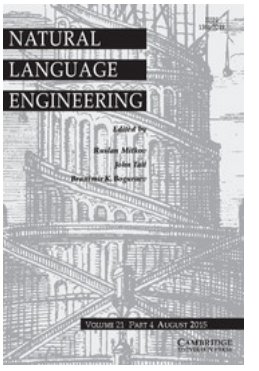


# A tree-based learning approach for document structure analysis and its application to web search

F. CANAN PEMBE and TUNGA GÜNGÖR

Natural Language Engineering / Volume 21 / Issue 04 / August 2015, pp 569 - 605
DOI: 10.1017/S1351324914000023, Published online: 24 February 2014

*Natural Language Engineering* **21** (4): 569–605. © Cambridge University Press 2014
doi:10.1017/S1351324914000023


# *A tree-based learning approach for document structure analysis and its application to web search*

F. CANAN PEMBE[1] and TUNGA GÜNGÖR[2]
[1]*TÜBİTAK BİLGEM, 41470, Gebze, Kocaeli, Turkey*
*e-mail:* canan.pembe@tubitak.gov.tr
[2]*Department of Computer Engineering, Boğaziçi University, İstanbul, 34342, Turkey*
*e-mail:* gungort@boun.edu.tr

(*Received 9 October 2012; revised 23 January 2014; accepted 24 January 2014; first published online 24 February 2014*)

**Abstract**

In this paper, we study the problem of structural analysis of Web documents aiming at extracting the sectional hierarchy of a document. In general, a document can be represented as a hierarchy of sections and subsections with corresponding headings and subheadings. We developed two machine learning models: heading extraction model and hierarchy extraction model. Heading extraction was formulated as a classification problem whereas a tree-based learning approach was employed in hierarchy extraction. For this purpose, we developed an incremental learning algorithm based on support vector machines and perceptrons. The models were evaluated in detail with respect to the performance of the heading and hierarchy extraction tasks. For comparison, a baseline rule-based approach was used that relies on heuristics and HTML document object model tree processing. The machine learning approach, which is a fully automatic approach, outperformed the rule-based approach. We also analyzed the effect of document structuring on automatic summarization in the context of Web search. The results of the task-based evaluation on TREC queries showed that structured summaries are superior to unstructured summaries both in terms of accuracy and user ratings, and enable the users to determine the relevancy of search results more accurately than search engine snippets.

## 1 Introduction

Information retrieval research generally focuses on documents in electronic form which are prepared for access and browsing by humans, as in the case of Web information retrieval. Documents on the Web are usually prepared in hypertext markup language (HTML) format and automatic analysis of them is not considered during the preparation process. However, with the drastic increase of the number of documents available on the Web, automatic document analysis has begun to be important in various application areas, including search engines and summarization.

The structural and layout analysis of Web documents is a challenging task for mainly two reasons. The first one is the practice employed by Web designers while

authoring a Web page. During this process, the designers concentrate on properties related to the visual appearance of documents (Klink, Dengel and Kieninger 2000). Although a set of tags is provided by the markup language to enforce a structural layout, the users often prefer to use some stylistic features (e.g. formatting features) to obtain the desired effect. The intuition behind this decision is the ease of using such properties and the flexibility attained compared to adhering to a restricted set of tags. The HTML format is not intended for a semantic representation of the data and hence the tags do not always correspond to a meaningful division of the content.

The second reason is the complex organization of content in Web documents. Web documents are heterogeneous documents containing different types of entities such as text, images, interactive forms, menus, advertisements, and tables. The designers can freely organize the content and use all these entities in a complicated manner. This causes Web documents to have a very complex layout. In order to understand the contents of a document, human readers use several cues such as the context, conventions and information about the language, along with a reasoning mechanism (Chaudhuri 2006). However, automating this process and revealing the structure of Web documents pose several difficulties for computer applications.

In the literature, there has been extensive research on general document analysis and recently the analysis of Web documents has gained considerable attention. As will be detailed in Section 2, most of these studies focus on flat or hierarchical segmentation of documents and headings in the documents are not utilized during this process. The problem of sectional hierarchy extraction based on the information provided by the headings in Web documents was investigated in a few studies using a rule-based approach (i.e. an approach based on a set of manually constructed rules). In general, a rule-based approach is less adaptive and less robust when compared with a machine learning approach (Desmarais, Gagnon and Zouaq 2012). The number of features used to model the contents of Web pages is large and it is not easy to build a rule set that covers all the usage patterns. Also, the existence of tagging errors made during the design of a page prevents such cases to be handled by built-in rules (Li *et al.* 2013). A machine learning approach can be more flexible in the sense that a classifier can combine different features to make a decision rather than using a predefined set of rules to filter the headings. In addition, the rules may require a significant amount of manual effort to generate and they do not scale easily in terms of their coverage and between domains (Irmak and Kraft 2010).

Based on these observations, in this paper, we propose methods for the structural analysis of Web documents using machine learning techniques. The target of the system is general Web documents without any domain restriction. Web documents are diverse in structure, having sections and subsections with different formatting and topics. In this study, we develop two main machine learning models: heading extraction model and hierarchy extraction model. The heading extraction model investigates the classification of text units in a document as heading or not. The aim of the hierarchy extraction model is to build the document sectional hierarchy (i.e. a tree) based on the identified headings and subheadings. Here, a structure-based learning (i.e. tree-based learning) approach is needed rather than the simpler case of classification. The classification or clustering task aims at determining the correct

class of a given instance with respect to a learned model or a measure reflecting particular relationships (e.g. similarity) between instances. In the case of document structural analysis, on the other hand, we do not possess a predefined set of classes and we form the structure of a given instance (document) based on a schema such as a model, heuristics or grammar rules (Mao, Rosenfeld and Kanungo 2003). In this respect, the process of document analysis necessitates the use of an information extraction approach that takes the relationships and constraints between different types of elements in a document into account.

The main difficulty in developing a tree-based learning approach is the exponential search space that occurs during the process due to the nature of the problem (Hastie, Tibshirani and Friedman 2009; Mandhani and Meila 2009). The branching factor and the number of levels in a tree representing a possible solution of the problem are large. This is the case for the problems related to natural language processing, such as syntactic parsing, segmentation, discourse analysis, or document processing. The common characteristic of these problems is the existence of a large number of entities to deal with and several alternative interpretations for each entity, causing the number of combinations to increase rapidly. This property prevents one to consider all possible solutions in order to retrieve the best one. To alleviate this problem, in this paper, we adopt an incremental learning approach where the subtrees are generated in consecutive steps using support vector machines (SVM) and perceptron algorithms. The basic idea underlying the approach is making a sequence of locally optimal choices in order to approximate a globally optimal solution. To the best of our knowledge, this is a novel approach in the structural analysis of Web documents.

The system built was evaluated using two alternative strategies. First, the document structure analysis methods were evaluated with respect to the accuracy of the heading extraction and hierarchy extraction models. Second, a task-based evaluation was performed in Web search context where the task is structure-based and query-biased summarization. In this approach, we analyzed the effect of structural information in automatic summarization using English document collections and queries.

The rest of the paper is organized as follows: Section 2 gives a summary of related work. The outline of the system architecture is shown in Section 3. Section 4 discusses in detail the structural processing models built in this work. Section 5 explains the summarization method. The evaluation of the system and the experimental results are given in Section 6. Section 7 concludes the paper.

## 2 Related work

Structural analysis of a document is the process of making the logical structure underlying the document explicit by means of some automated methods. As the structure within the document is extracted, it can be represented in the form of a structured document based on a schema. The main motivation behind this process is facilitating the use of the information contained in the documents since structured infor1)mation can be interpreted much more easily by computer applications. Based on this observation, there has been a great deal of studies for extracting the structural representation underlying documents written as flat texts.

The analysis of document structure and extraction of structural and semantic information in the documents is becoming increasingly important especially for Web documents due to the wide spread use of the World Wide Web. Web documents are usually encoded in HTML format (HTML 4.01 Specification 1999) and can contain rich structural information. However, since HTML is concerned with the presentation of content, it does not always correspond to the semantics of the data. As a result, Web documents are considered as 'semi-structured' documents (Chaudhuri 2006). The practical applications of document analysis include the display of content on small-screen devices such as personal digital assistants (PDA), summarization of content, more intelligent retrieval of information, etc. Most of the studies that target analysis of HTML documents make use of the document object model (DOM) tree representation since it provides a global view of the document structure (Jensen, Madsen and Moller 2011). DOM is a platform- and language-independent interface and allows programs to dynamically access and update the content, structure and style of documents. In the DOM paradigm, documents are represented as a hierarchy of nodes, which is referred to as the DOM tree (Document Object Model 2012).

One approach used for document structural analysis is considering it as a syntactic analysis problem (Mao *et al.* 2003). In this approach, the order and containment relations between physical components (paragraphs, words, figures, etc.) and logical components (titles, authors, sections, etc.) of a document are represented in the form of an ordered tree. This is analogous to the analysis of a sentence which can be described as a tree with grammatical relationships among its words. In this respect, some syntactic analysis methods used in natural language processing are adapted to the document analysis problem.

In one of the studies, the transformation-based learning method that was applied previously to syntactic parsing is used for the conversion of HTML documents into extensible markup language (XML) format (Curran and Wong 1999). The authors use a large number of transformation templates and an evaluation function that is specific to the error types in XML tagging. Although the structure of the system and the details of the system components were explained, no evaluation results were given. In another study, the logical structure of a document is modeled via a structural representation of patterns formed of entities in the document (Brugger, Zramdini and Ingold 1997). The document structure is constructed in a hierarchical way where local tree node patterns are defined similar to n-grams. The originality of the work stems from using the n-gram concept to model nodes in a tree in addition to modeling sequential entities. Branavan, Deshpande and Barzilay (2007) propose a method that automatically generates a table-of-contents structure for long documents such as books. They first segment a document hierarchically and then generate an informative title for each segment. A model is learnt using the incremental parsing approach of Collins and Roark (2004), which is also the approach we use for extracting hierarchical structures of documents in this work. In this approach, the parse tree is incrementally constructed by applying simple corrective updates to the parameters during training and beam search is incorporated to reduce the size of the exponential search space of possible parses.

Similar to approaching the problem of document structure analysis as a syntactic analysis problem, some works aim at modeling the document structure explicitly by grammar rules and parsing a given document with respect to the grammar built. In an early study that targets extracting logical structures of documents from their physical representations, a document given in image form is first converted into text form (Niyogi and Srihari 1995). The structure of documents is modeled by a set of context-free grammar rules formed for a specific domain. The documents are parsed according to the defined grammar and the hierarchical structures are obtained. In another study, it is argued that different types of document analysis tasks can be solved using discriminative grammar formalisms (Shilman, Liang and Viola 2005). In the case of document structure analysis, document layout is modeled by a grammar based on the assumption that a document is a sequence of pages where a page is formed of paragraphs, a paragraph is formed of lines, and so on. A grammar learning algorithm is trained on a set of training examples and the optimal parse for a given document is obtained using the learned model. The experiments on a small corpus showed about 85 per cent success rates (f-measure).

The grammar-based approaches used in document layout analysis have the advantages that the grammar rules are easy to interpret since they reflect the logical segmentation of documents into entities and there exist plenty of sound mechanisms for parsing. However, it is quite difficult to build a grammar in advance that can sufficiently represent the hierarchical relationships between entities in a document, except in very formal domains such as academic papers. This is especially the case for Web documents due to their cluttered structures and irregular forms (Gupta *et al.* 2003). In addition to the actual content, Web documents include several entities such as advertisements, unrelated images, links scattered around the screen, and navigation menus, which makes modeling such pages with a grammar difficult. In grammar-based approaches, this problem is handled by identifying a domain (e.g. newspaper stories) and then building a grammar specific to that domain.

Some studies in the document analysis field approach the problem from a content extraction perspective rather than (or, in addition to) structural analysis. In one of these studies, Rahman, Alam and Hartono (2001) propose a rule-based approach that extracts the contents of a Web document and produces a summary to be used in PDA devices. The methodology used in the work follows the stages of analyzing the structure of a document, decomposing the document into zones, analyzing and summarizing each zone, and reordering the summary sentences based on their importance. Gupta, Kaiser and Stolfo (2005) address the problem of domain dependency that degrades the performance in document analysis studies. They split the content extraction problem into two stages. First the genre of a Web page is determined using search engine snippets and then its content is grouped into related parts using a predefined setting for the identified genre. The major drawback of this strategy is that it requires deciding on a set of genres in advance and building a schema for each genre. In another study, the authors worked on eliminating the non-content in the documents (i.e. cluttered parts around the body of a document that distract the user from actual content) rather than identifying the important content directly (Gupta *et al.* 2003). For this purpose, they use a set of filters and

the document that remains after the non-content parts are eliminated is given as the output. Although the cluttered parts in the documents are removed by this approach, it fails in distinguishing important and unimportant parts in a document and outputs all the remaining content.

There are some works that limit document analysis to the extraction of important entities from documents. Xue *et al.* (2007) use support vector machines and conditional random fields (CRF) to extract the main title from the content of a Web document using DOM tree and vision-based features. The authors address HTML documents and form a specification of a title in these documents based on formatting features. The method was evaluated on a document retrieval setting and was compared with a simple baseline method. In their work, Le and Thoma (2003) considered the problem of assigning predefined labels (title, author names, affiliation, and abstract) to scientific documents. After the DOM tree representation of a document is built, the unnecessary nodes in the tree are eliminated, a set of string patterns are generated, and the document parts are labelled using a pattern matching algorithm. The patterns are formed using both position-based and content-based features. As the research is restricted to formal documents and clearly-written parts in these documents, success rates up to 95 per cent were achieved. A work that uses CRF for document analysis was given by Pinto *et al.* (2003). In this work, the problem is restricted to the extraction of tables in the documents. The authors identified a rich set of features and compared the performance of the model with the hidden Markov model (HMM) and the maximum entropy (MaxEnt) approach. The f-measure results obtained were about 65 per cent, 85 per cent and 93 per cent for HMM, MaxEnt and CRF, respectively.

These works have a limited scope in the sense that they focus on specific document entities and develop algorithms specialized for this purpose. They do not target identifying the relationships between elements in the documents. A work addresses a more general problem, which is the problem of finding the headings in a document together with the underlying hierarchy (Pembe and Güngör 2009). The authors use a rule-based approach based on a number of heuristics and DOM tree processing. The proposed methodology was evaluated on a search engine setting and it was shown that the use of structural information in extracts output by search engines improves the effectiveness of the search.

Some researchers consider the document analysis problem as a semantical analysis problem and use clustering or machine learning techniques to identify related parts in documents. Liu, Wang and Wang (2006) aim at segmenting a Web page into a set of semantically related blocks based on the idea that text portions inside a particular segment usually share some common contents and presentation styles. They propose a vision-based page segmentation algorithm that first converts an HTML DOM tree into a semantic tree and then applies a Naive Bayes classifier using a set of content and spatial features. In another work, it is assumed that subtrees that have a similar structure in the DOM tree representation of a document belong to the same group (cluster) (Mukherjee *et al.* 2003). This is referred to as a schema, which is defined as representing the concepts and relationships among the document parts in a hierarchical fashion. The authors propose a partitioning algorithm to divide a given

Web page into a set of schemas. Kao, Ho and Chen (2004) split a Web document into information blocks of different types by means of a form of clustering. The work basically targets more efficient display on PDAs and proposes that hierarchical search on such devices is easier than recursive search. The authors use a three step process based on the information space tree concept and obtain about 80–90 per cent f-measure scores on different datasets. Yang and Zhang (2001) view the content of a Web document as formed of objects and employ a classification algorithm to identify different types of objects. They base the classification on a number of similarity measures used for objects of the same type. Given a document, the parts that have similar contents are extracted using a pattern detection algorithm. Based on the identified objects, document hierarchy is built recursively. The approach basically focuses on the visual appearance of a Web page by means of the so-called content objects and does not take into account the actual content within these objects. Feng, Haffner and Gilbert (2005) use two learning approaches in order to segment a given Web document into blocks and to identify the semantic categories of the blocks. They define 12 category types that occur frequently in Web pages and make use of different types of features such as visual clues and linguistic phrases. The work aims at obtaining a flat sequence of segments without considering the hierarchical structure in the document.

One of the practical applications of Web document analysis that has drawn attention of researchers is the display of content on small-screen devices such as PDAs. Some of the studies in the literature concentrate on this goal and develop algorithms tailored for this particular area. In a study, a browsing method, named as accordion summarization, for displaying the contents of Web pages on PDAs was proposed (Buyukkokten, Garcia-Molina and Paepcke 2001; Buyukkokten *et al.* 2002). Three different summarization techniques are applied, which are page summarization, keyword-driven summarization and automated view transitions. In each, some heuristics are used related to the contents of the documents and a hierarchical summary is formed. Another work that targets similar devices extracts the hierarchy of a Web page by exploiting the observation that the variety of information content decreases as we move towards lower parts of the tree (Chen, Ma and Zhang 2003). A rule-based method is applied by using heuristics for different types of page elements (header, footer, separators, etc.) and the content is displayed using a page adaptation mechanism that splits the page to fit on the small screen of a device. Xiao *et al.* (2009) attempt to partition the content of a Web page into a set of subpages, each of which fits on the screen. A tree formed of page blocks is built by taking into account some factors such as the size and number of blocks and semantic coherence between blocks. The internal nodes of the tree serve as index pages that allow access to the actual contents of the Web page.

The works on document analysis can be classified with respect to a number of criteria such as the aim of the analysis (extracting entities, a flat structure, or a hierarchical structure), the scope of the analysis (all entities, titles, tables, etc.), the approach used (rule-based, machine learning, clustering, grammar-based, etc.), being domain-independent or not, and the targeted environment (computer, PDA, mobile phone, etc.). The difference of the current work from previous studies

lies in the methodology used and in expanding the coverage of those studies by building the structural representation of the whole document content. We attempt to obtain the hierarchical structures of documents using a two-step process by first identifying the headings that will aid in determining the structure and then forming a sectional hierarchy based on these headings and also the non-heading content. The work addresses the whole document content, is domain-independent, and employs a learning paradigm. In addition, it builds a summarization framework that makes use of the results of structural analysis and forms a structural summarization component that is useful in many application areas. In these respects, the current work can be regarded as proposing a new way for solving the document analysis problem. As will be shown in Section 6 and discussed in Section 7, the methods built in this work yield high success rates in document structure analysis and structural summarization and outperform the approaches used in similar works.

## 3 System architecture

The components of the system built in this work are shown in Figure 1. First, the documents collected from the Web are processed in order to obtain their DOM tree representations. In this work, we compiled the document collection using a set of TREC (Text Retrieval Conference 2010) queries by submitting each query to a search engine and retrieving the results output by the search engine (Section 6.1). The DOM tree representations were obtained using the Cobra HTML Renderer and Parser open source toolkit (Cobra: Java HTML Renderer and Parser 2010). Although the DOM tree of a document can be regarded as a form of hierarchical representation, it is too complicated and does not reveal the actual content of a Web page. This is due to the fact that DOM tree only aims at making the relationships between document entities determined by HTML tags explicit. However, as mentioned previously, these tags are used to get a desired visual effect and do not reflect the actual layout of a document. Thus it is necessary to process further the DOM tree to obtain the structure of the document.

In the methodology used, we accept a text unit as the basic element that is used by the learning algorithms. A text unit corresponds to a text fragment (e.g. sentence, paragraph, heading) that is assumed to be coherent in itself. In our preliminary experiments, we observed that dividing a document into such units instead of sentences yields more meaningful results. After the DOM tree of a document is built, the text units within the document are identified (Section 4) and the values of the features for each text unit are computed. The work makes use of a rich set of features formed of different types of features to encode both the properties of individual units and the relationships between consecutive units (Section 4.3).

The output of the feature extraction component which consists of the text units in the documents accompanied with feature values is fed into the heading extraction module (Section 4.1). In the training phase, the heading extraction module learns a model that discriminates headings and non-headings. In the test phase, given

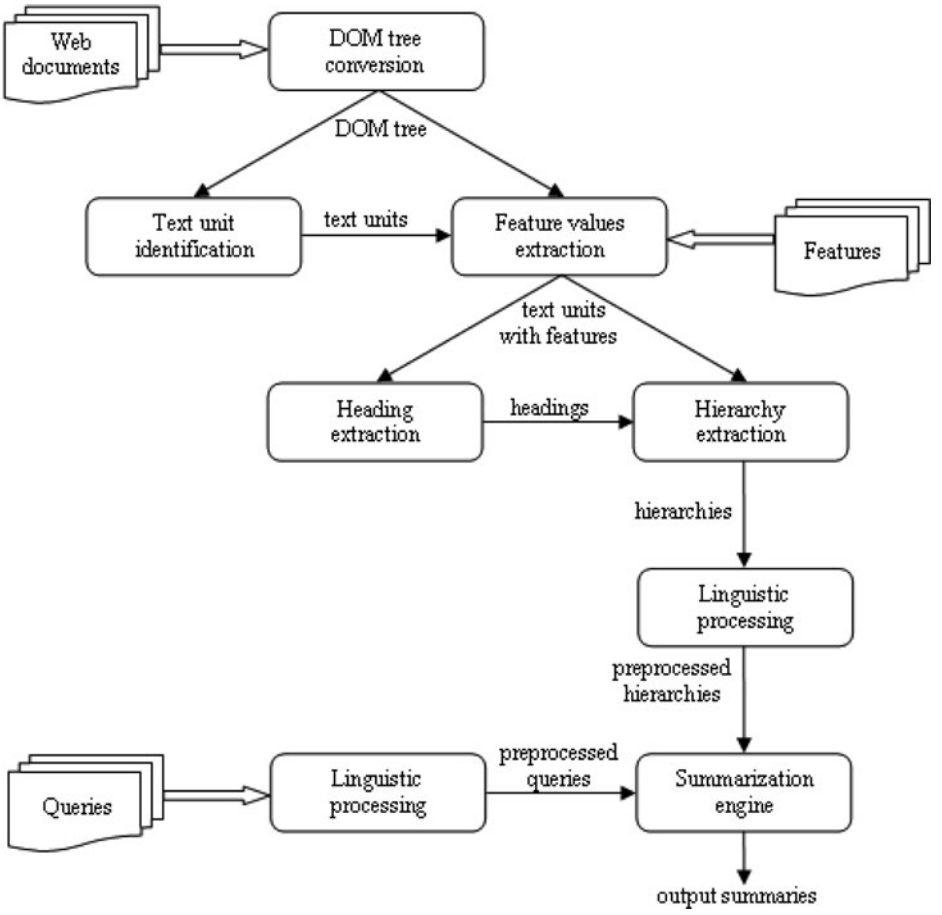


Fig. 1. System architecture.

a document, this model is used and a binary classification is performed in order to identify the headings in the document. This process is followed by hierarchy extraction in which the hierarchical relationships between the heading units are determined and the hierarchical structure of the document is built using the heading and non-heading units (Section 4.2). As will be detailed later, an incremental approach is employed during hierarchy construction and a number of alternative variations of the main algorithm are tested in order to determine the best method (Section 4.4).

The last step in the system architecture is testing the usefulness of the methodology in a search engine setting. For this purpose, a given user query and hierarchical representations of documents are first subjected to simple linguistic preprocessing operations such as stemming, case folding, and stop words elimination. Then the documents are summarized by taking into account the query terms and structural summaries are obtained (Section 5). Finally, the performance of the summarization component is evaluated.

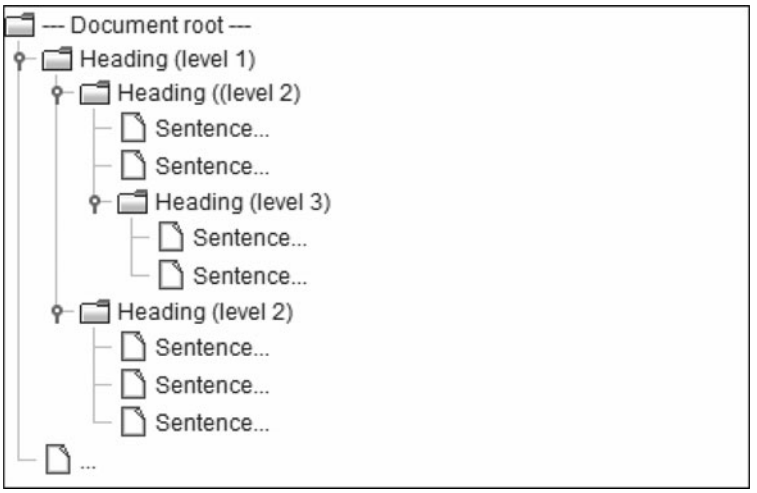


Fig. 2. Output of structural processing.

## 4 Structural processing of Web documents

In general, the structure of a document can be considered as a hierarchy where the document is formed of sections, each section has subsections, and so on, together with the corresponding headings and subheadings. The format of the documents used as input in the system built in this work is HTML, because it is still the most frequently used format on the Web. The output of structural processing is a tree representing the sectional hierarchy of a document where headings and subheadings are at the intermediate nodes and other text units are at the leaves. An example hierarchical structure is shown in Figure 2. The root contains a dummy unit covering the whole document. As can be seen, headings at different levels form a hierarchy together with the sentences under the headings.

In the current system, a Web document is modeled as a sequence of text units based on their orders in the HTML source of the document. When we process the HTML code of a document in a top-down manner, we obtain a unique ordering of different types of elements (text, images, side bars, etc.) in the document. The analyses in later steps are based on this ordering of page contents. The modeling approach used here 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 character (i.e. paragraph delimiter) as illustrated by rectangles in Figure 3. Two machine learning models were developed: heading extraction model and hierarchy extraction model. The models are detailed in the following subsections together with the features used, learning and testing approaches, and implementation.

### *4.1 Heading extraction model*

Although there exist a number of heading tags in the HTML format to identify the headings in a page, they are either rarely used by web page designers due to the restrictions they impose or sometimes used to obtain visual effects on plain texts.

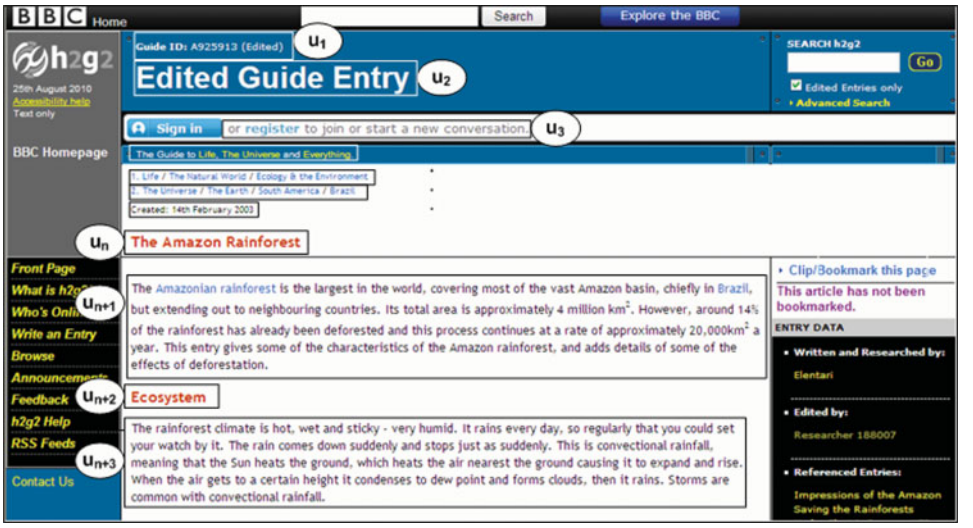


Fig. 3. (Colour online) Part of an example HTML document with units identified.

Hence, rather than relying on these tags, we attempt to identify the headings in Web documents by making use of the features of text units. In this respect, all text units in a document are treated uniformly regardless of being marked as heading or not in the underlying HTML source. The heading tags are included as separate features in the feature set. This increases the chance of a text unit marked as a heading in the HTML format as being identified as a heading by the algorithm, provided that the heading tags were used properly while authoring the Web page.

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 $i = 1,\ldots,n$, where $u_i$ corresponds to a text unit and $y_i$ to its label, and $n$ is the number of units in the document. The label denotes whether the text unit is a heading or not. A text unit $u_i$ is represented as a sequence of features $x_{ij}$, $j = 1,\ldots,k$, where $k$ corresponds to the number of features used by the model. Here, the task is to learn the classification model distinguishing positive instances (headings) from negative instances (non-headings).

### *4.2 Hierarchy extraction model*

In hierarchy extraction, the problem of learning a mapping from a set of documents $X$ to a set of possible sectional hierarchies $Y$ is considered. This is analogous to syntactic parsing in which $X$ corresponds to a set of sentences and $Y$ to a set of possible parse trees (Collins and Roark 2004). We formulate the problem as follows:

- Training examples $(x_i, y_i) \in X \times Y$ for $i = 1,\ldots,t$, where $t$ is the size of the training set
- A function $GEN(x)$ which enumerates a set of possible outputs for an input $x$
- A representation $\Phi(x_i, y_i)$ mapping each $(x_i, y_i)$ to a feature vector
- A parameter vector $\alpha$

The training set includes $(x_i, y_i)$ pairs where $x_i$ is a Web document and $y_i$ is the gold standard tree corresponding to the document sectional hierarchy. The learning task is to estimate the parameter vector $\alpha$ using the training examples as evidence. As the parameter vector is obtained, Eqn. (1) is used to choose the most likely hierarchical structure ($y_{max}$) for a given test document $x$. That is, at each step of the incremental learning process (Section 4.4), a number of alternative partial trees are evaluated with respect to the parameter vector and the feature set, and the one which maximizes Eqn. (1) is identified as the correct hierarchy.

$$y_{\max} = \arg \max_{y \in GEN(x)} \Phi(x, y) \cdot \alpha \tag{1}$$

In general, the main difficulty in developing a tree-based (document trees in document processing, parse trees in syntactic parsing, etc.) learning approach is the exponential search space encountered during the process. 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. This approach has previously been successfully applied to other tasks in the literature, including syntactic parsing and generating table-of-contents for general documents (Collins and Roark 2004; Branavan *et al.* 2007). In this method, 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.

### *4.3 Features*

We define different types of features corresponding to different levels in a document. The first type consists of a set of features related to the properties of the smallest item (i.e. text unit) in a document. This is followed by features based on the context of a unit that indicate the relationships between neighboring units. Finally, global features are defined considering the document as a whole.

#### *4.3.1 Unit features*

A text unit is considered as the smallest item in the system and is associated with a set of features. Text units can be automatically detected using certain tags that specify paragraphs in the HTML format, such as <*br*> and <*p*>. A text unit may correspond either to a single node in the HTML DOM tree or to more than one node if different parts of the unit are enclosed within different HTML tags. For instance, a paragraph (text unit) may contain one or more parts written in bold (enclosed inside <*b*> tags). In this case, the paragraph will be divided into parts as the boldness changes and will be represented as a sequence of nodes in the DOM tree.

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 the CSS information. During

Table 1. *Formatting features for 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 |

parsing, the parts of a text unit divided into several nodes in the DOM tree are combined to form a single unit. After the parsing process, text units in the document are associated with features. The unit features include formatting features, DOM tree features, content features, and other types of features.

*Formatting features* are based on HTML tags and attributes used for formatting the text units, such as font size, boldness, color, etc. (Table 1). The formatting information is obtained after CSS information is incorporated.

*DOM tree features* are related to the DOM tree parse of the document. Although the DOM tree is concerned with the presentation of the contents, it can also contain valuable information about the structural organization of the document. In HTML documents, the organization is achieved by using nested tables (<*table*>) and divisions (<*div*>). The DOM path and the DOM address (Feng *et al.* 2005) of a unit are used as features in this work. The DOM path of a unit refers to the sequence of node labels from the root node (<*html*>) to the node that is the parent of the leaf node that corresponds to the unit (e.g. '*html.body.div.table.tr.td.b*'). The DOM address of a unit is similar in the sense that it also represents the path from the root to the unit leaf, but the node labels are replaced with numbers. The children of each node in the tree are numbered consecutively starting from zero. Then the DOM address is formed as a sequence of the numbers on the path (e.g. '*0.1.0.2.1.3.0*'). Another feature used is the position of a text unit within the innermost table or division according to the DOM tree. This information is especially useful in heading extraction because headings are often found at the first position within a table or division.

*Content features* are related to the textual content of a unit, such as features that specify whether a unit contains certain cue words or phrases ('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 (if any). Such features are especially important in heading extraction; usually, headings are limited in length, consist of a single sentence, and contain no punctuation marks at the end.

Several other features are 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). For this purpose, the $x$ and y coordinates of the text units are computed using the Cobra HTML Renderer and Parser. Other features include a feature designating whether a unit is the document root or not and a feature indicating the use of a horizontal line (<*hr*> tag in HTML) to separate content.

### *4.3.2 Contextual features*

We define a contextual feature of a unit as a feature that encodes a relationship (in terms of unit features) between this unit and another unit in its context. For heading extraction, we limit the context of a unit to the two preceding and the two succeeding units in the document. Given a unit $u_i$, we use the notation $F_{ij}$ ($j = i$-2, $i$-1, $i$+1, $i$+2) to denote the set of features related to $u_i$ and $u_j$. For hierarchy extraction, the document tree built during the incremental learning process (see Section 4.4) is considered: the context of a unit is the set of all units in the tree that the unit can attach to. For a unit $u$, $u_{ij}$ denotes the unit $i$ levels above it and $j$ units to its left in the partial tree constructed so far. For instance, given a unit $u$, $u_{10}$ represents its parent unit, $u_{01}$ its preceding sibling unit, and $u_{20}$ its grandparent unit.

Contextual features utilize the difference and distance between two units in the same context. This can provide useful information in determining the headings and the sectional hierarchy in a document. Intuitively, a heading unit (i.e. parent unit) is more emphasized than the underlying text units or subheadings in terms of formatting. Similarly, text units under the same heading, i.e. sibling units, generally have similar formatting features.

Contextual features of two units are defined in terms of the formatting, DOM tree, and visual unit features. For a binary (boolean) or integer formatting feature, the corresponding contextual feature is the difference of the values of the units' features. For instance, the contextual value of the *h1* feature for two units $u_i$ and $u_j$ is $u_i$.*h1*- $u_j$.*h1* (-1, 0 or 1). As another example, if $u_i$ has a font size of 12 and $u_j$ has a font size of 14, then the contextual font size feature will be -2.

We define a contextual DOM address feature for two units as the length of the path common to their DOM addresses starting from the root. The idea is that semantically related parts in a document show spatial locality in the DOM tree and have similar DOM addresses. This value is normalized by dividing it to the depth of the overall DOM tree for that document in order to smooth the depth differences between documents. We define contextual features related to the visual positions ($x$ and $y$ coordinates) of two units. The differences of $x$ coordinates and $y$ coordinates

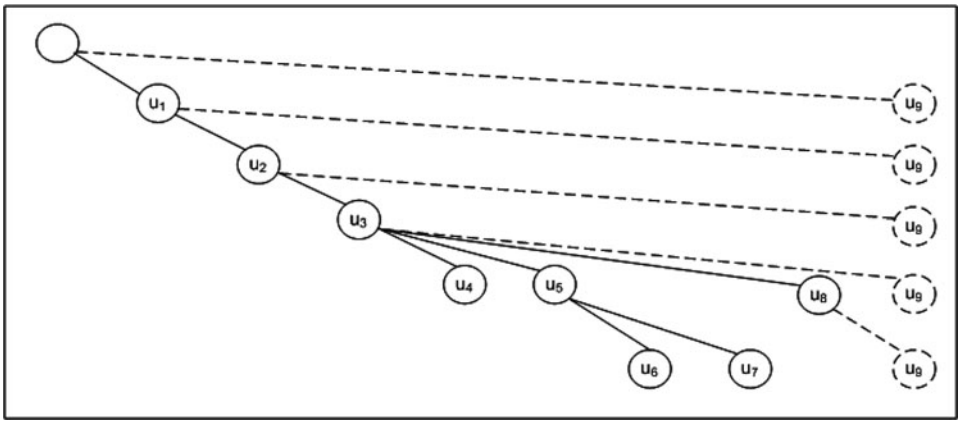


Fig. 4. Part of an example document tree (units $u_1$-$u_8$). The dashed lines indicate potential attachments of a unit $u_9$.

are considered as positive, negative, or zero. For instance, if the difference of the $y$ coordinates between two units $u_i$ and $u_j$ is positive, this means that $u_i$ comes below $u_j$ in the visual display of the Web document; therefore, $u_i$ cannot be a heading of $u_j$. The difference of the $x$ and $y$ coordinates can also be compared to a threshold value. As an example, if the visual $y$ position difference between two consecutive units $u_i$ and $u_j$ is very large, then it is quite unlikely that $u_j$ is a heading of $u_i$.

#### *4.3.3 Global features*

In addition to the features of a single unit and the contextual features, we also define global features by considering the document or the document sectional hierarchy as a whole. One such feature is the depth of the tree built so far during the learning process. These features are used in the hierarchy extraction stage.

### *4.4 Incremental learning approach*

In document sectional hierarchy extraction, we use an incremental approach in the machine learning model. Figure 4 (units $u_1$-$u_8$) shows an example partial hierarchy for a Web document. (The unit $u_9$ and the dashed lines in the figure will be explained later.) 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 unit indices are arranged according to their order of appearance in the document. The solid lines in the figure indicate the dependency relations between the node pairs, i.e. the parent-child relationships (*heading-underlying text* or *heading-subheading*). These correspond to positive examples for the learning process. The negative examples (not shown in the figure) are the potential dependency relations which are not realized in the gold standard hierarchy.

The main training algorithm is given in Figure 5. The input to the algorithm is the training set consisting of Web documents and corresponding gold standard hierarchies. For each document in the training set, the algorithm works on the units

```
Module 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
```

Fig. 5. Training algorithm for hierarchy extraction.

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 extracting the negative examples, two constraints due to the document flow are applied. First, a unit cannot be attached to a unit coming after it in the document. Second, the connections in the tree cannot cross each other according to the projectivity rule, as in dependency parsing (Covington 2001):

*Projectivity Rule: When searching for the parent of a unit $u_i$, consider only the previous unit $u_{i-1}$, the parent of $u_{i-1}$, the parent of the parent of $u_{i-1}$, and so on until the root of the tree.*

The projectivity rule states that if a unit $u_{i-k}$ is the parent of $u_i$, then all the units between $u_{i-k}$ and $u_i$ must also be descendants of $u_{i-k}$ in the hierarchy. The projectivity rule is implemented in lines 5–11 in Figure 5. For instance, in the example hierarchy, the connection $u_4$-$u_6$ violates the projectivity rule and hence does not count as a negative example.

In the test phase, the hierarchy for a previously unseen document is built incrementally using beam search. The testing algorithm (Figure 6) operates on each text unit sequentially based on the order in the document and maintains a set of partial solutions (*PartialTrees*). We adapt two operations similar to the previous work on incremental parsing (Collins and Roark 2004; Branavan *et al.* 2007): ADV (advance) and FILTER.

Whenever a unit in the document is processed, its potential attachments to the existing set of partial trees are considered and the set is updated to include new partial trees (ADV operation). In Figure 4, potential attachments of a unit ($u_9$) to an example partial tree are shown with dashed lines. During this process, the two constraints due to the document flow (order of units and projection principle) are also applied. To prevent the exponential growth of the set of partial trees, the

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

Fig. 6. Testing algorithm for hierarchy extraction.

FILTER operation is introduced. Using the output of the machine learning model, only the top $k$ (beam width) highest scored partially generated trees are maintained at each step.

In this process, we also utilize the output of the heading extraction model. We apply a preprocessing step to each document that enforces two constraints. First, if a unit $u_i$ is a heading, then the unit $u_{i+1}$ is automatically attached to $u_i$ indicating that a heading must have at least one text unit (heading or non-heading) as the underlying content. Second, a unit is allowed to attach only to a heading unit, since a non-heading unit cannot serve as the parent of a text unit.

We also developed several variants of the main testing algorithm and analyzed their effects on the accuracy of sectional hierarchy extraction. The main algorithm uses a greedy approach in the sense that at each step only the partial trees with maximum scores are considered and the rest are eliminated. The disadvantage of this approach is that it is possible for a partial tree that gets high scores in most of the steps can be eliminated when it gets a low score in a step. In the modifications 1, 3 and 4 explained below, on the other hand, we include some knowledge from the previous steps in determining the best partial trees. The first variation takes into account the probabilities from the root to the current node in the partial trees. The third and fourth variations consider also the score values of a partial tree (in terms of ranks) in the previous steps. As another type of modification, the second

variation analyzes the effect of using the hierarchy of headings in building the overall hierarchical structure. The details of the modifications are given below:

- *Modification 1:* Instead of using the score obtained from the machine learning model directly, it is possible to convert it into a probability value using a sigmoid function as shown below (Platt 1999; Mayfield *et al.* 2003):

$$f(x) = 1/(1 + \exp(Ax + B)) \tag{2}$$

In Eqn. (2), the function parameters $A$ and $B$ can be estimated using an iterative method (Platt 1999). Alternatively, fixed parameters can be used (Mayfield *et al.* 2003), which is also the approach we take here with $A = -2$ and $B = 0$. We calculate the probability of building a partial tree formed of $n$ units by using the multiplication rule of probabilities as shown in Eqn. (3):

$$\prod_{i=1}^{n} P(parent(u_i) = u_j) \tag{3}$$

where *parent*($u_i$) denotes the parent unit of $u_i$. We make the simplifying assumption that the attachments in the hierarchy are mutually independent of each other.

- *Modification 2:* The testing algorithm is run in two levels. In the first level, the algorithm is applied to heading units only, i.e. heading units are connected in order to obtain the overall heading hierarchy. Note that taking into account only the heading units in this step does not cause any change to the algorithm in Figure 6. The tree obtained will be similar to a tree that would be obtained when all the units are considered, except that the leaf nodes (non-heading units) will be missing. In the second level, non-heading units are attached to the correct positions in the hierarchy using the output of the first level. This is accomplished by executing the same testing algorithm under the constraint that some of the units (headings) have already been connected (line 6 in Figure 6).
- *Modification 3:* Instead of using the score output by the machine learning model directly, the partial trees are assigned integer ranks starting from '1' which is given to the best scored tree. During the filtering process, the number of times a partial tree has obtained rank '1' in previous steps is calculated to obtain its score. The trees with higher scores are favored. In this way, the success of a partial tree at each step is taken into account, rather than considering its success at the current step only.
- *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. The intuition behind this modification is the same in the sense that previous success rates of the partial trees are also considered.

### *4.5 Implementation*

The models defined for heading and hierarchy extraction have been implemented as a standalone application in Java using two different machine learning algorithms:

support vector machines and perceptron. In the heading extraction phase, document units are represented in terms of features and a binary classification (heading or non-heading) is performed. In the hierarchy extraction phase, positive and negative examples are created, the partial trees are represented with features, and the incremental learning process is applied.

Support vector machine is a machine learning method commonly used in the classification domain. In the current system, we utilized the SVM-light implementation (Joachims 1999). In addition to the linear kernel, we also experimented with nonlinear kernel functions including polynomial and radial basis function (RBF) kernels (Joachims 2002; Alpaydın 2004). Perceptron is a type of artificial neural network that performs classification by mapping an input $x$ to an output $y$ based on a weighted sum (Alpaydın 2004).

## 5 Summary extraction

We used a structure-based and query-biased summarization method which was shown to be effective in Web search tasks (Pembe and Güngör 2009). It is an extractive method consisting of two levels of scoring: sentence scoring and section scoring. In the first level, following a preprocessing stage (stemming, stop words elimination, etc.), sentences in a document are scored using statistical summarization methods by taking into account the output of the structural processing step. The methods used in this work are heading, location, term frequency, and query methods. The score of a sentence is calculated as the weighted sum of the normalized scores obtained in each method.

- *Heading method:* Headings in a document include terms that give a general idea about the document contents and can be used for automatic summarization (Mani 2001; Yang and Wang 2008). Following this observation, using the output of the heading identification process, we give a heading score to sentences containing heading words based on the number of such words they contain. We consider all the headings in a similar manner without taking into account their levels in the hierarchy. We leave the process of differentiating between headings at different levels and using this information in the heading score as a future work.
- *Location method:* Sentences located at certain positions in a document may convey important information (Mani 2001). Based on the output of the hierarchy extraction step, we give a positive score to sentences located at the beginning of a section or subsection.
- *Term frequency method:* Sentences that include terms that occur frequently in a document may be assumed to focus on the actual content of the document (Mani 2001). There are several variations of the term frequency method in the summarization literature (Baeza-Yates and Ribeiro-Neto 1999; White, Jose and Ruthven 2003; Pembe and Güngör 2009). In this work, we adapt a simple form and give a term frequency score to sentences based on the frequencies of the words they contain.

- *Query method:* In the information retrieval context, using query-biased summaries is more effective than using generic summaries as produced by the systems in other domains (Tombros and Sanderson 1998; White *et al.* 2003). For instance, a search engine should extract the fragments in a document that the user is interested in, rather than giving an overview of the document even if it reflects the content better. Based on this observation, in the current system, we give a query score to sentences containing query words based on the number of query terms they contain.

The second level of the process deals with the scoring of the sections and subsections that were identified during structural processing. The score of a section is calculated by summing the scores of the sentences it contains. Sections are represented with different number of sentences in the summary proportional to their scores. In a previous study, it has been shown that summaries longer than those of the traditional search engines improve the search experience of Web users (White *et al.* 2003). Such summaries were shown to be more effective than the summaries of Google and AltaVista on a task-based evaluation. Based on this idea, we define a quota for the summary of a document (e.g. 25 sentences) and this quota is shared hierarchically among the sections according to their scores. In the output summaries, the hierarchical structure of the document is also preserved and displayed explicitly.

The structure-based summarization approach that we employ allows us to incorporate structural information about Web pages into the summarization process in two ways. The first one is using a modified form of the heading and location methods in determining the saliency of sentences by taking into account the extracted headings and hierarchies. The second one is treating each section and subsection separately, rather than considering the document as a whole, which enables more important sections to be represented more heavily in the summary. In this respect, the document structure is provided to have an important role during summarization and to be represented explicitly in the summary.

## 6 Experiments and results

### *6.1 Data collection*

To evaluate the system, we need a sufficiently large and representative corpus of Web documents. Such a corpus can be compiled from the queries employed by users in real search tasks. Studies on users' behavior related to query formation in search engines show that users usually prefer boolean queries consisting of only a few words (Ingwersen and Jarvelin 2005; Markey 2007). We followed the same strategy of using short queries in this work and compiled a set of TREC queries (from TREC 2004 Robust Track, topics 301–450) by taking into account the search interests of users in various domains (see Table 2). To collect the documents used in the experiments, each query was submitted to the Google search engine and the top 25 results (HTML documents) for each were taken. In this way, the corpus was formed of 20 TREC queries and a total of 500 documents. The corpus represents a diverse and realistic set of documents in terms of both structure and content due to

Table 2. *Queries used for building the corpus*

| 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 |

the use of queries corresponding to different search interests and the use of preferred (top) results retrieved by the search engine. The documents in the corpus also show variations with respect to the document length and the markup tags used.

The documents in the corpus contain an average number of 110.7 text units and 1,340 words. The headings in the documents were manually annotated using a specification guide (Figure 7) formed by the authors that shows the general guidelines for identification of headings. During the annotation process, we observed that the annotators can decide on whether a text unit is a heading or not more easily for units marked explicitly as headings using the heading tags. Document sectional hierarchies were also manually marked based on the identified headings and the document organization. We used two annotators for the tagging of the corpus. The documents were divided into two groups randomly and each document was tagged by one annotator. In order to observe the agreement between the annotators, we performed an additional test using a subset of the corpus (100 documents). Each document was marked by the two annotators. The agreement between the annotators was measured as 70 per cent. The annotation process resulted that a document contains 10.6 headings and has a hierarchy depth of 4.1 on the average.

### *6.2 Structural processing experiments*

Following the proposed two-level approach, the structural processing phase was evaluated in two steps: heading extraction and hierarchy extraction. The outputs of

**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 can not 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.

Fig. 7. Specification guide used for manual heading annotation.

the methods were compared with the gold standard headings and document hierarchies determined manually for each document. The machine learning experiments were performed using five-fold cross-validation.

We used the rule-based approach developed by Pembe and Güngör (2009) as a baseline. Using the heading tags in HTML or the DOM tree structure does not give rise to a proper comparison and cannot serve as a baseline, since they result in low success rates. As mentioned before, the heading tags in the HTML source of a document may not correspond to the headings in the document. The DOM tree structure, although it gives information about the layout in a document, does not represent the actual hierarchy (headings and sections) of the document. The DOM tree presents a complex view of the document and includes many irrelevant nodes. In addition, only the leaf nodes contain textual data in the DOM tree, whereas all the nodes in the trees built in the current system are formed of textual content.

In order to evaluate the accuracy of heading extraction, we adapted the recall, precision, and f-measure metrics that are widely used in information retrieval (Baeza-Yates and Ribeiro-Neto 1999). For each document, four different values are computed: *TP* (true positive – the number of heading units identified correctly), *FP* (false positive – the number of non-heading units identified as heading), *FN* (false negative – the number of heading units identified as non-heading), and *TN* (true negative – the number of non-heading units identified correctly). Based on these values, we calculated the recall (*R*), precision (*P*), and f-measure (*F*) results for the heading extraction experiment as shown in Eqn. (4):

$$R \equiv \frac{TP}{TP + FN} \quad P \equiv \frac{TP}{TP + FP} \quad F \equiv \frac{2 \times P \times R}{P + R} \tag{4}$$

The accuracy of hierarchy extraction is determined by exploiting the structure in the output trees. We focus on the parent-child relationships in the trees since they correspond to the *heading-subheading* and *heading-underlying text* relationships. We define the accuracy for a document as the ratio of the number of correctly identified parent-child relationships (as compared with the gold standard) to the total number

Table 3. *Feature sets used in heading extraction*

| Feature set | Features | Number of features |
|---|---|---|
| $\Phi_1$ | $F_i$, $F_{i(i+1)}$ | 58 |
| $\Phi_2$ | $F_i$, $F_{i(i+1)}$, $F_{i(i-1)}$ | 86 |
| $\Phi_3$ | $F_i$, $F_{i(i+1)}$, $F_{i(i+2)}$ | 82 |
| $\Phi_4$ | $F_i$, $F_{i(i+1)}$, $F_{i(i+2)}$, $F_{i(i-1)}$ | 110 |
| $\Phi_5$ | $F_i$, $F_{i(i+1)}$, $F_{i(i+2)}$, $F_{i(i-1)}$, $F_{i(i-2)}$ | 134 |

of parent-child relationships. Formally, given a manually identified hierarchy and an automatically extracted hierarchy for a document $i$, if there exists an edge between a node pair $(p,c)$ in both of the hierarchies, we say $e(p,c) = 1$; otherwise, $e(p,c) = 0$. Given the set of manually identified parent-child node pairs $PC_i$ for document $i$, the hierarchy accuracy is computed as shown in Eqn. (5). Note that we do not use the precision and recall metrics in addition to the accuracy metric in hierarchy evaluation, since precision and recall yield the same results as accuracy in this problem.

$$Accuracy_i \equiv \frac{\sum_{(p,c)\in PC_i} e(p,c)}{|PC_i|} \tag{5}$$

### *6.2.1 Heading extraction*

We used the classification model explained in Section 4 to classify the text units as heading or non-heading. Intuitively the idea is that heading and non-heading units can be determined by considering their features relative to the units in their context. For instance, a heading unit is usually more emphasized than the immediately following unit. We defined five feature sets using different combinations of features as shown in Table 3. $F_i$ denotes the features of the current unit; $F_{i(i+1)}$ and $F_{i(i+2)}$ denote the contextual features related to the current unit and, respectively, the succeeding unit and the unit following the succeeding unit; and $F_{i(i-1)}$ and $F_{i(i-2)}$ denote the contextual features related to the current unit and, respectively, the preceding unit and the unit before the preceding unit.

We evaluated the performance of heading extraction using SVM and perceptron with different feature sets. In the heading extraction task, the number of negative examples (non-heading text units) is much larger than the number of positive examples (heading units), resulting in relatively low recall rates. In the literature, there exist several approaches for dealing with such unbalanced distributions, including oversampling or undersampling the classes (by random resampling, directed resampling, or adding new samples), adjusting the costs of the classes, recognition-based learning, or using modified forms of learning algorithms (Chawla 2005; Weiss, McCarthy and Zabar 2007; Ganganwar 2012). In this work, we adopted the method of using different costs for the positive and negative classes. In the case of SVM, we experimented with different cost factors to adjust the cost of training errors on positive examples (false positives) versus the cost of errors on negative examples

Table 4. *Performance results for heading extraction*

| Method | Feature set | Recall | Precision | F-measure |
|---|---|---|---|---|
| SVM – Linear | $\Phi_1$ | **0.85** | **0.78** | **0.81** |
| | $\Phi_2$ | 0.83 | 0.78 | 0.80 |
| | $\Phi_3$ | 0.81 | 0.77 | 0.79 |
| | $\Phi_4$ | 0.83 | 0.78 | 0.80 |
| | $\Phi_5$ | 0.83 | 0.78 | 0.80 |
| SVM – Polynomial | $\Phi_1$ | 0.87 | 0.80 | 0.83 |
| | $\Phi_2$ | 0.85 | 0.80 | 0.82 |
| | $\Phi_3$ | 0.87 | 0.82 | 0.84 |
| | $\Phi_4$ | 0.85 | 0.80 | 0.82 |
| | $\Phi_5$ | **0.87** | **0.84** | **0.85** |
| SVM – RBF | $\Phi_1$ | 0.84 | 0.76 | 0.80 |
| | $\Phi_2$ | 0.84 | 0.79 | 0.81 |
| | $\Phi_3$ | 0.87 | 0.81 | 0.84 |
| | $\Phi_4$ | **0.88** | **0.83** | **0.85** |
| | $\Phi_5$ | **0.87** | **0.83** | **0.85** |
| Perceptron | $\Phi_1$ | 0.71 | 0.77 | 0.74 |
| | $\Phi_2$ | 0.70 | 0.78 | 0.74 |
| | $\Phi_3$ | 0.71 | 0.84 | 0.77 |
| | $\Phi_4$ | **0.78** | **0.82** | **0.80** |
| | $\Phi_5$ | 0.77 | 0.81 | 0.79 |
| Rule-based approach | – | **0.72** | **0.64** | **0.68** |
| SVM – Polynomial | $F_i$ | 0.77 | 0.74 | 0.70 |
| | $F_{i(i+1)}$ | 0.78 | 0.73 | 0.76 |
| | $F_{i(i+2)}$ | 0.70 | 0.72 | 0.67 |
| | $F_{i(i-1)}$ | 0.76 | 0.70 | 0.69 |
| | $F_{i(i-2)}$ | 0.65 | 0.68 | 0.62 |

(false negatives). In the experiments presented in this section, a cost factor of two was used as it yields the best performances. We also experimented with different kernel types, which are the linear, polynomial and RBF kernels. In polynomial kernel, we used $d = 2$ as the degree of the polynomial which yielded more accurate results in the tests.

Table 4 shows the recall, precision and f-measure results for heading extraction. The best results in each method with respect to the f-measure are displayed in bold. The best results were obtained using SVM with polynomial and RBF kernels. It seems that as more features are included the accuracies improve in both of the learning algorithms. In this respect, $\Phi_4$ and $\Phi_5$ yield the highest f-measure values. The effect of using different feature sets is less obvious for linear SVM. However, especially for SVM with RBF kernel and perceptron, incorporating more contextual information in the model generally improves the f-measure rates. The results also show that machine learning approaches provide significant increase in the accuracy compared to the rule-based approach. The improvements are more clear with the use of a nonlinear technique (i.e. polynomial or RBF kernel) in SVM. ANOVA

Table 5. *Feature sets used in hierarchy extraction*

| Feature set | Features | Number of features |
|---|---|---|
| $\Gamma_1$ | $F_{10}$ | 17 |
| $\Gamma_2$ | $F_{10}$, $F_{01}$ | 40 |
| $\Gamma_3$ | $F_{10}$, $F_{01}$, $F_{20}$ | 57 |
| $\Gamma_4$ | $F_{10}$, $F_{01}$, $F_{20}$, $F_{02}$ | 73 |

tests verify that the results are significant with $p < 0.05$ for recall, $p < 0.0001$ for precision, and $p < 0.0001$ for f-measure.

To understand the effect of different types of features on heading extraction, we also performed experiments using the unit features and the contextual features separately. We used SVM with polynomial kernel as it shows successful behavior in heading extraction. The last part of the table shows the results for the unit features $F_i$, and the contextual features $F_{i(i+1)}$, $F_{i(i+2)}$, $F_{i(i-1)}$ and $F_{i(i-2)}$. We see that when only the unit features are used or only a limited amount of context is taken into account, the success rates drop significantly. This indicates that, while deciding on whether a unit is a heading or not, both features about that unit and features that encode its relations with its context should be considered. We also notice that the context following a unit is much more important than the previous context. This observation is in accordance with the results obtained in other natural language processing tasks (Indurkhya and Damerau 2010).

### *6.2.2 Hierarchy extraction*

We evaluated the tree-based learning approach used for the identification of sectional hierarchies in documents. The initial experiments showed that unit features are not effective in hierarchy extraction since they encode only absolute values related to a single unit (e.g. font size of a text unit). Instead, contextual features which show the relationships between pairs of units (e.g. font size difference between two units) should be used. This observation is in parallel with the nature of the hierarchy extraction task that aims to determine the correct parent-child unit pairs in the documents. Hence, we employed only the contextual features in the hierarchy extraction models.

Following the notation given in Section 4.3.2, we name the contextual features of a unit $u$ as $F_{10}$ (features related to $u$ and its candidate parent unit), $F_{01}$ and $F_{02}$ (features related to $u$ and its two candidate sibling units that are on the left in the partial tree), and $F_{20}$ (features related to $u$ and its candidate grandparent unit). We defined four feature sets using different combinations of features as shown in Table 5.

The initial experiments also showed that the use of heading information improves the accuracy of hierarchy extraction. Thus, the output of the heading extraction step was also utilized in the hierarchy identification step (line 2 in Figure 6). The

Table 6. *Accuracies of SVM and perceptron in hierarchy extraction*

| Learning algorithm | Feature set | | | |
|---|---|---|---|---|
| | $\Gamma_1$ | $\Gamma_2$ | $\Gamma_3$ | $\Gamma_4$ |
| 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 |

headings obtained with the polynomial SVM and the feature set $\Phi_5$ (all features) were used since it was one of the best models in heading extraction.

The accuracies obtained in the hierarchy extraction experiments are given in Table 6 for different feature combinations and a beam width of 100. The results show that the SVM-based approach performs significantly better than the perceptron-based approach for high feature numbers. The nonlinear SVM models (polynomial and RBF kernels) yield more successful results than the linear kernel. Surprisingly, the performance of the perceptron algorithm, which is a linear approach, decreases as more contextual features are included. The best accuracies were obtained when the contextual differences with only the parent unit were considered. The analysis of the success rates with respect to the feature sets shows that each algorithm shows a similar behavior with $\Gamma_2$, $\Gamma_3$ and $\Gamma_4$. This indicates that the information provided by non-adjacent units (grandparents and distant siblings) does not contribute much in identifying the correct hierarchical position of a unit. The rule-based approach used in Pembe and Güngör (2009) resulted in about 61 per cent accuracy for hierarchy extraction. Considering this as a baseline, we see that the machine learning approach (nonlinear SVM models with increasing number of contextual features) outperforms the rule-based approach significantly.

In order to analyze the effect of the beam width on the incremental learning paradigm, we repeated the experiments with varying beam width factors. Table 7 shows the results for SVM polynomial and RBF kernels for beam widths ranging from 1 to 100. As can be seen, the beam width does not have a significant effect on the accuracies. This result indicates that it is sufficient to store only a few partial trees at each iteration, highly reducing the time and space complexities. The incremental learning approach allows us to solve the problem in polynomial time, as opposed to an exponential time approach that considers all possible tree structures that can be formed from the text units in a document. Similarly, with regard to the space requirements, we need to store only a number of trees that is dependant on the beam width during processing, rather than storing all possible partial trees.

We also analyzed the success rates of the models explained in Section 4.4 and compared with the main model. Table 8 shows the results of the main model ($M_0$) and its variations ($M_1$ to $M_4$) for the feature set $\Gamma_4$ and a beam width of 100. It seems that all the models except $M_2$ where hierarchy extraction is formed of two consecutive steps show similar behavior. First extracting the hierarchy of the headings and then placing the non-headings into this hierarchy does not work. The

Table 7. *Effect 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 |

Table 8. *Accuracies of 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 |

reason is that at each of these independent steps the algorithm cannot make use of the information used in the other step. For instance, while determining the correct hierarchical location of a heading, in addition to the locations of other headings in the hierarchy, the properties (features) of the non-headings may also be important. The other variations of the main model have similar performance which indicates that using scores produced by the machine learning component or converting them into probabilities or ranks does not have a significant impact on the success rates.

Considering heading identification as a preprocessing step for hierarchy extraction, we measured the performance loss in hierarchy extraction that results from this preprocessing step. For this purpose, we compared the results of hierarchy extraction that is based on automatically identified headings (as discussed above) and manually identified (gold standard) headings. Table 9 shows the results for the feature set $\Gamma_4$ and a beam width of 100 for both rule-based and machine learning models. As can be expected, using gold standard headings causes a significant improvement on the success rates. Given the heading structure of a document, it is possible to extract the hierarchical structure with about 80 per cent performance using learning algorithms. On the other hand, when the headings are also extracted by the algorithm, the success rate drops significantly. However, it should be noted that the latter case is a fully automatic and more realistic approach. Given an input document, the document hierarchy (heading and non-heading hierarchies) is completely extracted. The methods introduced show that this can be done with nearly 70 per cent success rates.

### *6.3 Summarization experiments*

We conducted a task-based evaluation of the system where the task is Web search. The document sectional hierarchies were fed into the summarization module explained in Section 5. The summaries output by the summarization system were

Table 9. *Manual versus automatic heading identification in hierarchy extraction*

| Method | Heading extraction model | Manually extracted headings |
|---|---|---|
| Rule-based approach | 0.61 | 0.81 |
| Perceptron | 0.51 | 0.82 |
| SVM | 0.68 | 0.79 |

evaluated in terms of their usefulness in a search engine. Five types of summaries were used in the experiments:

- *Unstructured1*, *Unstructured2* – Query-biased summaries without the use of structural information.
- *Structured1*, *Structured2* – Structure-preserving and query-biased summaries created by the system.
- *Google* – Query-biased extracts provided by Google.

In the models *Unstructured2*, *Structured1*, and *Structured2*, the parts in the documents that cause a cluttered view (navigation menus, links, etc.) were identified using some heuristics and eliminated. In *Unstructured1*, such document parts were kept in the documents. *Structured1* is based on the output of the structural processing step, whereas *Structured2* is based on the manually identified document structure. All the summaries in these models are long summaries with similar sizes (about 25 sentences) to make them comparable with each other. The model referred to as *Google* corresponds to the extracts (2–3 lines summaries) displayed by the search engine Google as a result of a user query. Since the outputs of Google and the proposed models have different sizes, we cannot make a direct comparison between the model *Google* and the other four models. However, we include *Google* in this evaluation in order to compare the results obtained in this work with those of a state-of-the-art search engine.

The structured summaries are displayed in a hierarchical way in accordance with the sectional hierarchy. The headings and subheadings are shown in bold. The context of the text fragments selected as part of the summary is provided within this structure. In this way, the users are expected to judge the relevance of the search results better. An example summary output of the system (*Structured1* model) for the query *Amazon rain forest* is given in Figure 8.

The queries used in the evaluation (see Table 2) cover different types of information needs of users (Ingwersen and Jarvelin 2005): search for a number of items (12 queries), decision search (2 queries), and background search (6 queries). A repeated measures design was used with four subjects. The summaries were evaluated based on the relevance judgment paradigm instead of relying on a gold standard (Mani *et al.* 2002). As the success criterion, we used the relevance prediction measure that was shown to yield more reliable results than manual metrics for summary evaluation (Hobson *et al.* 2007). Human subjects were asked to determine the relevance of original documents with respect to the given query (either relevant or not) using the summaries. That is, a subject's judgment on a summary is compared

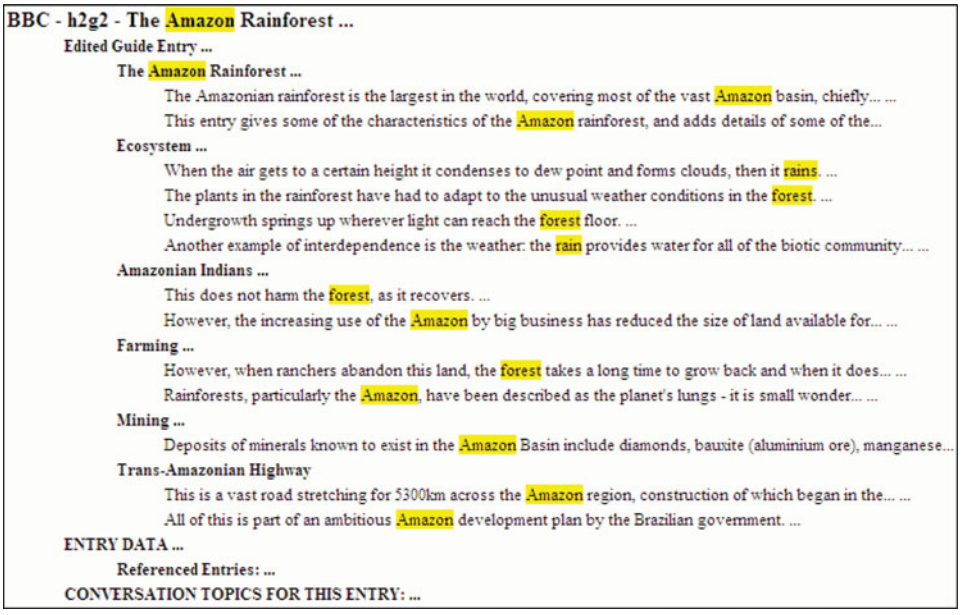


Fig. 8. (Colour online) An example summary output of the proposed system.

with his or her own judgment on the original document. If the subject marks both the document and the summary as relevant or irrelevant, the summary is deemed as a successful summary. In this way, the effect of differences in human subjects is reduced. The summaries and documents were presented in random order in order to reduce carry-over effects and an original document was not displayed until all the summaries of that document were displayed. In addition, we included a user poll during evaluation to rate the helpfulness of each summary. For this purpose, a 5-point Likert scale was used with 1 as not helpful and 5 as very helpful (Tombros and Sanderson 1998; White *et al.* 2003).

For each summarization model, four different metrics were calculated by comparing the relevance judgments of the users for the summaries and the original documents: *TP* (true positive), *FP* (false positive), *FN* (false negative), and *TN* (true negative). Using these values, the recall ($R$), precision ($P$), and f-measure ($F$) results were obtained (Eqn. (4)). In addition, accuracy ($A$), false negative rate ($FNR$) and false positive rate ($FPR$), which correspond to, respectively, the number of useful summaries over the number of all summaries, the number of summaries that correspond to documents relevant to the query but marked as irrelevant over the number of all relevant summaries, and the number of summaries that correspond to documents that are not relevant to the query but marked as relevant over the number of all irrelevant summaries were computed as shown in Eqn. (6):

$$A \equiv \frac{TP + TN}{TP + TN + FP + FN} \quad FNR \equiv \frac{FN}{FN + TP} \quad FPR \equiv \frac{FP}{FP + TN} \tag{6}$$

A total of 400 relevance judgments were obtained for each summarization method. The results are shown in Table 10. We used the most successful hierarchy extraction model (model $M_4$, SVM with polynomial kernel, feature set $\Gamma_4$) for *Structured1*. The

Table 10. *Results of the summarization experiment*

| System | TP | FP | FN | TN | A | P | R | F | FNR | FPR |
|---|---|---|---|---|---|---|---|---|---|---|
| *Unstructured1* | 179 | 54 | 59 | 108 | 0.72 | 0.77 | 0.75 | 0.73 | 0.23 | 0.32 |
| *Unstructured2* | 176 | 53 | 62 | 109 | 0.72 | 0.77 | 0.73 | 0.72 | 0.24 | 0.30 |
| *Structured1* | 185 | 50 | 53 | 112 | 0.74 | 0.78 | 0.77 | 0.76 | 0.20 | 0.30 |
| *Structured2* | 183 | 40 | 55 | 122 | 0.75 | 0.82 | 0.76 | 0.77 | 0.22 | 0.24 |
| *Google* | 118 | 36 | 120 | 126 | 0.57 | 0.72 | 0.47 | 0.52 | 0.50 | 0.23 |

Table 11. *Average judgment times, summary/document sizes and ratings*

| System | Time (seconds) | Size (words) | User rating |
|---|---|---|---|
| *Unstructured1* | 17.70 | 298 | 2.77 |
| *Unstructured2* | 18.44 | 306 | 2.77 |
| *Structured1* | 17.51 | 277 | 3.03 |
| *Structured2* | 17.02 | 274 | 3.12 |
| *Google* | 10.20 | 30 | 2.60 |
| *Original Document* | 23.59 | 1340 | – |

results show that structured summaries are superior to unstructured summaries in terms of accuracy, precision, recall and f-measure. Also, they significantly reduce the false negative rate (the number of search results missed by the user) and the false positive rate (the time spent with irrelevant items). When compared to the model *Google*, we see that the long outputs of the unstructured and structured models enable the users to determine the relevancy of search results better than the extracts produced by Google. The repeated measures ANOVA test verifies that the results (structured models over unstructured models and unstructured models over *Google*) are significant with $p < 0.05$ for both accuracy and f-measure. Table 11 shows the average time for deciding the relevancy of a summary/document, the average size of a summary/document, and the average user ratings. The last row of the table corresponds to the original document for comparison. The table shows that the models based on (structured or unstructured) summaries have acceptable judgment times despite the summary sizes much longer than search engine snippets. Also, we observe that structured summaries have quite high user ratings compared to unstructured summaries and search engine extracts.

It is worth noting that we use the model *Google* in the experiments in order to compare the approach proposed in this work (long and structured summaries) with the approach used by traditional search engines. We do not aim at making a direct comparison between the outputs, which would result in an unfair comparison due to the difference in the size of the content displayed to the user. The use of a search engine in the experiments shows us useful information in two dimensions. First, we observe that a model that displays structured and richer content to the users increases the relevancy of the results significantly and thus seems to be more helpful than the current search engine models. Second, we see that the users can judge

the relevancy of the results in an acceptable time although they are provided with much more content. In these respects, we conclude that the summaries produced by the system built in this work direct the users more effectively at a cost of a small increase in response time.

## 7 Discussion

In this section, we compare the results obtained in this work with the results of previous studies and discuss some issues relevant to document structure analysis and summarization. There are only a few works that address the problem of extracting the structures of general Web documents in terms of sectional hierarchies. Some of the studies attempt to identify the important contents in the documents and represent these in a suitable form like a table-of-contents structure, whereas others conduct a more detailed analysis to convert a document into a structural representation. Here we list a number of studies most related to the current work with respect to the research goal.

Related to the problem of heading extraction, Xue *et al.* (2007) applied machine learning techniques to extract the main title in HTML documents. The proposed method is based on a specification for titles and the DOM tree representation of Web pages. Similar to the current work, the documents are represented using formatting features, visual features, and a combination of both types of features. The experiments on a set of documents from the TREC Web Track yielded about 0.75–0.80 f-measure results for title extraction. The authors also measured the effectiveness of the extracted titles in a Web retrieval task. By using the extracted titles together with the titles and contents of the Web pages, the retrieval performance of the system was tested on three types of TREC queries. It was shown that the output of title extraction improves the success rates significantly and precision values up to 65 per cent can be obtained. In the current work, instead of extracting the main title (i.e. a single heading) in the documents, we focused on a more general and challenging problem which involves extracting all the headings in a given HTML document. We have shown that success rates of about 0.85 f-measure can be obtained for the identification of headings and subheadings by using a learning algorithm based on a rich feature set.

The work of Branavan *et al.* (2007) dealt with the problem of building a table-of-contents structure for long documents. This problem can be approached from either text summarization or title generation perspective. The work made use of two types of features in the learning algorithm: local features formed of words and word sequences to represent each part of a document with a meaningful title and global features to obtain a coherent structure in the table-of-contents. The proposed method was compared with four alternative approaches that generate flat and hierarchical title structures. A detailed performance analysis was not given; it was shown by manual evaluations that the proposed method significantly outperforms the alternative methods. The main difference of this work from the current work is that it aims at representing document segments with titles, whereas

in the current work we aim at extracting both the titles and the important content in the documents.

The main aim of structural processing is to obtain the heading-based sectional hierarchy for an HTML document. As part of developing a summarization framework, Pembe and Güngör (2009) focused on a similar problem and used a rule-based approach for identifying the hierarchical structures implicit in Web documents. A set of formatting features that are important in discriminating different parts in a document was identified and a hierarchy extraction algorithm was developed based on the formatting differences. The experiments on TREC documents showed about 71 per cent success rates. In the current work, on the other hand, we use a fully automatic approach where the hierarchy is built incrementally instead of relying on a predefined set of rules. In the fully automatic setting, we obtained an accuracy of 68 per cent. We also observed that when a partial structure is provided in the form of manually identified headings, the accuracy increased to 82 per cent. This success rate signals that the machine learning-based approach resulted in 11 per cent improvement over the rule-based approach developed in the mentioned work.

Another study that uses the approach of rule-based processing of the DOM tree to obtain hierarchical structures of Web documents was proposed by Mukherjee *et al.* (2003). This work is based explicitly on the structure of the paths in the DOM tree in the sense that two nodes that have common tags in their paths from the root node are deemed as similar. The DOM tree of a given Web page is processed with respect to this similarity measure and the nodes are clustered accordingly, which results in a hierarchical organization of the page. The method was applied to documents in a number of domains (portals, news, and office products) and the hierarchies built were given, but a detailed performance analysis was not performed. The main difference between this work and the current work lies, as in some of the other mentioned studies, in the content of the documents taken into consideration. The study aims at obtaining a hierarchical structure of the documents without dealing with the task of representing the actual content within the hierarchy.

The problem of expressing the results of a search engine query in terms of document summaries was addressed by a few works. White *et al.* (2003) developed a system called WebDocSum that produces query-biased summaries for a search engine. Similar to the summary extraction approach used in the current work, WebDocSum made use of a number of heuristics well-known in the text summarization domain (e.g. document title or query words) to generate the output summaries. The system was evaluated using a user questionnaire that compares it with other traditional search engines and it was shown that it outperforms the other engines with respect to user ratings. Although the system built in this work produces query-biased summaries, it does not take into account the structural information in the documents. A work that makes use of structural data in the documents as well as the information provided by the query was given in Pembe and Güngör (2009). In this work, a number of summarization metrics that take the hierarchical structures in Web documents into account were developed. Experiments on TREC queries showed around 80 per cent success rates. The main difference from the current work is that the summarization system was based on semi-automatic and

manual hierarchical structures. In the current work, we developed a fully automatic summarization system that yields around 76 per cent success rates.

In the rest of this section, we comment on some issues related to the results obtained by the developed system. Although we obtained a high performance in heading extraction (0.85 f-measure), the use of the output of this step instead of gold standard headings causes a 10–15 per cent decrease in accuracy in hierarchy extraction. To understand the source of this behavior, we conducted an error analysis on the hierarchy extraction results. We saw that false negatives in heading extraction result in a loss of structure, whereas false positives result in the creation of structures that do not actually exist. Another source of error is due to using a beam width during incremental learning which causes in some cases the correct partial trees to be eliminated during processing. Other sources of inaccuracies originate from the cluttered structures of Web documents and from the errors made by Web document authors, such as ambiguous or wrong usage of tags and styles. Besides such errors, we can conclude that the method results in an acceptable performance as a fully automatic approach for sectional hierarchy extraction which can be used by practical applications.

The application of the proposed methods to the document summarization problem in the domain of search engines showed that structure-based and query-biased summaries improve the effectiveness of Web search. Structured summaries allow the users to determine the relevancy of a search result to a given query 50 per cent more accurately. Although they are about nine times longer than the extracts produced by search engines, such summaries result in an increase of only 70 per cent in user response time. Besides, the ratio of relevant documents missed by the users using search engine extracts is more than twice the ratio of relevant documents missed using structured summaries. We see that there is a tradeoff between the response time and the success rate when longer, more structured summaries are used.

In the case of common-place queries (e.g. *the population of Germany*), the users can locate the relevant information just by browsing a few of the top search results. The time overhead caused by the summarization method is less important in such cases. In the case of more complex queries and background search, the accuracy becomes more important. For those types of queries, structured summaries are preferable since they result in a reduced number of missed items and direct the users to the relevant documents without wasting time with irrelevant documents. User ratings verify that structured summaries are more helpful and preferable by users compared to short extracts. The system built is a practical approach which can be incorporated into a search engine. The structural processing stage can be executed offline and only once as new documents are added, whereas the summarization stage has linear time complexity.

## 8 Conclusions

In this paper, we investigated the problem of heading-based sectional hierarchy extraction for domain-independent Web documents. To overcome the exponential search space problem that occurs naturally in such tasks, we developed an

incremental learning approach using SVM and perceptron algorithms. The system is formed of two stages: heading extraction and hierarchy extraction. In the heading extraction stage, an f-measure of 0.85 was obtained. The output of heading extraction was used in the hierarchy extraction stage which resulted in 68 per cent accuracy. We observed that the machine learning approach used in this work as a fully automatic structure identification model outperformed the rule-based approach.

The output of the structural processing phase was utilized for automatic summarization of documents in the Web search context. The performance of the structure-based and query-biased summarization method was measured as 76 per cent in terms of f-measure in a task-based evaluation. We showed that there is a statistically significant performance gain over unstructured summaries. Also, a comparison with a state-of-the-art search engine showed that the outputs produced by the system built in this work direct the users to relevant documents much more accurately.

The proposed structural processing approach can be used in several application areas in addition to text summarization and search engines, including areas related to browsing, indexing and classification of documents. Besides the Web domain, it can be used in information systems that deal with large amounts of documents, such as medicine, libraries and law. As future work, we plan to work on methods to identify some document components commonly encountered in Web pages, such as menus and advertisements. Such information can be used to improve the document hierarchies and the summaries by eliminating irrelevant information.

## Acknowledgements

This work was supported by the Boğaziçi University Research Fund under the Grant number 07A106.

## References

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

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

Branavan, S. R. K., Deshpande, P., and Barzilay, R. 2007. Generating a table-of-contents. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* 176-183,Prague, Czech Republic.

Brugger, R., Zramdini, A., and Ingold, R. 1997. Modeling documents for structure recognition using generalized n-grams. In *Proceedings of International Conference on Document Analysis and Recognition*, Ulm, Germany, pp. 56–60.

Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. 2001. Accordion summarization for end-game browsing on PDAs and cellular phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, pp. 213-220.

Buyukkokten, O., Kaljuvee, O., Garcia-Molina, H., Paepcke, A., and Winograd, T. 2002. Efficient web browsing on handheld devices using page and form summarization. *ACM Transactions on Information Systems* **20**(1): 82–115.

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

Chawla, N. V. 2005. Data mining for imbalanced datasets: an overview. In O. Maimon, and L. Rokach (eds.), *Data Mining and Knowledge Discovery Handbook*, pp. 853–67. New York: Springer.

Chen, Y., Ma, W., and Zhang, H. 2003. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th International World Wide Web Conference*, New York, NY, USA, pp. 225–33.

Cobra: Java HTML Renderer and Parser 2010. http://lobobrowser.org/cobra.jsp.

Collins, M., and Roark, B. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics,* Stroudsburg, PA, USA, pp. 111–8.

Covington, M. A. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of ACM Southeast Conference*, Athens, GA, USA, pp. 95–102.

Curran, J. R., and Wong, R. K. 1999. Transformation-based learning for automatic translation from HTML to XML. In *Proceedings of the 4th Australasian Document Computing Symposium*, Coffs Harbour, NSW, Australia, pp. 55–62.

Desmarais, F.-X., Gagnon, M., and Zouaq, A. 2012. Comparing a rule-based and a machine learning approach for semantic analysis. In *Proceedings of the 6th International Conference on Advances in Semantic Processing*, Barcelona, Spain, pp. 103–8.

Document Object Model 2012. http://www.w3.org/dom.

Feng, J., Haffner, P., and Gilbert, M. 2005. A learning approach to discovering web page semantic structures. In *Proceedings of the 8th International Conference on Document Analysis and Recognition*, Washington, DC, pp. 1055–9.

Ganganwar, V. 2012. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering* **2**(4): 42–7.

Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. 2003. DOM-based content extraction of HTML documents. In *Proceedings of the 12th International Conference on World Wide Web*, New York, NY, USA, pp. 207–14.

Gupta, S., Kaiser, G., and Stolfo, S. 2005. Extracting context to improve accuracy for HTML content extraction. In *Proceedings of the 14th International Conference on World Wide Web*, New York, NY, USA, pp. 1114–5.

Hastie, T., Tibshirani, R., and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 3rd ed. New York: Springer.

Hobson, S. P., Dorr, B. J., Monz, C., and Schwartz, R. 2007. Task-based evaluation of text summarization using relevance prediction. *Information Processing and Management* **43**(6): 1482–99.

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

Indurkhya, N., and Damerau, F. J. 2010. *Handbook of Natural Language Processing*, 2nd ed. Boca Raton, FL: CRC Press.

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

Irmak, U., and Kraft, R. 2010. A scalable machine-learning approach for semi-structured named entity recognition. In *Proceedings of the International World Wide Web Conference*, New York, NY, USA, pp. 461–70.

Jensen, S. H., Madsen, M., and Moller, A. 2011. Modeling the HTML DOM and browser API in static analysis of JavaScript web applications. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, Szeged, Hungary, pp. 59–69.

Joachims, T. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola (eds.), *Advances in Kernel Methods - Support Vector Learning*. Cambridge, UK: MIT Press.

Joachims, T. 2002. *Learning to Classify Text Using Support Vector Machines*. Boston: Kluwer Academic Publishers.

Kao, H.-Y., Ho, J.-M., and Chen, M.-S. 2004. DOMISA: DOM-based information space adsorption for web information hierarchy mining. In *Proceedings of the 4th SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, USA, pp. 312–20.

Klink, S., Dengel, A., and Kieninger, T. 2000. Document structure analysis based on layout and textual features. In *Proceedings of the International Workshop on Document Analysis Systems*, Kaiserslautern, Germany, pp. 99–111.

Le, D. X., and Thoma, G. R. 2003. Automated document labeling for Web-based online medical journals. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Florida, USA, pp. 411–15.

Li, Y., Wang, L., Wang, J., Yue, J., and Zhao, M. 2013. An approach of web page information extraction. In *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*, Hangzhou, China, pp. 2217–9.

Liu, Y., Wang, Q., and Wang, QX. 2006. A heuristic approach for topical information extraction from news pages. In *Proceedings of the 7th International Conference on Web Information Systems Engineering*, Wuhan, China, pp. 357–62.

Mandhani, B., and Meila, M. 2009. Tractable search for learning exponential models of rankings. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, Clearwater, Florida, USA, pp. 392–9.

Mani, I. 2001. *Automatic Summarisation*. Amsterdam: John Benjamins.

Mani, I., Klein, G., House, D., Hirschman, L., Firmin, T., and Sundheim, B. 2002. SUMMAC: a text summarization evaluation. *Natural Language Engineering* **8**(1): 43–68.

Mao, S., Rosenfeld, A., and Kanungo, T. 2003. Document structure analysis algorithms: a literature survey. In *Proceedings of SPIE Electronic Imaging*, Santa Clara, California, USA, pp.197–207.

Markey, K. 2007. Twenty-five years of end-user searching, Part 1: research findings. *Journal of the American Society for Information Science and Technology* **58**(8): 1071–81.

Mayfield, J., McNamee, P., Piatko, C., and Pearce, C. 2003. Lattice-based tagging using support vector machines. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, New Orleans, Los Angeles, USA, pp. 303–8.

Mukherjee, S., Yang, G., Tan, W., and Ramakrishnan, I. V. 2003. Automatic discovery of semantic structures in HTML documents. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*, Edinburgh, UK, pp. 245–9.

Niyogi, D., and Srihari, S. N. 1995. Knowledge-based derivation of document logical structure. In *Proceedings of the International Conference on Document Analysis and Recognition*, Montreal, Canada, pp. 472–5.

Pembe, F. C., and Güngör, T. 2009. Structure-preserving and query-biased document summarisation for web searching. *Online Information Review* **33**(4): 696–719.

Pinto, D., McCallum, A., Wei, X., and Croft, W. B. 2003. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada, pp. 235–42.

Platt, J. C. 1999. 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. Cambridge, UK: MIT Press.

Rahman, A. F. R., Alam, H., and Hartono, R. 2001. Content extraction from HTML documents. In *Proceedings of the 1st International Workshop on Web Document Analysis*. Seattle, Washington, USA.

Shilman, M., Liang, P., and Viola, P. 2005. Learning non-generative grammatical models for document analysis. In *Proceedings of the 10th IEEE International Conference on Computer Vision*, Beijing, China, pp. 962–9.

Text Retrieval Conference 2010. http:// trec.nist.gov.

Tombros, A., and Sanderson, M. 1998. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp. 2–10.

Weiss, G. M., McCarthy, K., and Zabar, B. 2007. Cost-sensitive learning vs. sampling: which is best for handling unbalanced classes with unequal error costs? In *Proceedings of the International Conference on Data Mining*, Las Vegas, Nevada, USA, pp. 35–41.

White, R. W., Jose, J. M., and Ruthven, I. 2003. A task-oriented study on the influencing effects of query-biased summarization in web searching. *Information Processing and Management* **39**(5): 707–33.

Xiao, X., Luo, Q., Hong, D., Fu, H., Xie, X., and Ma, W-Y. 2009. Browsing on small displays by transforming web pages into hierarchically structured subpages. *ACM Transactions on the Web* **3**(1): 4:1–4:36.

Xue, Y., Hu, Y., Xin, G., Song, R., Shi, S., Cao, Y., Lin, C. Y., and Li, H. 2007. Web page title extraction and its application. *Information Processing and Management* **43**(5): 1332–47.

Yang, C. C., and Wang, F. L. 2008. Hierarchical summarization of large documents. *Journal of the American Society for Information Science and Technology* **59**(6): 887–902.

Yang, Y., and Zhang, H. J. 2001. HTML page analysis based on visual cues. In *Proceedings of the 6th International Conference on Document Analysis and Recognition*, Washington, USA, p. 859.