Web Search Based on Hierarchical Heading-Block Structure Analysis

Department of Social Informatics,
Graduate School of Informatics,
Kyoto University, Japan
Tomohiro Manabe

Web Search Based on Hierarchical Heading-Block Structure Analysis (the 1st imp. of the 1st ed.), written by Tomohiro Manabe, finished Feb. 28th, 2016. Doctoral dissertation, Department of Social Informatics, Graduate School of Informatics, Kyoto University, Japan.

ABSTRACT

Authors write headings for splitting a document into multiple semantic blocks of different topics. A block may include some other blocks, and the blocks in a document compose hierarchical heading structure. Information on hierarchical heading structure is very useful in document retrieval because the structure represents the topic structure, which is the most important factor in document retrieval. We can use the information on hierarchical heading structure in various ways for improving document retrieval systems. In this dissertation, we focus on the largest collection of digital documents, the World Wide Web. We first propose a method for extracting hierarchical heading structure in web pages, and also propose four methods that use the information on the extracted structure for improving web search systems. In the following, we explain the overview of these five methods.

Extracting Logical Hierarchical Structure of HTML Documents Based on Headings

We propose a method for extracting hierarchical heading structure in web pages in hypertext markup language (HTML). Human readers easily understand the structure by exploiting the following properties of headings: (1) headings appear at the beginning of the corresponding blocks, (2) headings are given prominent visual styles, (3) headings of the same level share the same visual style, and (4) headings of higher levels are given more prominent visual styles. Our method also exploits these properties. Our experiment shows our method outperforms existing methods.

Subtopic Ranking Based on Hierarchical Headings

We propose methods for generating diversified rankings of subtopics of keyword queries. Our methods are aware of hierarchical heading structure. Each heading concisely describes the topic of its corresponding block. Therefore, hierarchical headings in documents reflect the hierarchical topics referred to in the documents. Our methods score subtopics based on matching between the subtopics and hierarchical headings in documents. They give higher scores to subtopics matching hierarchical headings associated to more contents. Our methods generated significantly better subtopic rankings than query completion results by major commercial search engines.

Abstract

Heading-Aware Proximity Measure and Its Application to Web Search

Proximity of query keyword occurrences is one important evidence useful for effective document scoring. If a query keyword occurs close to another in a document, it suggests strong relevance of the document to the query. Most web pages contain hierarchical heading structure, and it affects logical proximity. For example, occurrences in headings describe the topics of the entire corresponding blocks, and the occurrences are strongly connected to any term occurrences in the blocks regardless of the simple distance between them. Based on such observations, we developed a heading-aware proximity measure, heading-aware semi-distance, applied the measure to three existing proximity-aware document scoring functions, and evaluated the existing and modified functions on the data sets of a well-known Text Retrieval Conference (TREC) web tracks. Our heading-aware Span method generated rankings significantly better than the existing Span method with simple distance.

Heading-Aware Block-Based Web Search

We propose a web search system that improves the ranking of search results by exploiting the hierarchical heading structure in web pages. Because a heading is a concise description of the topic of its associated block, query term occurrences in a heading indicate high relevance of the block to the query intent. In hierarchical heading structure, headings of ancestor blocks describe the topics of all their descendant blocks, although the ancestor headings may be located far from the descendant blocks. Based on these facts, we developed a search system which retrieves and scores blocks by taking into account their ancestor headings as well as their contents. We evaluated our system by using TREC data sets. The result indicates that our system can eliminate many irrelevant blocks with satisfactory recall and produce significantly better document ranking.

Heading-Aware Snippet Generation for Web Search

We propose new methods to generate search result snippets of web pages. To generate the snippets indicating the relevance of the pages, many existing methods extract sentences containing many query keywords. According to our observation, heading words are very often omitted from their associated blocks because readers can understand the topic of the block by reading its heading first. To score sentences considering the omitted heading words, our method counts keyword occurrences in their connected headings as well as in the sentences themselves. Finally, we evaluate and compare four methods, namely a naive method, the existing method, our method, and the combination of the existing method and our method. Our evaluation indicated that the combination generates the best snippets for the queries with clear intents and the queries containing four or more keywords.

CONTENTS

1 Preface **1**

1.1 Hierarchical Heading Structure in Documents 2

1.2 Outline of This Dissertation 3

2 Related Work **7**

2.1 Structure Extraction 7

2.1.1 Document Segmentation 7

2.1.2 Heading Structure Extraction 8

2.1.3 Other Types of Structure 9

2.2 Subtopic Mining 9

2.2.1 Subtopic Candidate Extraction 10

2.2.2 Subtopic Feature Extraction 10

2.2.3 Subtopic Ranking 11

2.2.4 Subtopic Diversification 11

2.3 Proximity Search 12

2.4 Passage Retrieval 12

2.4.1 Heading-Aware Region-based Search 13

2.4.2 Region-Based Search 14

2.4.3 Heading-aware Search 15

2.5 Snippet Generation 16

3 Extracting Logical Hierarchical Structure of HTML Documents Based on Headings **17**

3.1 Introduction 17

3.2 Logical Document Structure 21

3.2.1 Definitions of Blocks and Headings 21

3.2.2 Properties of Headings 22

3.2.3 Correspondence between Logical and Mark-up Structure 23
3.3 Structure Extraction Method 26
3.3.1 Preprocessing 26
3.3.2 Classification of Nodes by Visual Styles 27
3.3.3 Sorting Candidate-Heading Lists 28
3.3.4 Recursive Document Segmentation 30
3.4 Evaluation 33
3.4.1 Document Collection 34
3.4.2 Ground Truth and Its Representation 34
3.4.3 Evaluation Measures 36
3.4.4 Evaluation Results and Discussions 37
3.5 Summary 44

4 Subtopic Ranking Based on Hierarchical Headings 45
4.1 Introduction 45
4.2 Definition of Subtopics 46
4.3 Subtopic Ranking Methods Based on Hierarchical Heading Structure 47
4.3.1 Subtopic Scoring on a Single Page 47
4.3.2 Score Integration for Multiple Pages 50
4.3.3 Diversifying Subtopic Ranking 51
4.4 Evaluation 52
4.4.1 Evaluation Method 52
4.4.2 Evaluation Measures 53
4.4.3 Data Set 53
4.4.4 Implementation Details 54
4.4.5 Evaluation Results 55
4.4.6 Discussions 56
4.5 Summary 58

5 Heading-Aware Proximity Measure and Its Application to Web Search 59
5.1 Introduction 59
5.2 Proximity Measures 60
5.2.1 Distance 61
5.2.2 Heading-Aware Semi-Distance 61
5.3 Proximity-Aware Scoring Methods 62
5.3.1 MinDist 62

5.3.2 P6 . . . 62
5.3.3 Expanded Span . . . 63
5.3.4 Heading-Aware Methods . . . 64
5.4 Implementation Details . . . 64
5.5 Data Sets and Evaluation Measures . . . 65
5.6 Parameters and Fine Tuning with Training Data . . . 66
5.6.1 Optimization Method . . . 66
5.6.2 Optimized Parameter Values . . . 67
5.6.3 Discussions . . . 68
5.7 Evaluation with Test Data . . . 69
5.7.1 Evaluation Method . . . 69
5.7.2 Evaluation Result . . . 69
5.7.3 Discussions . . . 69
5.8 Summary . . . 70

6 Heading-Aware Block-Based Web Search 71
6.1 Introduction . . . 71
6.2 Block-Level Boolean Retrieval Methods . . . 73
6.2.1 Block-Level Boolean AND Search . . . 74
6.2.2 Removal of Child Blocks . . . 75
6.2.3 Addition of Ancestor Headings . . . 75
6.2.4 Four Proposed Block-Level Boolean Retrieval Methods . . . 76
6.3 Heading-Aware Block Ranking Method . . . 76
6.3.1 Basic Block Ranking . . . 77
6.3.2 Heading-Aware Block Ranking . . . 77
6.3.3 Parameters and Fine Tuning . . . 78
6.3.4 Ranking Combination . . . 79
6.4 Evaluation . . . 80
6.4.1 Evaluation Method . . . 80
6.4.2 Evaluation of Block-Level Boolean Retrieval . . . 81
6.4.3 Evaluation of Heading-Aware Ranking . . . 83
6.5 Summary . . . 88

7 Heading-Aware Snippet Generation for Web Search 89
7.1 Introduction . . . 89
7.2 Snippet Generation Methods . . . 90

7.2.1 Basic Snippet Generation Method 90
7.2.2 Occurrences of Heading Words in Sentences 92
7.2.3 Keyword Occurrences in Headings 94
7.2.4 Combination of Two Advanced Methods 94
7.2.5 Parameters and Fine Tuning 95
7.3 Evaluation 95
7.3.1 Evaluation Method 95
7.3.2 Data Set 96
7.3.3 Evaluation Measures 96
7.3.4 Evaluation Results and Discussions 97
7.4 Summary 99

8 Conclusions and Future Directions **101**
8.1 Conclusions 101
8.2 Future directions 103
8.2.1 Incompleteness of Our Assumptions and Observations 103
8.2.2 Data Structure and Algorithms for Hierarchical Heading Structure 103

Bibliography **105**

Publications **115**

List of Figures

1.1 Various types of structure in documents. 2
1.2 Logical hierarchical structure fills gap between paragraphs and regions. 3
1.3 Topic structure of this dissertation. 3
1.4 Data flow diagram of modern search engines. 4
1.5 Roles of our proposed methods. 5
3.1 Example web page with heading structure. 18
3.2 Example of heterogeneous search results. 19
3.3 Heading structure of document in Figure 3.1. 24
3.4 Node sequence representing block in HTML document in Figure 3.3. 25
3.5 General structure of sentence-breaking node. 26
3.6 How we compute depth of blocks associated with heading lists. 29
3.7 Detection of blocks starting from front nodes found in Figure 3.6. 31
3.8 How scan starting from front node of heading in Figure 3.4 stops. 32
4.1 Example web page with hierarchical heading structure. 46
4.2 Comparison of scoring results of page in Figure 4.1 by four scoring methods. . . 48
4.3 Example re-scoring result of page in Figure 4.1. 52
5.1 Example document with heading structure. 60
6.1 Page with hierarchical heading structure. 72
6.2 Hierarchical heading structure of page in Figure 6.1. 73
6.3 Target text of block in Figure 6.2 for each block-level Boolean AND search method. 76
7.1 Example web page with hierarchical heading structure. 90
7.2 Hierarchical heading structure in Figure 7.1. 91
7.3 Example baseline snippet for example query. 92
7.4 Example heading-aware snippet for example query. 93

LIST OF TABLES

3.1 Thresholds θ and t optimized for training data set. 33
3.2 Assignment of pages to annotators and data sets. 35
3.3 Accuracy of HEPS for training/test data sets. 38
3.4 Accuracy of HEPS for test set by each annotator. 38
3.5 Comparison with naive method and existing methods. 39
3.6 Accuracy of naive methods and HEPS variations. 41
3.7 Accuracy of relationship extraction with HEPS. 43
4.1 D-nDCG score comparison with query completion by Google. 55
4.2 D-nDCG score comparison with query completion by Yahoo. 56
4.3 Comparison with our merged baseline result. 56
5.1 Optimized parameter values of our $hasd$. 67
5.2 Optimized values of other parameters. 67
5.3 Comparison of average evaluation scores. 69
6.1 Comparison of our block-level Boolean retrieval methods. 82
6.2 Values of BM25F parameters optimized for each method and training queries. . . 85
6.3 Comparison of mean nDCG for test queries. 86
6.4 Comparison with top-3 category B runs for TREC 2011 web track adhoc task. . . 87
7.1 Boost for occurrence of words of each type in each field. 95
7.2 Comparison of average evaluation scores of four methods. 97
7.3 Average evaluation scores of four methods for each type of queries. 97
7.4 Average GM scores of four methods and query length excluding stopwords. . . . 98
7.5 Median amount of required time in sec. to check snippets of 20 pages for one query. 99
7.6 Comparison of average evaluation scores of four participants. 99

CHAPTER 1

PREFACE

Autonomy and distribution of the World Wide Web have rapidly grown the web. Anyone can distribute information in free formats on the web, and web pages are one of such formats of web information. Text contents in web pages are an important information source. Moreover, the web itself is composed of hyperlinks between web pages, and therefore, web pages are recently becoming more important as hubs of multimedia contents, e.g., images, musics, and movies.

Because of the autonomy and distribution, we can mine huge amount of opinions written by millions of authors from the web. However, because of the free formats, it is difficult to mine the opinions automatically. For example, in the hypertext mark-up language (HTML), which is a major web page description language, both elements for logical meanings and for visual styles are defined. For concrete example, many web browsers render `em` (logically emphasized) elements and `i` (visually italicized) elements in the same visual style. Because the main concern of non-expert page authors is visual style of web pages, these elements are very often misused. Like this concrete example, logical structure of most web pages is not obvious. Therefore, automatic structuring of web information, structure-aware page scoring, and structure-aware page summarization have been all interesting research problems.

Web pages compose and contain various types of structures, e.g., hyperlink structure, logical structure, sentence structure, and so on. The HITS and PageRank [57, 80] are the most well-known approaches which use hyperlink structure. They use hyperlinks as popularity voting for scoring the pages. Many natural language processing tasks are based on sentence structure. In this dissertation, the remaining *logical structure* is a term for denoting general structure in web pages composed of larger units than ones sentence structure are composed of. Based on the logical structure, extraction of content bodies, lists, and tables in web pages have been studied.

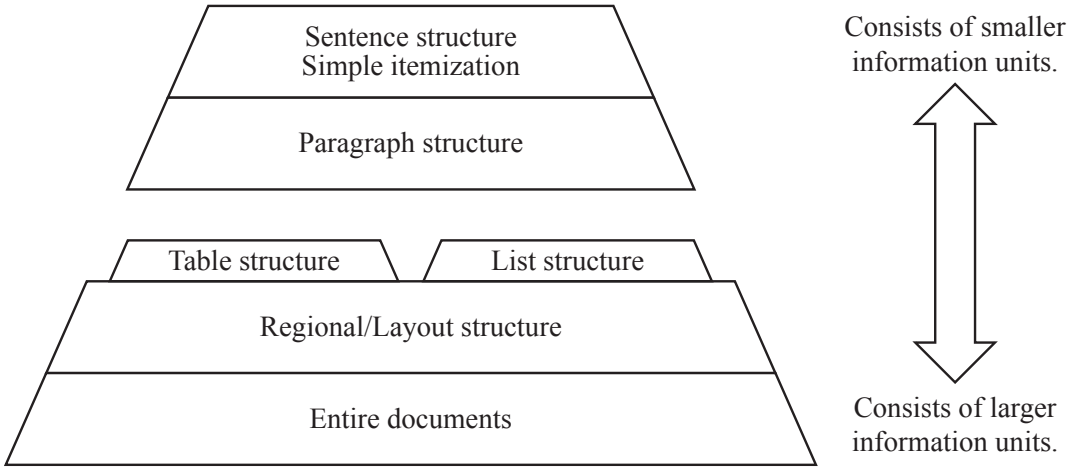


Figure 1.1: Various types of structure in documents.

1.1 Hierarchical Heading Structure in Documents

Not only web pages, but general documents include various types of structure. Sentence structure, paragraph structure, table/list structure, and regional/layout structure are all such structures. In this order, each type of structure contains less information in its information units (see Figure 1.1). Paragraph structure is used for wrapping up multiple sentences in a topic. Regional/layout structure is used for two-dimensional layout of regions like headers, menus, content bodies, and so on. In these types of structure, table/list structure are special structure. More generally, many documents contain logical hierarchical structure. Logical hierarchical structure fills the gap between paragraph structure and regional/layout structure and generalizes table/list structure (see Figure 1.2). Logical hierarchical structure is used for wrapping up multiple paragraphs in a topic, for splitting content bodies into finer regions, and for organizing the finer regions.

In this dissertation, logical hierarchical structure is considered to be mostly expressed by hierarchical heading structure. In other words, we consider that humans detect logical hierarchical structure mainly based on headings. This is the first and main reason why hierarchical heading structure and its components are important enough to be our research target.

In our definitions, each *heading* describes the topic of a part of a document and is easily distinguished from other components by humans. Each *block* is such a partial document with its heading. *Hierarchical heading structure* is composed of nested blocks. Our defined hierarchical heading structure generalizes table/list structure, and it is the second reason why the structure and its components are important. Table structure is two-levels special heading structure whose upper-level headings are row headers, and lower-level headings, which are common to all rows,

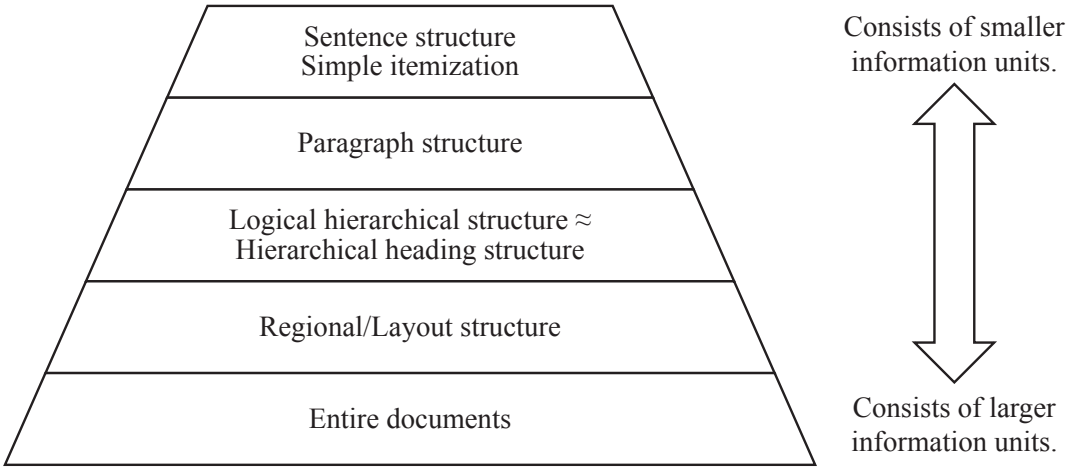


Figure 1.2: Logical hierarchical structure fills gap between paragraph structure and regional/layout structure.

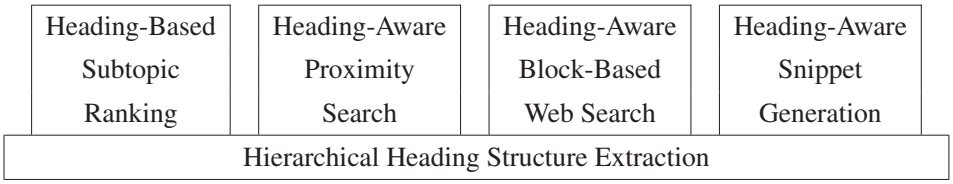


Figure 1.3: Topic structure of this dissertation.

are column headers. On the other hand, list structure is one-level heading structure in which structure inside the blocks are homogeneous.

1.2 Outline of This Dissertation

Based on the above discussion, in this dissertation, we focus on hierarchical heading structure, which is one of the most prevalent logical structure, in web pages, which is one of the largest collections of digital documents. In this dissertation, we survey related work in the next chapter (Chapter 2) then propose a method for extracting hierarchical heading structure in web pages, and also propose four methods that utilize the information on the extracted structure. All these four methods are based on our extraction method (see Figure 1.3). Also, all our methods are for upgrading modern search engines (see Figure 1.4, which is a typical data flow chart of such engines) to heading-aware search engines (see Figure 1.5). In this dissertation, we explain our methods in order of the data flow.

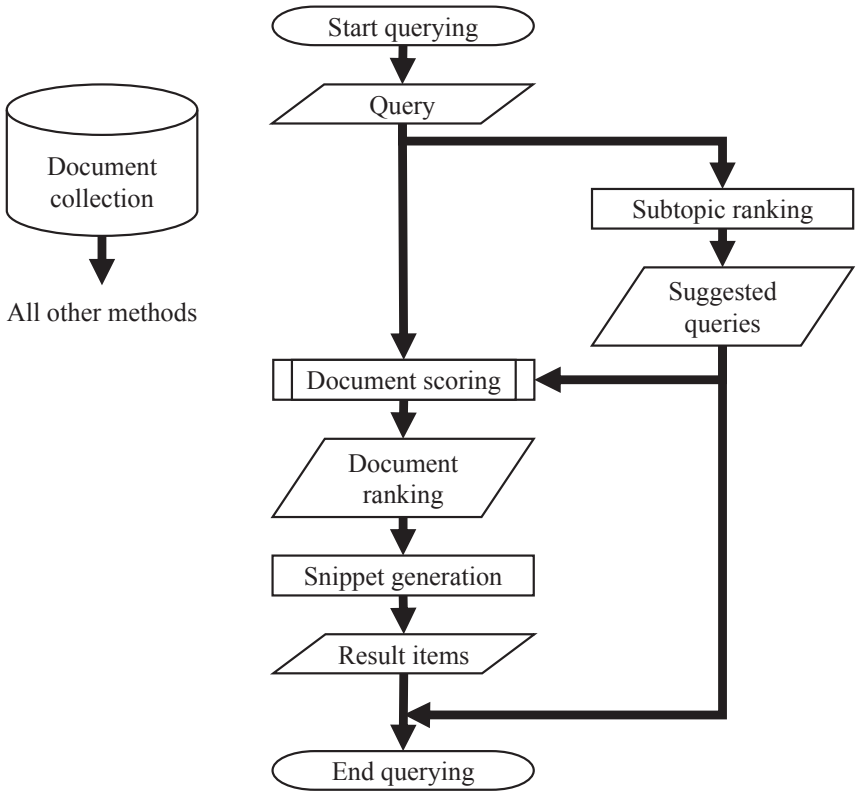


Figure 1.4: Data flow diagram of modern search engines.

Extracting Logical Hierarchical Structure of HTML Documents Based on Headings (Chapter 3)

We propose a method for extracting hierarchical heading structure in web pages. Human readers easily understand the structure by exploiting the following properties of headings: (1) headings appear at the beginning of the corresponding blocks, (2) headings are given prominent visual styles, (3) headings of the same level share the same visual style, and (4) headings of higher levels are given more prominent visual styles. Our method also exploits these properties. Our experiment shows that our method outperforms existing methods.

Subtopic Ranking Based on Hierarchical Headings (Chapter 4)

We propose methods for generating rankings of subtopics of keyword queries. Subtopics are described by query-like subtopic strings and are useful for search result diversification and query suggestion. Hierarchical headings in documents reflect the hierarchical topics referred to in the documents. Our methods score subtopic candidates based on matching between them and hierarchical headings in documents. They give higher scores to candidates matching hierarchical

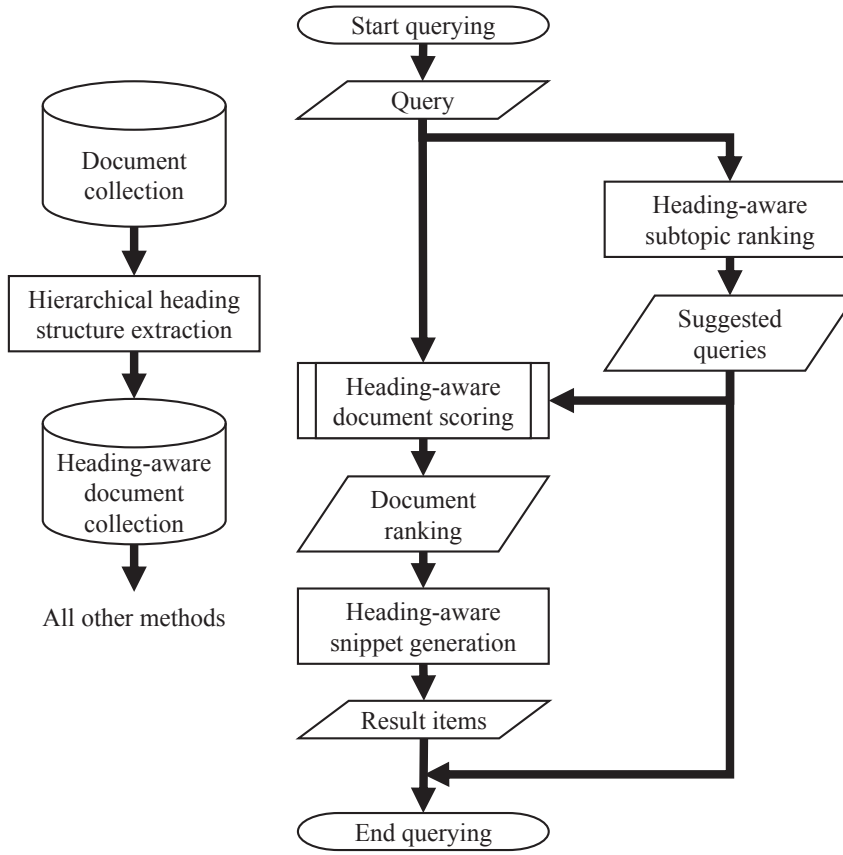


Figure 1.5: Roles of our proposed methods.

headings associated to more contents. Our methods generated significantly better subtopic rankings than query suggestion by major commercial search engines.

Heading-Aware Proximity Measure and Its Application to Web Search (Chapter 5)

Proximity of query keyword occurrences is one important evidence useful for effective document scoring. If a query keyword occurs close to another in a document, it suggests strong relevance of the document to the query. Our defined hierarchical heading structure affects logical proximity. For example, occurrences in headings describe the topics of the entire corresponding blocks, and the occurrences are strongly relevant to any term occurrences in the blocks regardless of the distance between them. Based on such observations, we developed a heading-aware proximity measure, applied the measure to three existing proximity-aware document scoring functions, and evaluated the existing and modified functions on the well-known Text Retrieval Conference (TREC) data sets.

Heading-Aware Block-Based Web Search (Chapter 6)

Because a heading is a concise description of the topic of its associated block, query term occurrences in a heading indicate high relevance of the block to the query intent. In a hierarchical heading structure, headings of ancestor blocks describe the topics of all their descendant blocks, although the ancestor headings may be located far from the descendant blocks. Based on these facts, we developed search methods which retrieves and scores blocks by taking into account their ancestor headings as well as their contents. We evaluated our system by using TREC data sets. The result indicates that our system can eliminate many irrelevant blocks with satisfactory recall and produce significantly better document ranking.

Heading-Aware Snippet Generation for Web Search (Chapter 7)

We propose methods to generate search result snippets of web pages. To generate the snippets indicating the relevance of the pages, many existing methods extract sentences containing many query keywords. According to our observation, heading words are very often omitted from their associated blocks because readers can understand the topic of the block by reading its heading first. To score sentences considering the omitted heading words, our method counts keyword occurrences in their relevant headings as well as in the sentences themselves. Finally, we evaluate and compare four methods, namely a naive method, the existing method, our method, and the combination of the existing method and our method. Our evaluation indicated that the combination generates the best snippets for the queries with clear intents and the queries containing four or more keywords.

Chapter 8 concludes this dissertation and indicates two future directions of our research.

CHAPTER 2

RELATED WORK

In this chapter, we survey related work of our methods, namely structure extraction, subtopic mining, proximity search, passage retrieval, and snippet generation, in this order.

2.1 Structure Extraction

First of all, we survey related work of structure extraction from documents.

2.1.1 Document Segmentation

Hierarchical structure extraction is a kind of document segmentation. There have been several studies on Web page segmentation. Cai et al. [16] proposed a method called VIPS, which recursively divides a page mainly based on the margins between regions in the rendered page. Their method works well for typical top-level layout structure of pages composed of main bodies, menu blocks, sidebars, and so on, but it does not work well for the hierarchical heading structure inside main text bodies. For example, it cannot detect blocks delimited by headings in bold fonts without any special margins.

Kohlschutter and Nejdl [58] proposed a method based on text-density. Their method can distinguish main bodies filled with full sentences from menu blocks including only short phrases, and also from advertisements mostly composed of images. Hattori et al. [48] proposed a method that divides a page at points where the HTML mark-up structure largely changes. Their method also works well only for the top-level layout structure of Web pages.

Chakrabarti et al. [20] formulated the page segmentation problem as an optimization problem on weighted graphs and showed how to solve the problem by learning weights from manually labeled data. Their method is also focused only on the top-level structure of pages, and it is not

clear how to adapt their labeling and learning schemes to nested structure in the main bodies.

HTML5 has tags for marking-up hierarchical block structure. However, methods that rely on them cannot be applied to the huge asset of existing Web pages that are not in HTML5 format. In addition, given that the ratio of headings described by proper tags is not high, we cannot expect that most Web pages in future would be marked-up with proper HTML5 tags for the structure.

Debnath et al. [36] proposed a method of detecting main blocks in pages, but they focused on how to choose important blocks, and page segmentation is simply done by a predefined set of HTML tags; they first segment a page by TABLE tags then by TR tags, and so on. Lin and Ho [62] also proposed a method of estimating the importance of blocks based on tabular tags. Song et al. [92] used VIPS [16], explained above, for main block detection. Gupta et al. [45] also proposed a method of main block detection, and Keller and Nussbaumer [55] proposed a method of extracting side menus.

In summary, these page segmentation methods and main block detection methods focus on the top-level page layout, while we focus on structure inside the main body. These methods and our method are, therefore, complementary to each other. Another difference between these methods and ours is that our method extracts not only hierarchical blocks but also headings associated with them.

2.1.2 Heading Structure Extraction

The SphereSearch engine [44] converts heterogeneous HTML documents into XML documents for unified ranking, and their conversion includes the detection of headings and their corresponding blocks based on simple heuristic rules. Their rules, however, depend on specific tags, such as H1 and B, and can work only for well-structured HTML documents described by these tags.

El-shayeb et al. [40] proposed a method that extracts hierarchical document structure by using the same information as ours: visually prominent headings. Their method, however, examines each node separately and determine if its visual style is likely to be a heading. They do not use information on other nodes sharing the same visual style, which is useful according to our evaluation results. Tatsumi and Asahi [99] proposed a method that detects headings based on their visual styles, but they also examine each node separately, and they do not extract hierarchical relationship among them.

There are also heading extraction methods based on machine learning [13, 77, 82, 89]. These learning-based methods, however, can detect a heading only when the training data set includes positive examples of headings sharing the same features (e.g., visual style and partial tree structure). In [77] and [89], the evaluations were based on cross-validation, in which the training and test data sets may share headings in the same format from the same page. In [82], their method

was evaluated on a data set consisting of top-ranked pages by Google, which must contain many pages that are from the same domain (e.g., Wikipedia) and have the same format.

There have also been research on structure extraction from PDF documents. Gao et al. [43] has proposed a method of extracting hierarchical structure from the table of contents (ToC) of PDF documents. If a document does not have a ToC, they apply the same method to the list of headings extracted from the main body [42]. Their method, however, assumes that the level of a heading is either one-level lower, the same, or higher than the level of the immediately preceding heading. It holds for headings of sections, but does not hold for headings in a wider sense. For example, an enumerated list with headings appears at any level in section structure. If a document includes at least one such heading, their method fails.

2.1.3 Other Types of Structure

There has been extensive research on the extraction of structured data, such as lists and tables, from Web pages [1, 5, 70, 90, 117] and from plain text [30]. All these methods exploit repetitive occurrences of tree patterns, tag sequences, word patterns, or visual features. This strategy, however, cannot extract a list of items with completely heterogeneous structures. Our method can extract such a heterogeneous list by detecting its homogeneous headings.

Anjewierden [4] has also proposed a method of extracting logical structure from PDF documents. Their approach is to define domain-specific grammars for parsing documents. It is not applicable to a general Web corpus including diverse structure and design.

Chao and Fan [21] proposed a method of extracting text blocks from PDF documents by merging neighboring lines with similar styles. Their method, however, extracts disjoint blocks without hierarchical relationship.

2.2 Subtopic Mining

Generally, a term *topic* has two sense in informatics [49]. One is an implicit topic represented by a (fuzzy) set of terms [50, 51], and the other is an explicit topic represented by a short string like a keyword query. Our research target is explicit topic. More precisely, we focus on subtopics of the topics behind the keyword queries input by users.

For mining such subtopics, four elemental technologies are required. They are namely subtopic candidate extraction, extraction of their features, and subtopic ranking and diversification based on the features. We survey related work on these technologies in this order.

2.2.1 Subtopic Candidate Extraction

This step is not the main topic of this dissertation. However, we briefly survey related work on this step for reference.

Query recommendation/suggestion/completion by commercial search engines generates many related queries of the original queries. They are very popular resources of subtopic candidate strings [67, 103, 104, 106, 111, 112, 115], and therefore, the snapshots of them for the NTCIR-10 INTENT-2 task [88] is publicly available. We also adopted them as baseline subtopics. Google Insights and Google keywords generator are similar services [112]. Raw query logs of search engines [11, 67, 104, 106, 111] should be also useful.

Disambiguation pages in Wikipedia contain multiple subtopics of many ambiguous article titles of Wikipedia, and are very well-organized by humans. Therefore, they are also a very popular resource of subtopic candidate strings [67, 106, 106, 111, 111, 112, 115]. Redirect pages and tables of contents in Wikipedia should also be useful [111].

Of course, search result documents themselves can be a resource of subtopic candidate strings. Methods based on frequently occurring words [79, 105, 106, 113, 119, 120], words frequently co-occurring with query keywords [108], pseudo-relevance feedback [11], syntactic patterns [56], search result summaries [112] are proposed.

Titles [79, 113], anchor texts of in-links [49, 112], and explicitly tagged top-level headings (H1 nodes) of HTML documents [112] all describe the topics of the entire documents. Therefore, they may be important as subtopic candidate strings. Their idea is similar to ours, but they do not use detailed hierarchical heading structure. In addition, we use it for ranking candidate subtopic strings in this dissertation, not for extracting the candidates.

The QDMiner system extracts *query dimensions* each of which refers to one important aspect of the original query [39]. The system is based on list extraction from web pages. Their idea of query dimension is highly relevant to the idea of subtopic, and therefore some existing methods extract them as components of subtopic candidate strings [8, 104].

2.2.2 Subtopic Feature Extraction

Like most existing document ranking methods, many existing methods are based on term frequency (TF), i.e., the number of occurrences of subtopic strings or their component terms, or document frequency (DF), i.e., the number of documents that contain subtopic strings or their component terms [34, 56, 108, 113, 119]. The occurrences in some types of document metadata, e.g., document titles, anchor text of in-links, and top-level headings, should be more important than other occurrences [112, 113].

Similarity between subtopic candidate strings and their search result documents or their original queries is a popular feature [34, 67, 74, 119].

The document coverage of a subtopic candidate string is the weighted summation of the scores of documents that both the string and its original query retrieved [56], and the distinctness entropy of subtopic candidate strings measures the distinctness among the document sets that the strings retrieved [56, 116].

The SEM group at NTCIR-10 used the co-occurrence of subtopic candidate strings in query logs and the edit distance between the strings and their original queries [104].

Query-independent features like readability of candidates should also be useful [104, 108].

2.2.3 Subtopic Ranking

Subtopic ranking is essential for filtering out noises and for sorting subtopic strings in their intent probability. The simplest way is to sort them in order of linear combination of features. However, like document ranking, more sophisticated functions, e.g., TFIDF (TF over DF) and BM25 [87], are also used [56, 104, 106, 108].

Many methods assign different weights for different sources of subtopic candidate strings [67, 112]. For example, the THUIR group at NTCIR-11 assigned the weights of 0.75 for Google keywords generator, 0.15 for Google insights, and 0.05 for query completion/suggestion by commercial search engines [112].

Ullah and Aono proposed a method that represents each subtopic candidate string by its feature vector then score them by their cosine similarity with the mean vector [103].

It is notable that the THUSAM group at NTCIR-12 adopted a variant of learning-to-rank methods that are state-of-the-arts methods for document ranking [67].

2.2.4 Subtopic Diversification

One important application of subtopic mining methods is search result diversification. Therefore, diversity of subtopic rankings is also important.

One promising way to diversify subtopic rankings is subtopic clustering and extraction of the median subtopics of each cluster [105, 106, 111–113, 115]. The K-means [113], affinity propagation [111, 115], a variant of K-medoids [112] algorithms are used.

The THCIB group at NTCIR-10 clustered implicit topics by the affinity propagation algorithm, then assigned explicit topics to each cluster by Latent Dirichlet Allocation [107].

The Hierarchical InfoSimba-based Global K-means (HISGK-means) algorithm, clusters search result snippets then labels each cluster [38, 73]. InfoSimba is a term co-occurrence based similarity measure between snippets, and the HISGK-means algorithm recursively clusters snip-

pets based on the measure and the global K-means algorithm. Each label is obtained as the centroid of a cluster.

Recently, some methods adopted word embedding models [67, 74]. In word embedding models, we can *subtract* subtopic candidate strings from their original query. Based on this idea, the HULTECH group at NTCIR-11 recursively subtracted subtopic candidate strings from their original query then compared the difference and the remaining subtopic candidate strings every time they adopt the subtopic candidate string with the best score [74].

The maximal marginal relevance (MMR) framework also concatenate items into rankings one-by-one [19]. In each iteration, MMR selects the item with the best balance of the score and dissimilarity to the already ranked items. Of course, it is useful for subtopic diversification [103].

As explained above, no existing method scores or diversifies subtopic candidate strings based on detailed logical hierarchical structure in documents, e.g. hierarchical heading structure, like our method.

2.3 Proximity Search

One simple way to score documents considering proximity of query keyword occurrences is to count frequency of n-grams ($1 < n$) that contain multiple query keywords as well as frequency of query keywords [14, 85]. However, because these methods does not scale with n, it is difficult for these methods to consider proximity of distant occurrences.

Tao and Zhai [98] proposed five proximity-aware document scoring functions. They concluded that *MinDist*, which takes into account only the closest occurrence pair of two different query keywords, is the most effective for modern search engines. Cummins and O'Riordan [31] proposed 10 basic proximity functions and combined 12 basic proximity functions by using Genetic Programming. They introduced three obtained functions. Tian et al. proposed another function, first-appearance term distance, which is distance between the first appearances of two query keywords [100]. Song et al. [93] proposed an efficient method for segmenting a document into *expanded spans*, and a simple proximity function based on the numbers of keyword occurrences and all term occurrences in spans. Svore et al. [95] proposed other scoring functions for expanded spans and discussed their application to learning to rank. To the best of our knowledge, no study has proposed proximity measures that consider properties of heading structure.

2.4 Passage Retrieval

Our work is characterized by its heading-aware and block-based methods. Therefore, it is the most related to the field of heading-aware region-based search. Both region-based and heading-aware search are also important. In this section, we briefly survey these areas in this order.

2.4.1 Heading-Aware Region-based Search

There exists a few heading-aware and region-based search methods.

There are two heading-aware and query-biased web page summarization methods [81, 101]. Both score sentences and extract them in descending order of their scores. We can regard these methods as sentence-based search. These methods are based on either of the following observations on heading structure: important sentences contain many words included in headings [81] or headings themselves are important sentences [101]. However, headings are not just sentences, and the number of heading word occurrences does not necessarily reflect the importance of the corresponding block for keyword-based search. For example, in an example page, the word "reviews" may occur only in the page title; however, we understand that the entire page and all the other blocks in the page are about reviews.

Proximity retrieval is another approach to improve search and is not exactly the same as passage retrieval [72]. Lu et al.'s proximity-aware method is for entity retrieval from web pages [65]. Because entities are very often described by regions, we can regard their work as region-based search. Their method uses page titles and explicitly tagged headings (H1 to H6 and TH HTML elements). Moreover, it adds anchor text of in-links and page categories extracted from site menus to titles of pages. Their method use different weights for headings of different logical distance from the entity description. However, their method cannot assign higher weight to more distant headings. According to our experimental results, such setting is practical.

XML elements are a type of regions in XML documents [12, 46, 47]. We can simply measure importance of the elements by their TFIDF score [46]. For XML element retrieval, if structural constraints are specified by users, we can replace IDF values with *inverse path frequency* (IPF) values, i.e., IDF values among only the elements satisfying the constraints. Broschart and Schenkel proposed a proximity-aware scoring method for XML elements [12]. It regards that pairs of term occurrences in different elements are logically distant than pairs in an element. The size of this distance penalty is different for different element names. For example, pairs in different sections are more distant than pairs in different paragraphs. It may be applicable to score pairs in a heading and its associated block higher. However, their method also cannot weight term occurrences in more distant headings higher.

The BM25E function [66] is also for XML element scoring. The authors evaluated their method on XML collections of academic articles. Their method takes into account article titles, abstracts, and section titles. Their section titles are very similar to our headings. However, they assigned the same weight to all self-and-ancestor headings. According to our experimental results, they are quite different depending on their logical distance from the block.

Beigbeder et al.'s method assigns higher scores to XML elements of certain tag names that frequently contain relevant information [9]. Headings may have such tag names, however it does not consider other features of headings like heavier weight for scoring than other components.

In the field of XML retrieval, it is also common to score elements considering the contents of their hierarchical ancestor elements as well as their own contents [3]. The *contextualization* [6, 7] and *shrinkage* [75, 76] methods perform such scoring. However, most XML element retrieval methods do not distinguish headings from other components of elements. This distinction is important because ancestor elements that represent headings usually have larger influence on their descendant elements than the other kinds of ancestor elements. In the field of video retrieval, an early work considered the hierarchy of video clips and metadata contextualization [78].

2.4.2 Region-Based Search

The first problem in effective region-based search, or passage retrieval, of web pages is to extract regional structure reflecting topic structure. There are many region extraction methods and some have already been applied to passage retrieval [91].

Early passage retrieval methods were based on *windows*, i.e., fixed-length overlapped substrings of a document, for scoring documents of varied length correctly [18,53]. However, today's basic scoring functions are already designed to score documents or passages of varied length correctly [86, 87]. Therefore, more sophisticated passage retrieval methods have been studied.

The vision-based page segmentation method (VIPS) [16] is one of the most well-known extraction methods for hierarchical regions. VIPS detects the margins in web pages then segments the pages into blocks based on the margins. To extract the hierarchy of blocks, VIPS measures the weight of the margins mainly based on their width then recursively merges the blocks split by the lightest margin. The same authors also proposed a passage retrieval method for web pages [17]. Their method assigns scores to blocks and carries out page-level scoring focusing only on the block with the best score in the page. Our basic idea is similar to theirs. However, while they use VIPS and windows of 200 words to extract blocks, we use our proposed HEPS method to extract headings as well as their associated blocks. This difference allows us to score blocks by considering the headings of their ancestor blocks for better ranking.

Song et al. also adopted VIPS and proposed a method for estimating block importance by machine learning [92]. The importance of a block is estimated based on the topic correlation between the block and the entire page; therefore, it is independent from queries. In contrast, our proposed retrieval and ranking methods determine the relevance between the intent behind a query and the topic of a block.

Denoyer et al. regarded a web page as a string of regions and built Hidden Markov Model to

generate the string [37]. Bendersky and Kurland [10] proposed *passage language model* which is the weighted summation of three language models based on a passage, the document including the passage, and the document collection including the document. However, for heading-aware search, it is important to regard a page as hierarchy of regions because headings represents the topics of their corresponding blocks and all the blocks' hierarchical descendant blocks.

2.4.3 Heading-aware Search

As discussed above, heading-aware web passage/proximity retrieval has not been studied sufficiently. On the contrary, heading-aware web page retrieval has been frequently studied. Especially, many methods take into account headings of entire pages, i.e., document titles. The BM25F function [86] is a variant of the BM25 function [87], a widely used document scoring function. The BM25F function was designed to score documents composed of multiple fields, such as titles, anchor text of in-links, and body text. In their paper, they assigned a far greater weight to keyword occurrences in document titles than in body text to gain the best precision. This fact supports the importance of headings. Note that most web pages contain their titles as explicit metadata.

Some other methods classify term occurrences in web pages into some classes including heading classes.

The early scoring function for HTML documents proposed by Cutler et al. uses six classes of tag names that include H1 or H2 and H3 to H6 heading tag classes [32]. A few other early methods extracts term occurrences including explicitly tagged headings even if they are in other documents that have links to the original document [71,96]. However, these methods are designed to score documents and consider neither scoring of regions or their ancestor headings.

De Moura et al. proposed block-based modification of ordinary term frequency (TF) for document scoring [35]. They first extract blocks by using VIPS, and classify them by the domain names of their source web sites and their tag paths. The tag path of a block is the sequence of tag names along the path from the root element to the block. After that, they calculate the value of their *block weight function*, bw, and weight the TF in the blocks by their bw score. Headings with the same tag path in the same web site should be regarded as a class in their method. However, their method does not distinguish such a heading class from other classes.

Dai et al. proposed a learning to rank method for ranking web pages considering the freshness of information in the pages [33]. It also consider headings in the pages as well as body text, title, and anchor text of in-links. However, it does not consider the correspondence between headings and blocks although headings are especially important for scoring the corresponding blocks, not entire pages. Note that they did not discuss detection of headings in web pages.

2.5 Snippet Generation

Generally, snippet generation methods uses some types of important words and document fragments. Almost all methods count the occurrences of query keywords. Additionally, some methods use pseudo relevance feedback to expand queries and to obtain more keywords [60, 109]. Frequently occurring words in a page may also be important for the page [81,97,101,109]. More frequently or less frequently occurring words in a corpus, not in a page, may also be important for some types of queries [97].

The first sentence of a paragraph [81] or the first paragraph of a page [101] may be important.

As listed above, most summarization methods do not focus on headings and heading words.

Two heading-aware summarization methods exist. The method by Tombros and Sanderson regards headings as important sentences and assigns higher scores to headings than to other sentences [101]. However, headings represents the topics of their corresponding blocks and are also important for scoring other sentences in the blocks. The method by Pembe and Güngör counts heading-word occurrences in sentences to score the sentences [81]. However, their method does not take the omission of heading words into account although heading words are very often omitted because the words are prominently indicated as headings themselves.

Some snippet generation methods focus on the locations of the occurrences of query keywords. Some count the occurrences in document titles, which are a type of headings [101, 109]. The method by Zhang et al. distinguishes attribute names, that are also a type of headings [118]. These methods, however, do not count query keyword occurrences in general headings.

Outside the field of web search, many snippet generation methods for XML documents are based on XML element retrieval [60, 109], and many XML element retrieval methods take hierarchical ancestors of elements into account [3, 7]. However, unlike our methods, most XML element retrieval methods do not distinguish headings from other components of elements. The BM25E function for element scoring distinguish headings from other components [66]. However, application of the function to snippet generation has not been discussed.

CHAPTER 3

EXTRACTING LOGICAL HIERARCHICAL STRUCTURE OF HTML DOCUMENTS BASED ON HEADINGS

3.1 Introduction

Since the wide-spread of the Internet, a huge amount of data have been accumulated on the web in the form of HTML documents. To take full advantage of this huge valuable asset, information extraction from these web pages has become an important research topic. There has also been extensive research on querying, ranking, summarizing, and efficiently browsing these web pages.

Because HTML documents are not plain text data, automatic understanding of the structure within them is important for improving these HTML processing. There has been much research on the extraction of various types of structure in web pages, such as list or table structure [5, 70, 90, 117] and layout structure consisting of a main body [45, 62, 92], side menus [55], and so on.

However, there remains one of the most prevalent types of structure in web pages: *logical hierarchical structure within their main bodies*. Logical hierarchical structure is important for correctly understanding documents. For example, suppose we want to extract temporal information from some corpus, which includes a web page shown in Figure 3.1. If we ignore the hierarchical structure of this document and simply regard the document as a sequence of words, the line for July 2010 can be mistaken as describing events in July 2012 because the line is equally close to the words 2010 and 2012. For the same reason, the hierarchical structure is also important when we rank this page for the query "2012 Jul construction." These are examples of

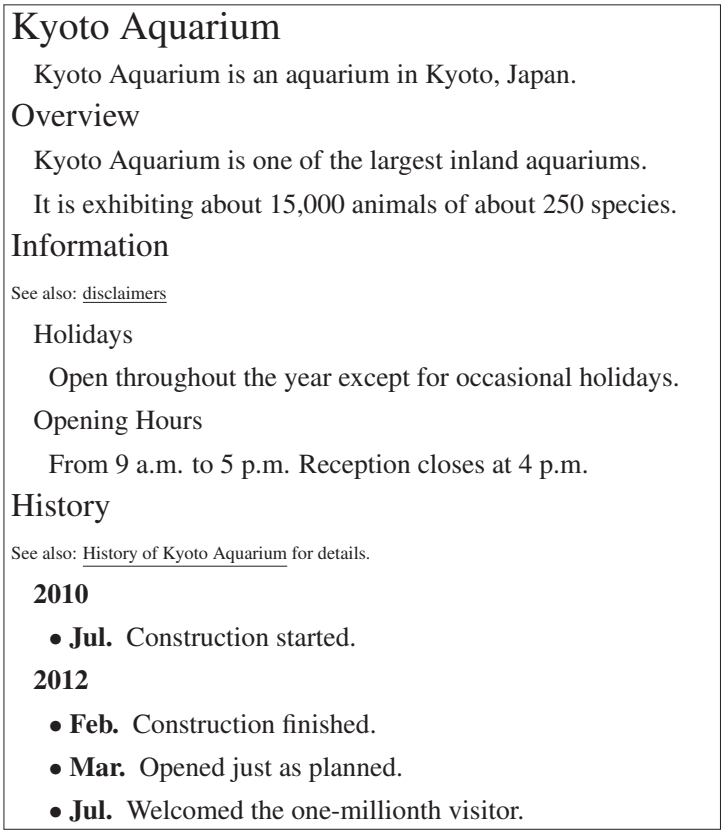


Figure 3.1: Example web page with heading structure.

HTML processing tasks where recognition of hierarchical document structure is important.

However, there has not been sufficient research on the extraction of logical hierarchical structure of HTML documents. The problem may not seem difficult as HTML documents include explicit nested mark-ups. These mark-ups, however, often describe physical layouts or visual appearances of text data, and their nested structure does not necessarily coincide with logical hierarchical structure. For example, the mark-up structure of a HTML document consisting of sections and subsections usually has no hierarchical structure corresponding to the inclusion relationship between sections and subsections. It instead includes many nested tags corresponding to visual decoration of text fragments, and also includes even wrong or abused usage of tags. Because of this discrepancy between mark-up structure and logical hierarchical structure in HTML, it is not trivial how to automatically extract the latter. Extraction of logical hierarchical structure is thus a crucial yet non-trivial step commonly required in various HTML processing tasks.

On the other hand, human readers easily recognize logical hierarchical structure of these

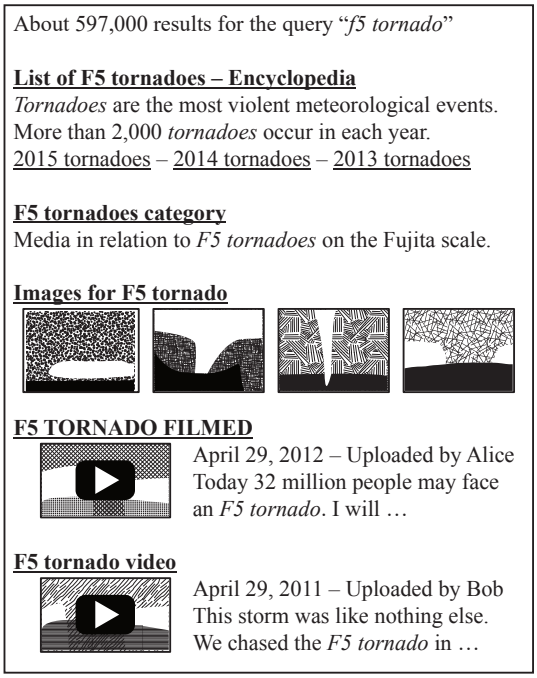


Figure 3.2: Example of heterogeneous search results.

HTML documents. The key information used by human readers is hierarchical headings in the documents. For example, a search result shown in Figure 3.2 includes three types of items: ordinary web pages, images, and videos. Modern search results may also include news, maps, recipes, web page with sitelinks, and so on, and they have heterogeneous structures. Human readers, however, can parse such a completely heterogeneous list simply because the items in the list have only one common component: the headings. Human readers can also recognize the hierarchical structure of sections and subsections in the document in Figure 3.1 because of their hierarchical headings.

Based on this observation, we develop a method that extracts logical hierarchical structure of HTML documents by exploiting hierarchical headings in them. This approach is expected to work even if the document has many noisy or abused tags as long as it is appropriately designed by the author so that human readers can recognize its hierarchical structure by its headings.

However, identification of hierarchical headings in HTML documents is also a non-trivial task. According to our survey on our data set (which will be explained in Section 3.4.1), only less than 1/3 of headings are marked up with proper HTML tags for headings, namely H1 to H6 and DT (definition term). The ratio may depend on how we define "headings." Our definition (explained in Section 3.2) is relatively loose (i.e., defined in a wider sense), and if we adopt a

tighter definition, the ratio may be higher than 1/3. Our loose definition is, however, preferred for our purpose, i.e., heading-based document segmentation, and as long as this usage of headings is concerned, the naive method based on tag names can extract only 1/3 of what we want.

In addition, H1–H6 or DT tags do not always represent headings. According to our same survey, only about 2/3 of these tags really represent headings even in our loose definition. There are many pages where these tags are used for marking up some metadata (e.g., author names and timestamps). There are also many pages where text nodes marked up with these tags are not visually prominent, which means they are not recognized as headings by human readers. We suspect the main reason of such usage of heading tags is search engine optimization (SEO); some search engines assign heavier weights to text between heading tags. There are also pages where heading tags are used to merely change visual styles.

Our method identifies hierarchical headings based on our assumptions on the visual design of headings. When users read a document with headings, they use each heading to determine whether its associated block is relevant to their needs. If they determine that the block is irrelevant, they skip it and jump to the next heading of the same level. If they determine that it is relevant, they look into it, and if it includes lower-level headings, they recursively repeat the same process in order to further narrow down blocks to read.

In order to help this process, the authors design headings in a special way: (1) they insert headings at the beginning of the corresponding blocks, (2) they give headings prominent visual styles, (3) headings of the same level are given the same visual style, and (4) headings at higher levels are given more prominent visual styles than headings at lower levels. We explain the details in Section 3.2.2. In this chapter, we propose a method that extracts hierarchical headings in HTML documents based on these properties of the headings, and extracts logical hierarchical structure based on the extracted headings. Our experiment shows that our method outperforms existing web page segmentation methods.

In addition, our method can identify not only hierarchical blocks but also their headings. Heading extraction is important in its own right because they are important for understanding the information in their associated blocks. For example, we developed a block-based web search system upon our block extraction method, and our experiment shows that we can significantly improve the ranking of blocks by complementing blocks with words in their ancestor headings. It is because words that have already appeared in some ancestor headings are often omitted in a block. The details are beyond the scope of this chapter, and will be reported in Chapter 6.

Heading extraction is also useful for document summarization and intra-document browsing support [15, 110]. For example, in the Accordion Summarization method [15], each block in a web page is first represented only by its first line, and the whole block is shown upon the users'

click on it. With our heading extraction method, we can show its heading instead of its first line. We also propose our heading-aware generation methods for search result snippets in Chapter 7.

Our method relies on only a few properties specific to HTML, and it can easily be adapted to other types of marked-up documents as long as their headings are designed on the same principle. We use the following HTML-specific knowledge: the knowledge on which CSS properties affect visual styles of HTML nodes and the knowledge that IMG elements represent images.

The remainder of this chapter is organized as follows. In Section 3.2, we give our definitions of blocks and headings, and explain our assumptions on properties and structure of them. We then explain our structure extraction method in Section 3.3. In Section 3.4, we explain the result of our experiment. Finally, we conclude this chapter in Section 3.5.

3.2 Logical Document Structure

In this section, we give our definitions of blocks and headings. We then explain our assumptions on properties of headings, and we also discuss the correspondence between logical hierarchical structures and nested mark-up structure in HTML documents.

3.2.1 Definitions of Blocks and Headings

We first need to define blocks in order to clarify what we want to extract, but there is no consensus on what are logical blocks in documents. For example, a sentence and paragraph should or should not be a block depending on the application. Our main applications are information extraction and information retrieval. In these applications, we want to extract a block that has different topic from its neighboring blocks. In other words, we want to divide a document at points where the topic changes [17].

For example, in Figure 3.1, the lines for 2010 and those for 2012 have different topics: year 2010 and 2012. Therefore, they should be segmented into different blocks so that the lines for 2010 July would not be mistaken as describing 2012 July.

The definition above, however, is still ambiguous. Topics have hierarchical subtopics, and it is unclear what level of topic change it means. In order to make it less subjective, we only consider topic changes that are substantial enough that the author inserts a heading to explicitly describe a new topic. Based on this discussion, we define logical blocks in documents as follows.

Definition 1 ***Blocks***
A block is a coherent segment of a document that has its own heading describing its topic.

We then need to define what are headings. We define it as follows.

Definition 2 ***Headings***

A heading is a visually prominent segment of a document describing the topic of another segment.

We define headings as shown above because we made the following observations on blocks and headings in documents.

Observation 1

When a document includes a block with its own topic and the author inserts a heading describing the topic, the author makes the heading visually prominent so that (1) readers can easily distinguish the block from its neighbors, understand its topic, and correctly interpret the information in it, and (2) readers can locate the information on that topic by scanning only headings.

For example, each block in the page shown in Figure 3.1 has its heading describing what topic of the block, and the user can locate relevant blocks by scanning only these headings. Many pages have hierarchical topics. We also made the following observation on headings in such pages.

Observation 2

When a page has hierarchical topics, its contents are organized into hierarchical blocks with hierarchical headings so that readers can locate information by a top-down recursive scan of its hierarchical headings, and can understand the topic of a block by its and its ancestors' headings.

For example, suppose users who want to find the information on the opening hours of the Kyoto Aquarium in the document in Figure 3.1. They first scan the top-level headings "Overview", "Information" and "History" then read the block associated with "Information". Next, they scan the next-level headings "Holidays" and "Opening Hours" within the block then read the block associated with "Opening Hours." As shown in this example, each heading corresponds to some significance level, and scan starts from the most significant headings and proceed to lower ones.

We regard an entire document as a block with regarding its title as its heading. Thus, all blocks in a document form a hierarchy rooted by a block corresponding to the entire document.

3.2.2 Properties of Headings

Based on the observations above, we assume that appropriately designed headings have the following properties. We will discuss the validity of these assumptions through the analysis of our HTML data set later in Section 3.4.4.

Assumption 1 ***Positions of Headings***

Each heading appears at the beginning of its corresponding block.

We assume that page authors insert headings at the beginning of the corresponding blocks so that readers can start reading the corresponding block soon after finding a relevant heading.

Assumption 2 ***Visual Styles of Headings***
Headings and non-heading segments never have the same visual style.

It is to help readers easily scan only headings without reading the other parts of a document. A *visual style* consists of various attributes, such as font size and font weight.

Assumption 3 ***Visual Styles of Heading Lists***
Two headings have the same visual style if and only if they are at the same significance level.

It is to help readers easily jump from a heading to the next heading of the same level in each step in a recursive scan.

Assumption 4 ***Visual Styles of Hierarchical Headings***
Headings at higher levels are given more prominent visual styles than headings at lower levels.

It is to help top-down recursive scan of hierarchical headings.

We call a sequence of headings with the same visual style (and therefore at the same significance level) a *heading list*. We also call a sequence of blocks corresponding to a heading list a *block list*. A block list in our definition has a broader sense than a list in prior research on list extraction. A block list may consist of heterogeneous items as long as their headings share the same visual style. For example, items in the page in Figure 3.2, and the three top-level sections in the document in Figure 3.1 form block lists in our definition although they have completely different internal structures.

The hierarchical structure of the document in Figure 3.1 is shown in Figure 3.3. Each rectangle represents a block, and a subscript of a heading represents which heading list it belongs to. Note that a block list in our definition may be interleaved by other components. For example, four blocks corresponding to one event in 2010 and three events in 2012 are not siblings and are interleaved by the higher-level heading "2012," but they form a block list in our definition. Such an interleaved list is split into multiple sets of sibling blocks when we later segment the page into hierarchical blocks.

We define the logical *depth* of a heading by the depth of its associated block in the logical block hierarchy. Note that headings of the same significance level may not have the same logical depth. For example, a subsection occurring inside some section and a subsection occurring immediately under a chapter without a section have the same significance level but have different logical depths.

3.2.3 Correspondence between Logical and Mark-up Structure

Nested mark-ups in a HTML document form a tree called a DOM tree, but it does not coincide with the logical hierarchy as mentioned before. We discuss correspondence between them below.

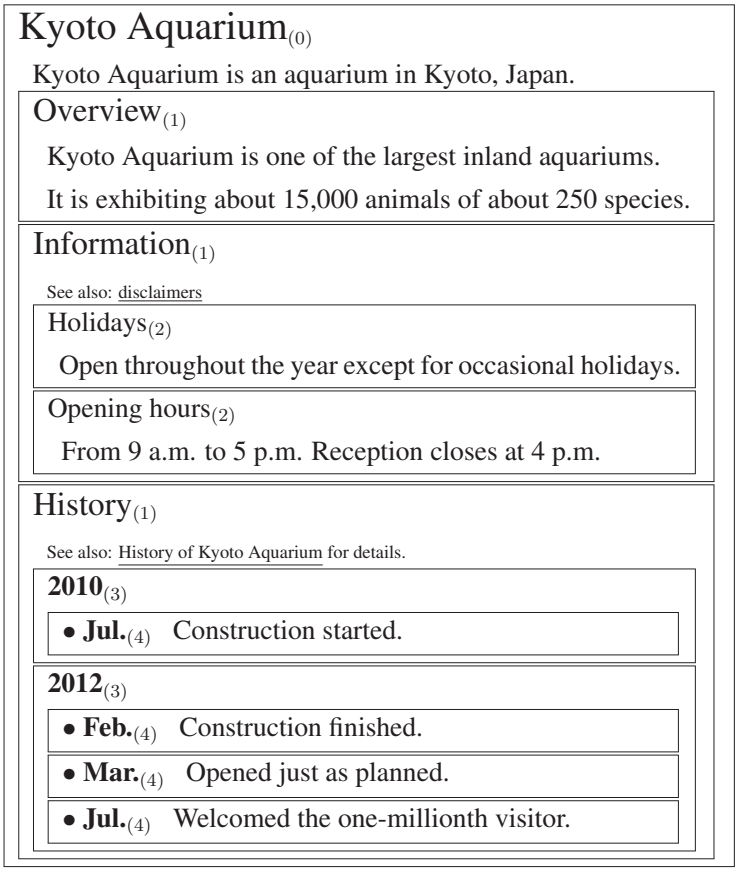


Figure 3.3: Heading structure of document in Figure 3.1. Subscripts (0) to (4) of headings corresponds to five heading lists.

Headings and Nodes

We first discuss the correspondence between headings and DOM trees. Although HTML has tags for headings, our survey shows that only 1/3 of headings in our definition are marked up with these tags as explained before. The other headings are expressed by a text or IMG element with an explicit specification of the visual style. Because a heading only describes the topic of some block, a single text or image node is sufficient to express a heading in most cases.

Blocks and Node Sequences

Next, we discuss how blocks are represented in a DOM tree. Because a block is a coherent segment, it corresponds to a contiguous segment in the HTML source text. In addition, it is very rare that a block only partially overlaps with a subtree in the DOM tree. This is because a block is a semantic coherent unit, while a block containing only the starting or ending tag of a tag pair

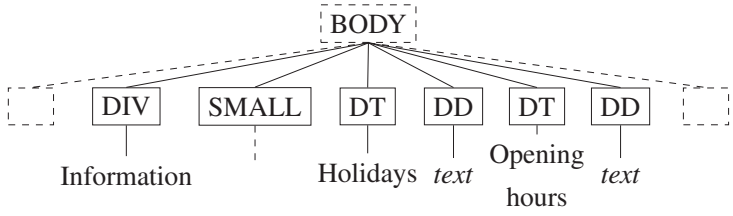


Figure 3.4: Node sequence representing "Information" block in HTML document in Figure 3.3. It consists of six sibling nodes inclusive of their descendants but not their parent node (BODY).

is not a self-contained HTML fragment. On the other hand, a block also never partially overlaps with a text node because we cannot change the visual style in the middle of a text node, which means it is difficult to represent a boundary of a block in the middle of a text node. The only substructure in a tree structure satisfying the three conditions above is a *node sequence* consisting of adjoining sibling nodes (or consisting of a single node) inclusive of their descendants. Therefore, a block corresponds to a node sequence in the DOM tree.

Because a heading appears at the beginning of its block (Assumption 1) and is represented by a single text or image node, the heading of a block is represented either by the first node of the corresponding node sequence or by its descendant.

Figure 3.4 shows a node sequence corresponding to the "Information" block in the document in Figure 3.3. It consists of six sibling nodes, and its heading is represented by the text node "Information," which is a descendant (child) of the first node of the sequence.

Nested Blocks and Block Lists

Lastly, we consider how nested blocks and block lists are represented in a DOM tree structure. When a block b_1 represented by a node sequence s_1 includes another block b_2 represented by a node sequence s_2, s_2 is either a substring of s_1 or a substring of the children of a self-or-descendant of a member of s_1. A block list is represented by a set of node sequences that have headings of the same visual style (Assumption 3). Blocks in a block list may have different logical depths in the block hierarchy, as explained before. On the other hand, node sequences representing blocks in a block list usually have the same depth in the DOM tree. This is because they have headings with the same visual style, and the visual style of an HTML node is affected by the tag paths of the node, i.e., a sequence of tags along the path from the root to the node. To add the same visual style to all headings in a heading list, the author usually places them beneath the same tag path. If two blocks in a block list are placed at different depths in the DOM tree, the author cannot transpose them while maintaining the tag paths of their headings. Therefore, all node sequences representing blocks in a block list usually have the same depth in the DOM tree.

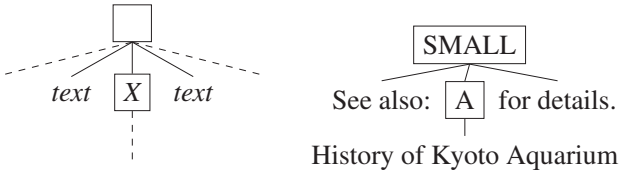


Figure 3.5: General structure of sentence-breaking node *X* (left) and example from document in Figure 3.3 (right).

3.3 Structure Extraction Method

In this section, we explain our structure extraction method, which we call HEPS (HEading-based Page Segmentation). HEPS simulates the behavior of human readers described in Observation 2 based on the assumptions explained in Section 3.2. It divides a document into nested blocks by recursively detecting and scanning headings in the order of their significance level. The method is composed of three steps except for preprocessing. First, it classifies DOM nodes in a given document into node sets according to their visual styles. Second, it tentatively assumes that every set represents a heading list, and sorts the sets in the order of their presumed significance level based on their position in the DOM tree and their visual prominence. Third, it determines a node set representing top-level headings and divides the document into blocks. It recursively repeats the last step for each obtained node set and extracts the hierarchical structure of the document.

3.3.1 Preprocessing

Before the main steps, we remove two kinds of nodes: blank nodes and sentence-braking nodes.

DOM trees of HTML documents may contain *blank nodes*, i.e., text nodes that only include whitespace characters (defined in Unicode 6.0). Because most blank nodes are used for indenting the source text and not for describing information, we remove all blank nodes from DOM trees before the other steps.

DOM trees also often contain subtrees corresponding to smaller structures than a "sentence." For example, link anchor tags and emphasizing tags are often used for marking up only a part of a sentence. They produce nodes that divide a sentence into multiple text nodes. Such *sentence-breaking nodes* are rarely headings, but our method often misread them for headings, and their removal significantly improves the accuracy of our method (see Section 3.4.4).

We regard nodes satisfying the following two conditions illustrated in Figure 3.5 (left) as sentence-breaking nodes: (1) a sentence-breaking node *X* must be an internal node, and (2) it must appear between two sibling text nodes. The two conditions above, however, cannot detect

sentence-breaking nodes at the beginning or end of a text. To detect such nodes, we use their visual styles. Because sentence-breaking nodes often have specific visual styles, if some nodes have the same visual style as some of the already-found sentence-breaking nodes, we also regard them as sentence-breaking nodes. The equivalence of visual styles is discussed in Section 3.3.2.

For example, the document in Figure 3.3 has two sentence-breaking nodes represented by small font size and underlines. The one in the "History" section satisfies the conditions above, as shown in Figure 3.5 (right). As a result, the other one of the same style in the "Information" section is also regarded as a sentence-breaking node.

We remove a sentence-breaking node by creating a text node from its contents and concatenating it with adjoining text nodes. We explain how to convert contents into a string in Section 3.4.2.

3.3.2 Classification of Nodes by Visual Styles

In the first main step of HEPS, we classify nodes by their visual styles. As we discussed in Section 3.2.3, a heading is either a text node or an IMG element. Therefore, we consider them as *candidate-heading nodes* and classify them. To determine their visual styles, we use three types of information explained below.

Tag path The tag path of a node is a sequence of tag names along the path from the root node to the node [70]. For example, `/HTML/BODY/TABLE/TR/TD/UL/LI/text()` is a tag path. We use tag paths to determine visual style because the visual style of a HTML node is affected by its and all the ancestor nodes' names. For example, a text node with the tag path above is rendered with an enclosing border by TD tags and a bullet attached by LI tags.

Computed style In HTML, authors can also directly assign visual styles to nodes by using CSS. To include styles specified by CSS, we also use the *computed styles* of nodes calculated by web browsers based on several factors, such as external style sheets and *style* attributes of the nodes. Computed styles contain many properties, but according to our survey, only a few of them are used for making headings prominent: *font-size*, *font-style* (to display in italics), *font-weight* (in bold), *text-decoration* (sidelines) and *color*. These properties are also effective for classifying image nodes because they specify the visual styles of their alternative texts.

Height of images According to our experiments, height is also useful for the classification of image nodes because the content of an image used as a heading is usually characters, and the height of such an image reflects the font-size of the characters.

We consider that two nodes have the same visual style if they have the same tag path, the same values for the five CSS properties above, and the same height if they are images. Because of Assumption 2 and Assumption 3, each set produced by this classification is either a heading

list defined in Section 3.2.2 or a set of nodes that are not headings. We call these node sets *candidate-heading lists*.

3.3.3 Sorting Candidate-Heading Lists

The first step above produces a set of candidate-heading lists. In the second step, we tentatively assume that all of them are true heading lists and sort them in their presumed significance level order. Those that are not heading lists will be removed later. We sort them based on three factors: depth in the DOM trees, visual styles, and the document order, in this priority order.

Sorting by Block Depth

To sort heading lists by their significance level, we first sort them by the depth of their associated blocks in DOM tree. Note that all blocks in a block list must have the same depth (Section 3.2.3).

It corresponds to topological sorting by inclusion relationship among block lists. Because the depth of a block is smaller than or equal to the depth of its descendant blocks (Section 3.2.3), the sorting by the depth of associated blocks never contradicts the ancestor-descendant relationship between heading lists. On the other hand, if two heading lists have no ancestor-descendant relationship in a page, it makes no sense to sort them by the depth of their associated blocks, but the ordering between them is unimportant anyway.

We determine the depth of the associated blocks of a heading list by finding the least upper bound of two adjoining headings in the heading list. We compute it by the following procedure.

Input: a candidate-heading list $[e_1, \ldots, e_n]$
1: if $n = 1$ return $[p_1]$ where p_1 is document root node
2: $a := [e_1, \ldots, e_n]$
3: $p := [p_1, \ldots, p_n]$ where p_i is parent node of e_i
4: if $p_i = p_j$ for some $i \neq j$ return a else $a := p$
5: jump to 3
Output: first nodes of blocks associated with $[e_1, \ldots, e_n]$

We call each node in the output of this procedure the *front node* of the corresponding input heading node because it is the first (in document order, i.e., preordering in the DOM tree) node in the block associated with the input heading node. The depth of the front node is the depth of the associated block.

Figure 3.6 illustrates how this procedure works when the input is the heading list (3) or (4) in Figure 3.3. When the input is the heading list (3), two nodes “2010” and “2012” in the input are first replaced with their parent nodes, which are B nodes. In the second round, because their parents are the same DIV node, the procedure returns an array of the two B nodes. They are

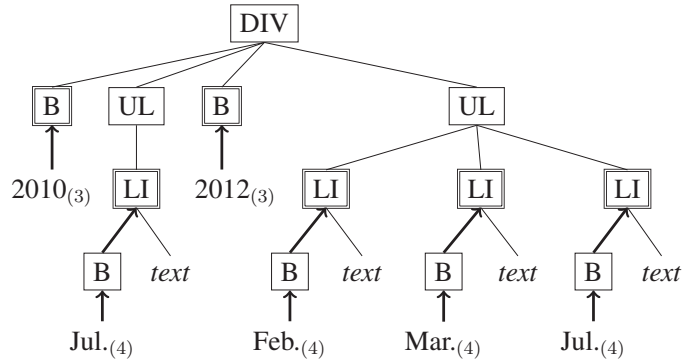


Figure 3.6: Partial DOM tree of document in Figure 3.3. This figure shows how we compute depth of blocks associated with heading lists (3) and (4). Arrows represent node replacement and double rectangles represent front nodes.

the first nodes of the blocks associated with the headings "2010" and "2012", and their depth is the depth of the associated blocks. When the input is heading list (4), the four input nodes are replaced with B nodes in the first round then by LI nodes in the second round. In the third round, because some nodes share the same parent node, which is the UL node at right, the procedure returns an array of the four LI nodes. They are the first nodes of the blocks associated with the four input headings, and their depth is the depth of the associated blocks.

All headings in a heading list have the same depth in the DOM tree (Section 3.2.3). In each candidate-heading list produced in the first step, all elements have the same depth in the DOM tree because we used tag paths to classify nodes. Thus all front nodes in the output of this procedure also have the same depth, and we can sort heading lists by the depth of their front nodes.

The procedure above works only if there is a pair of adjoining blocks. When there is such a pair of blocks, they correspond to two adjoining node sequences in the DOM tree, and all top-most nodes in these sequences share the parent node. Thus the procedure stops just before the parent, i.e., at the top-most nodes of the blocks. We start from the headings, each of which must be the first node of the corresponding node sequence or its descendant (Section 3.2.3). Therefore, the procedure returns the first node in the node sequence representing the associated block.

However, when there is no pair of adjoining blocks, the procedure above fails to return correct front nodes. It happens when a candidate-heading list consists of only one node, and also when all the associated blocks of the headings are interleaved by other components. In the former case, we treat the document root as its front node for convenience, but such a list is anyway removed in the next step. The latter case is very rare, and we ignore it in this dissertation.

For each candidate-heading list produced in the first step, we compute the depth of the asso-

ciated blocks by this procedure, and sort the lists in ascending order of the depth. When we have a tie, we break it based on their visual styles, as explained below.

Sorting by Visual Style

HTML has tags for specifying multi-level headings (H1, H2, and so on). Today's popular web browsers render upper headings with a larger font, or a bold font of the same size. We assume that authors who directly specify the visual styles of headings follow this convention so that readers can understand the relationship between the headings. When we have two candidate-heading lists whose front nodes have the same depth, we sort them first in descending order of their *font-size* then in descending order of their *font-weight*. All headings in a candidate-heading list have the same value for these properties because we use them when we classify nodes.

If we still have a tie, we break it based on document order.

Sorting by Document Order

Because headings are placed at the beginning of the corresponding blocks, when we have multi-level headings, the first heading of higher significance level usually appears prior to the first heading of lower significance level. Based on this observation, we finally break ties by sorting the heading lists in the document order of the first element of each list. Because all nodes in a document have different positions in document order, we have no more ties.

3.3.4 Recursive Document Segmentation

Given a sorted list of candidate-heading lists produced in the second main step, we first segment the document into top-level blocks by using the first candidate-heading list then segment these blocks by using the next candidate-heading list. We segment the document into nested blocks by repeating this process, as human readers do. In each step, we also determine whether the given candidate-heading list is really a heading list. To determine this, we first try to segment the given blocks by using the given candidate-heading list, and if it produces an inappropriate block structure, we determine that it is not really a heading list and skip it.

Because we use the document segmentation in that determining process, we first explain how we segment documents then how we determine whether a given list is really a heading list.

Segmentation Based on Headings

As explained in Section 3.3.3, a front node is the first node of a node sequence representing a block associated with a given heading. Therefore, we can extract the node sequence representing a block by starting from the front node, scanning its following siblings, then detecting the last node of the node sequence.

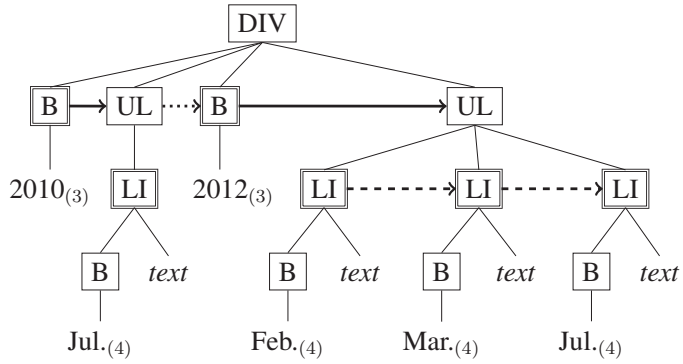


Figure 3.7: Detection of blocks starting from front nodes found in Figure 3.6. Scan proceeds along arrows and stop before dashed arrows. Double rectangles represent front nodes.

We use the following conditions for detecting the last node.

No following sibling If there is no more following sibling, we stop the scan and regard the current node as the last node.

Another front node Front nodes produced from one candidate-heading list represent the first nodes of blocks belonging to the same block list, and therefore, these blocks never include nor overlap each other. Therefore, if the following sibling is a front node of another heading in the current candidate-heading list, we stop the scan and regard the current node as the last node.

Node including already-found upper-level headings Because a heading appears at the beginning of its block, and two blocks never share the same heading, a lower block never contains a heading of any upper block. Therefore, if the next following sibling includes any already-found upper-level headings, we regard the current node as the last node. Because we determine headings and segment a document in top-down manner, in each recursive step, all the upper-level headings have already been determined.

Node not included in current upper block When we scan sibling nodes representing a block, its parent block has already been determined because of our top-down procedure. As discussed in Section 3.2.3, a node sequence of a block is contained by the node sequence of its parent block. Therefore, if the next sibling is not contained in the node sequence of the parent block of the current block, we regard the current node as the last node.

Figure 3.7 shows how we detect a node sequence representing a block starting at each front node found in Figure 3.6, which are represented by double rectangles. When we start from the left-most B node, we stop at the next UL node because the following sibling, i.e., the third B node from the left, is another front node for the same candidate-heading list. When we start from the

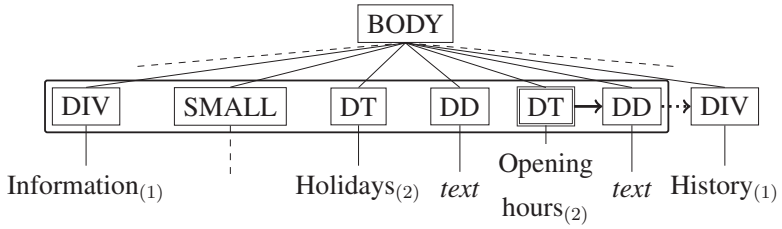


Figure 3.8: Scan starting from front node of heading "Opening hours" in Figure 3.4 stops at next DD node because the following sibling is outside upper "Information" block, which corresponds to six sibling nodes enclosed by rectangle in this figure.

left-most LI node, we stop immediately because there is no following sibling.

Figure 3.8 shows another example in which a scan stops at the last sibling node within the upper block. Six children of the BODY node, which are enclosed by a rectangle in this figure, correspond to the "Information" block in the document in Figure 3.3. When we start from the front node of the heading "Opening hours", which is the DT node on the right, we stop at the next DD node because the following sibling is outside the upper "Information" block.

Determining Heading Lists

Next, we explain how we determine whether a given candidate-heading list is really a heading list. We use five conditions below for detecting non-heading nodes. For each condition, we compute the ratio of the number of nodes satisfying it to the number of nodes in the list, and if it is larger than a threshold θ for any condition, we determine that the list is not a heading list.

1. Node whose front node includes upper-level headings If the front node of a candidate heading includes headings of upper blocks, the block produced from the front node would include them. A block, however, never includes headings of upper blocks, as explained before. Therefore, if a candidate heading has such a front node, we determine that it is not really a heading.

2. Node producing empty block Because a block must have a heading and a heading must have an associate block by their definitions, we do not consider a block including no text or image node except for its heading. Therefore, if we obtain such an empty block from a candidate heading, we determine that it is not a heading.

3. Node without sibling candidates We call blocks that share the same parent block *sibling blocks*. Similarly, we call nodes in the same already-found parent block and in the same candidate-heading list *sibling candidate headings*. Because headings are usually used to segment a block into multiple child blocks, it is rare that a block has no sibling block. Therefore, we consider that a candidate heading without a sibling candidate heading is not a heading. A

Table 3.1: Thresholds θ and t optimized for training data set.

Condition	Threshold θ
1. Including upper-level headings	0.1
2. Producing empty block	0.2
3. No sibling candidates	0.7
4. Non-unique contents	0.6
5. Too much content as a heading	0.3 ($t = 1.5$)

candidate-heading list consisting of only one node, which was discussed in Section 3.3.3, is always eliminated by this condition.

4. Node with non-unique contents Because a heading describes the topic of the associated block (Section 3.2.2), two sibling blocks rarely have headings with the same content. On the other hand, two blocks that are not siblings may have headings with the same content. For example, in the document in Figure 3.3, two blocks about July 2010 and July 2012 have the same heading "Jul." Therefore, we determine that a candidate heading is not a heading if there is another sibling candidate heading with the same content. The equivalence of content is determined after image-to-text conversion and space normalization, which is explained in Section 3.4.2.

5. Node with too much content As a heading describes the topic of its associated block, if the *length of the associated block* / *length of a candidate heading* $< t$, in other words, if the candidate heading is too long compared with its associated block, we consider that the candidate heading is not a heading. The length of nodes are defined by the number of UTF-8 characters in them (after image-to-text conversion and space normalization, see Section 3.4.2).

We optimized the thresholds t and θ for our training data set (see Section 3.4.2) by a greedy algorithm. The result is shown in Table 3.1.

We first examine if the given list is eliminated by any of these conditions, and if it is not, we also separately remove each candidate heading in the list satisfying conditions 1 or 2. We did not use conditions 3, 4, 5 in this node-level filtering because node-level filtering with these conditions significantly degraded the recall ratio in our experiment with the training data set.

3.4 Evaluation

In this section, we show the results of our experiments for evaluating HEPS. Because there is no well-known data set or evaluation measure for hierarchical structure extraction from documents, we created a data set by manually labeling blocks and headings in web pages based on our definitions (Section 3.2.1), and adopted precision and recall measures based on exact matching.

In both the manual labeling and the extraction experiment, we assume that we preprocess pages with some main body extraction method, and we only consider their main bodies as the target data. In our preliminary experiment, if we include headers, side bars, and so on in the target data, the accuracy of our structure extraction method is much higher. It is probably because structure in headers or side bars are easier to extract than structures in the main bodies.

3.4.1 Document Collection

As a standard sample of general web pages, we used the well-known web snapshot ClueWeb09 Category B document collection (ClueWeb09B)[1]. It was created by random crawling and includes many completely useless pages. In information extraction tasks and information retrieval tasks, the performance over useful pages are more important than the performance over all pages including such useless pages. To select pages meaningful for these applications, we randomly chose pages relevant to at least one query in Text Retrieval Conference (TREC)[2] 2009–2012 web track ad-hoc tasks. There were 11,008 relevant documents without duplication of URLs. We then removed all Wikipedia articles, which share the same document structure and amount to 2,728 articles (about 25%) in the collection, in order to prevent their specific document structure from having too much influence on the overall evaluation. We also removed pages known to be redirecting pages[3] and documents whose content type was not text/html. We finally obtained 8,013 documents.

Because ClueWeb09B contains no external files, such as external style sheets and images, which affect the visual styles of the pages, we tried to re-download the obtained 8,013 documents and their external files from Internet Archive[4]. To obtain a page data closest to that in ClueWeb09B, we selected snapshots closest to February 29, 2009; when ClueWeb09 crawling finished. We could download 7,187 documents. After that, we randomly chose 1,393 of the 7,187 documents due to limited annotation resource.

3.4.2 Ground Truth and Its Representation

Annotation process The annotation for creating the ground truth was done by seven participants including one of the authors. We first explained our definitions of headings, heading lists, blocks, and transitions (explained later) to the participants. We did not explain our assumptions in Section 3.2.2 in order to prevent bias in favor of our method. We also explained the definition

[1] ClueWeb09 Wiki `http://boston.lti.cs.cmu.edu/clueWeb09/wiki/`

[2] Text REtrieval Conference Home Page `http://trec.nist.gov/`

[3] ClueWeb09 Wiki: Redirects
`http://boston.lti.cs.cmu.edu/clueWeb09/wiki/tiki-index.php?page=Redirects`

[4] Internet Archive `https://archive.org/`

Table 3.2: Assignment of pages to annotators and data sets.

	A	B⋆	C	D	E	F	G	**Total**
Training set	101	100	93	96	26			416
Test set		200	100	103		200	200	803
Total	101	300	193	199	26	200	200	1,219

⋆ One of authors.

of the *content body* of a page in prior research [45, 62, 92]: a content body is *a clearly specified coherent segment that necessarily and sufficiently contains the information the author must have most wanted to disseminate and which should differentiate the page from the others in the same web site*. Each participant was asked to manually extract the content body from the randomly assigned pages and hand-annotate the hierarchical structure and the headings inside them with our original annotation tool.

To avoid over-fitting, we split the entire data set into a *training* data set and a *test* data set, and optimized the choice of conditions in Section 3.3.4 and their parameters in Table 3.1 with the training data set. The breakdown of the assignment of pages in each data set to annotators is shown in Table 3.2.

A block sometimes have more than one component that can be regarded as its heading. In most of such cases found during the annotation of the training data, a block is describing an entity, and both its name and picture can be regarded as its heading. In real applications, the users may regard either of them as the heading. To express such situations, we allowed the annotators to mark multiple headings for a single block. During the annotation, we also found that the borders of headings and blocks are sometimes ambiguous. To express such cases, we allowed the annotators to mark some ranges as *transitions*, which means that the components may or may not be a part of the heading or block. For example, a horizontal line between two blocks is a transition for both blocks.

Exceptions During the annotation, we removed 174 pages (12.5% of 1,393 pages) from the answer set because their structures were difficult to extract even for human users. The most significant reasons were redirecting page (2.6%) and too much content (2.5%). If a page needed more than 15 minutes for annotation, we removed it as a page with too much content to give priority to the number of annotated pages. We also removed a few dynamic pages (0.4%) because annotating dynamic content is difficult. Note that HEPS itself can be applied to such dynamic pages at any point in time.

Data set statistics Finally, we obtained annotations for 1,219 pages from 981 domains, which constitute our data set. The maximum number of pages from the same domain was only 9 pages, and 841 pages (69.0% of 1,219 pages) had no other pages from the same domain, which means the data set is sufficiently diversified. Among them, 953 pages (78.2%) contained at least one block other than the entire page, and 423 pages (34.7%) contained hierarchical structures, i.e., at least one parent-child pair of blocks. This result shows that both flat and hierarchical block structures are common in general web pages. The total number of blocks was 15,560, and 1,194 blocks (7.7%) contained two or more headings. The total number of headings was 16,817. Transitions were assigned to only 171 headings (1.0%) and 124 blocks (0.8%). The medians of the number of blocks were 6 per page and 3 per list.

Data representation To represent the annotation results, we defined the *raw string* of a page. It is obtained by joining all the text nodes on the page in their document order. To simplify the discussion, we handle an IMG element as a special text node containing the element's *src* (source) attribute value. We then replace adjoining whitespace characters (defined in Unicode 6.0) with a single space and remove leading or training spaces. We ignore the content of SCRIPT and STYLE elements because they are not rendered on browsers. We represent each heading or block as a set of ranges in the raw string. We use a set of ranges instead of a range because annotators sometimes mark some region that is not contiguous in the raw string (e.g., a column of a table) as a heading or a block.

3.4.3 Evaluation Measures

In our evaluation, an extracted heading or block *matches* an annotated heading or block if and only if they are exactly the same set of ranges except for those marked as transitions. However, if an extracted heading matches a heading of a block that has multiple annotated headings, no other extracted heading matches the remaining annotated headings of the same block.

By this definition of matching, we can adopt the standard measures: precision and recall. For a page from which at least one heading and block pair was extracted, the precision of heading extraction and block extraction, denoted by P_H and P_B, are defined as follows:

$$\begin{aligned} P_H &= |\textit{correctly extracted headings}|/|\textit{all extracted headings}| \\ P_B &= |\textit{correctly extracted blocks}|/|\textit{all extracted blocks}|. \end{aligned}$$

Similarly, for a page containing at least one annotated pair of a heading and a block, the recall of heading extraction and block extraction, R_H and R_B, are defined as follows:

$$\begin{aligned} R_H &= |\textit{correctly extracted headings}|/|\textit{all annotated headings}| \\ R_B &= |\textit{correctly extracted blocks}|/|\textit{all annotated blocks}|. \end{aligned}$$

Although HEPS first extracts headings then extracts blocks based on the extracted headings, P_B and R_B may exceed P_H and R_H, respectively, because HEPS may correctly extract blocks based on incorrectly extracted headings.

In the calculation of precision and recall, we did not count the entire content body as a block because its range is given. When we count the number of all annotated headings, we count multiple headings of a block as one heading.

Precision/recall over a data set are calculated by taking arithmetic means of precision/recall for each page (excluding pages with no extracted or annotated ones for the computation of precision or recall, respectively). We also calculate balanced F-measures:

$$\begin{aligned} F_H &= 2P_H R_H/(P_H + R_H) \\ F_B &= 2P_B R_B/(P_B + R_B) \end{aligned}$$

to integrate these two measures.

3.4.4 Evaluation Results and Discussions

In this section, we explain the result of our experiments.

Inter-annotator agreement

First, we discuss inter-annotator agreement because each page is labeled by only one annotator. We calculated Fleiss' Kappa coefficient [41] for five annotators of our test data set. For this purpose, we collected another *agreement* data set containing 102 pages by the same process as the training and test data sets explained before. From these pages, annotator B extracted content bodies then the five annotators labeled headings and blocks in the bodies.

We calculate Kappa coefficient by considering that annotators make binary judgments whether each candidate is a heading (or a block). Because it is impractical to count all the sets of ranges that can be labeled by our annotation system as the candidates, we only count text and IMG nodes as candidate headings, and only count block-level elements of HTML 4.0 as candidate blocks. They covered 73.1–95.8% (depending on the annotator) of annotated headings and 48.6–61.0% of annotated blocks in the agreement data set.

The arithmetic mean of Fleiss' Kappa coefficient computed for each page in this setting was 0.693 for heading annotation and 0.583 for block annotation. The former can be regarded as showing a good [41] and substantial [59] agreement, and the latter can be regarded as showing a fair to good agreement [41] or a moderate to substantial [59] agreement. This fact supports validity of our problem definition and reliability of our data sets.

Table 3.3: Accuracy of HEPS for training/test data sets.

Data set	P_H	R_H	F_H	P_B	R_B	F_B
Training	.691	.564	.621	.617	.539	.575
Test	.638	.569	.602	.586	.563	.574
Total	.656	.567	.608	.596	.555	.575

Table 3.4: Accuracy of HEPS for test set by each annotator.

Annotator	pp.	P_H	R_H	F_H	P_B	R_B	F_B
B $\star$	200	.692	.610	.648	.651	.603	.626
C	100	.616	.582	.599	.594	.589	.591
D	103	.556	.486	.519	.484	.468	.476
F	200	.573	.527	.549	.565	.548	.556
G	200	.705	.608	.653	.592	.579	.585

$\star$ One of the authors.

Over-Fitting to Training Data

Because the conditions in Section 3.3.4 and their threshold values in Table 3.1 are chosen based on the experiment with the training data set, we compared the results of HEPS for the training data set and for the test data set to see if over-fitting occurred. Table 3.3 lists the results. P_H most significantly degraded (-0.053) for the test data set, while F_B remained the same. These results mean minor over-fitting occurred especially in heading extraction.

Influence of Annotators

We also analyzed the influence of the annotators. Table 3.4 lists the performance of HEPS for the data subset corresponding to each annotator. Although annotator B is one of the authors, the heading extraction by HEPS worked best for the pages annotated by annotator G (+0.005 in F_H followed by B). This proved that HEPS does not over-fit to data by a specific annotator, which means that its effectiveness does not depend on readers. For block extraction, HEPS worked best for the pages annotated by B, but the difference was small (+0.035 in F_B, followed by C).

Comparison with Existing Methods

Next, we compare HEPS with some baselines. We first compare the performance of heading extraction. We chose the decision tree method [77] as the first baseline because its F_H (0.852) is better than F_H (0.851) achieved by Sano et al.'s method [89] according to the experiment reported in [77, 89], and detailed sets of rules or features for the other methods [82, 99] are not available.

Table 3.5: Comparison with naive method and existing methods.

Method	P_H	R_H	F_H	P_B	R_B	F_B
Decision tree [77]	.084	.884	.154			
Hn & DT	.668	.320	.433			
VIPS [16]				.215	.070	.106
HEPS	.638	.569	.602	.586	.563	.574

Following the description in [77], we constructed a decision tree by using the J4.8 algorithm of the data mining tool Weka[5], which is exactly the implementation used in [77]. The J4.8 algorithm has several parameters, e.g., the threshold for pruning, but the parameters used in their experiment are not described in [77], so we used default parameters. We trained it using all 5,379 headings and randomly chosen 5,379 non-heading nodes in our training data set. Its F_H by 10-fold cross validation with the training data set was 0.865, which is close to 0.852 reported in [77]. This suggests that we reproduced their results quite successfully. However, the decision tree trained by the training data set worked poorly for the test data set, as shown in Table 3.5. It was the best result of 10 attempts. This is probably because our data set is diversified, while their data set was not. As explained in Section 2.1, the evaluations in [77] were based on cross-validation, in which the training and test data sets may share headings in the same format from the same page.

We also tested a naive heading extraction method, *Hn & DT*, which extracts elements with the proper HTML tags for headings, namely H1 to H6 and DT. The R_H value of the Hn & DT method shown in Table 3.5 shows that only less than 1/3 of headings were marked up with these proper tags, as explained in Section 3.1. Our result also showed low P_H; around 1/3 of nodes marked up by them are not headings. To determine the reason, we inspected 100 pages with the worst P_H. In 45 of those 100 pages, some (or all) of H1–H6 and DT nodes mark up some metadata (e.g., author names and timestamps). In 33 pages, most nodes marked up with H1–H6 and DT were not so visually prominent and the annotators overlooked them, which means those nodes do not actually work as headings for readers. As explained in Section 3.1, we suspect that such usage of heading tags is related to search engine optimization (SEO).

The P_H of HEPS was only slightly worse (-0.030) than that of the Hn & DT method, which is quite satisfactory, given that the Hn & DT method is focused only on *explicit* headings. Moreover, HEPS achieved a far better R_H (+0.249) than the Hn & DT method. These results show that HEPS can extract many implicit headings while maintaining satisfactory precision.

We next evaluate the performance of block extraction by HEPS. We compared HEPS with

[5]Weka 3: Data Mining Software in Java `http://www.cs.waikato.ac.nz/ml/weka/`

VIPS [16]. VIPS is focused on a top-level page structure while HEPS is focused on a hierarchical structure in the main body as explained in Section 2.1. Nonetheless, we chose VIPS because there is no available method focused on a hierarchical structure in the main body, and for extraction of top-level page structure, VIPS [16] is most widely used [54,55,92]. The implementation of VIPS is available at Deng Cai's web site[6].

VIPS has a parameter called degree of coherence (DoC) that defines the granularity of blocks at which VIPS stops recursive segmentation. The larger DoC is, the deeper level VIPS proceeds to. When measuring P_B, we used the DoC value that achieves the best P_B value for each page. (However, we did not allow too low DoC values that make VIPS produce no block for pages that actually have blocks. Otherwise, we can inflate average P_B of VIPS by eliminating all pages from the computation of average P_B except for the page with the highest P_B.) When measuring R_B, we used the maximum DoC value, which makes VIPS extract as many blocks as possible.

The result summarized in Table 3.5 shows that HEPS achieved far better P_B (+0.371) and R_B (+0.493) than VIPS. It proves that hierarchical heading structures are far different from top-level layout structures targeted with the many existing methods. HEPS and existing methods are, therefore, complementary to each other.

Effects of Design Decisions and Assumptions

We next measure the effects of various design decisions in HEPS.

Heading candidate extraction We assumed that a heading is either a text node or an IMG node. If we extract all of them, R_H is 0.896 (Table 3.6(a)). It is 0.928 (only +0.032) if we extract all HTML nodes in the page, as in Okada and Arakawa [77] or Tatsumi and Asahi [99]. These results show that our strategy of simply choosing text and IMG nodes as candidates is reasonable. Table 3.6(a) also shows R_H values for all text nodes and all IMG nodes. Note that their sum is not equal to R_H for all text and IMG nodes because there are blocks that have both a text heading and a IMG heading.

Removal of Sentence-breaking nodes If we do not remove sentence-breaking nodes, P_H and R_H with HEPS become 0.492 (-0.146) and 0.578 (only +0.009), respectively (Table 3.6(b)). It means that most sentence-breaking nodes are not headings and that HEPS sometimes misread them for headings if we do not remove them.

Information on visual styles To see if the three types of information used in candidate heading classification are effective, we ran HEPS without each of them. The F_H is 0.593 (-0.009) without tag paths (but, instead, with the length of tag paths), 0.573 (-0.029) without computed styles, and

[6]VIPS: a VIsion based Page Segmentation Algorithm
`http://www.cad.zju.edu.cn/home/dengcai/VIPS/VIPS.html`

Table 3.6: Accuracy of naive methods and HEPS variations.

Method	P_H	R_H	F_H	P_B	R_B	F_B
HEPS	.638	.569	.602	.586	.563	.574
(a) Extraction of candidate heading nodes						
All text nodes	.118	.871	.208			
All IMG nodes	.110	.086	.097			
All text and IMG nodes	.108	.896	.193			
All HTML nodes	.026	.928	.051			
(b) Removal of sentence-breaking nodes						
Leave breaking nodes	.492	.578	.532	.453	.570	.505
(c) Classification of nodes by visual styles						
Without tag path	.647	.548	.593	.591	.541	.565
Without computed style	.612	.539	.573	.562	.532	.547
Without image height	.630	.571	.599	.579	.564	.571
(d) Sorting candidate heading lists						
Only by document order	.635	.568	.600	.566	.555	.560
(e) Determining actual heading lists						
Without condition 1.	.636	.573	.603	.581	.565	.573
2.	.638	.582	.609	.583	.577	.580
3.	.545	.563	.554	.361	.551	.436
4.	.597	.562	.579	.561	.561	.561
5.	.606	.574	.590	.550	.567	.558

0.599 (-0.003) without image height (Table 3.6(c)). These results show that all these types of information are more or less useful for the classification.

Sorting candidate-heading lists Even when we sort candidate-heading lists only by their document order, F_H dropped by only .0002 (Table 3.6(d)). There are two purposes to sorting of candidate lists: (1) processing the correct ones earlier and (2) processing the upper-level headings earlier and the lower-level headings later. If the step filtering out non-heading lists works well, only purpose (2) is important, and the document order may be sufficient for that.

Segmentation based on headings The precision P_B of blocks segmented by correctly extracted headings is 0.769. This supports the validity of our idea of segmenting a document based on its headings, and implies that we can improve the precision of block extraction by improving the precision of heading extraction.

Determining heading lists We measured the effects of five conditions for filtering out non-heading lists discussed in Section 3.3.4. Table 3.6(e) shows the performance of HEPS without set-level filtering with each condition. The values of F_H increased for conditions 1 (including upper-level headings) and 2 (producing empty block). We adopted these two because they were useful for the training data set, but they may not be useful in general. On the other hand, F_H degrades when we remove condition 3, 4, and 5.

We also evaluate the validity of our assumptions in Section 3.2.1, on which our method relies, by using all the true blocks and their headings in both the training and test data set (1,219 pages).

Positions of Headings In 94.6% of true blocks, their headings appear at the beginning of the blocks. It validates our assumption.

Visual Styles of Headings For each page, we computed the ratio of headings that have no non-heading nodes with the same visual style within the page. The ratio averaged over 1,129 pages was 73.6%, which is lower than expected. We examined the error cases, and found that this assumption itself is valid even in most error cases, but the set of visual features we used is not enough, and could not distinguish some headings from non-heading nodes. For example, some headings are distinguished from other texts by centering horizontally, or by using some prominent markers, but our method cannot recognize all of them. We should include more visual features in our method, but some of them are not easy, e.g., the detection of horizontally centered text in two-column pages.

Visual Styles of Heading Lists We also computed the ratio of true heading lists whose all members have the same visual style. Its average over 1,129 pages was 76.4%, which is also lower than expected. According to our error analysis, this is because our definition of equivalence between visual styles is too strict. For example, headings in the same list sometimes have slightly different font size (e.g., smaller fonts for longer headings). It suggests that some kind of clustering techniques may improve our method.

Accuracy of Hierarchical Structure Extraction

Finally, we measured the accuracy of hierarchical structure extraction with HEPS. Although tree edit distance is the well-known measure for comparing hierarchical structure, the edit distance is not intuitive. We instead measured the accuracy by precision/recall of the extracted relationships. To calculate them, we collect all instances of the relationship between a correctly extracted heading/block and another true or extracted headings/blocks, then calculate precision/recall of the extracted ones. For example, to measure the accuracy of ancestor block extraction, we collect all pairs of a correctly extracted block and its true or extracted ancestor blocks, then calculated

Table 3.7: Accuracy of relationship extraction with HEPS.

Relationship	P_H		R_H		F_H		P_B		R_B		F_B	
Baseline	.638		.569		.602		.586		.563		.574	
Ancestors	.810	(+26.9%)	.582	(+2.28%)	.678	(+12.6%)	.542	(-7.51%)	.431	(-23.4%)	.480	(-19.5%)
Parent	.808	(+26.6%)	.580	(+1.93%)	.675	(+12.2%)	.542	(-7.51%)	.431	(-23.4%)	.480	(-19.7%)
Siblings	.879	(+37.7%)	.813	(+42.8%)	.845	(+40.4%)	.757	(+29.1%)	.719	(+27.7%)	.737	(+28.4%)
Children	.489	(-23.3%)	.582	(+2.28%)	.532	(-13.2%)	.504	(-13.9%)	.563	(+0.00%)	.532	(-7.95%)
Descendants	.476	(-25.3%)	.585	(+2.81%)	.525	(-14.6%)	.504	(-13.9%)	.567	(+0.71%)	.534	(-7.56%)

P_H and R_H of the extracted ancestor blocks. Note that it is not symmetric for the descendant side and the ancestor side. Also note that one block is counted many times in the calculation of precision/recall of ancestor extraction when the block has multiple descendant blocks. It is reasonable because such a block is probably important as the ancestor block.

We evaluated HEPS for five types of relationship in a hierarchical structure, namely parent, ancestor (including parents), sibling (excluding self), child and descendant (including children). The results are listed in Table 3.7. We also listed the accuracy of HEPS for the entire test data set for comparison.

We can see three interesting properties in these results. First, accuracy of sibling heading/block extraction is significantly better than the others. It supports the validity of our observation on heading/block lists in Section 3.2.3. Second, HEPS shows higher precision for ancestors and parents than for children and descendants. It is probably because HEPS often incorrectly extract smaller structure as headings, e.g., sentence-breaking nodes which were not removed by our preprocessing step. Third, the accuracy of ancestor (or parent) block extraction is significantly worse than the overall accuracy of HEPS in spite of high precision of ancestor (or parent) *heading* extraction. It is probably because the block lists of higher level tend to contain fewer blocks, and blocks of higher level is more likely to have no next block in the same list, which is important for determining the last node of the block.

3.5 Summary

We developed a method that extracts logical hierarchical structure from HTML documents. Our method first extracts hierarchical headings based on several assumptions on their design, and segment a document into hierarchical blocks by using these headings. This approach is expected to work as long as the document is appropriately designed so that human readers can recognize its hierarchical structure based on its visually prominent headings.

We evaluated our method with a standard web page corpus, and our experiment shows that our method outperforms the existing page segmentation method. The existing methods, however, focus on top-level page layout structure. The existing methods and our method, therefore, are complementary to each other.

The reference implementation of our method and the data set we used in our experiment is publicly available on GitHub (`https://github.com/tmanabe`). The data set consists of a list of Internet Archive URLs, a list of XPaths representing the content body of each page, and manually labeled heading structures.

CHAPTER 4

SUBTOPIC RANKING BASED ON HIERARCHICAL HEADINGS

4.1 Introduction

Web search queries are sometimes ambiguous and/or referring to broad topics. To generate effective web page rankings for such queries, search result diversification techniques have been developed. Subtopic mining is one of the most promising approaches for search result diversification. Diversification methods based on subtopic mining first extract subtopic candidates of queries, then score and rank the candidates by their importance and diversity, and finally returns a few pages for each subtopic candidate with high score. Because of the importance of subtopic mining, competitions for subtopic mining methods have been held as the NTCIR INTENT/IMine tasks subtopic mining subtasks [64, 88, 94].

In general, documents contain hierarchical heading structure reflecting their topic structure. Hierarchical heading structure consists of nested logical blocks and each block has its heading. A heading describes the topic of its associated block and the hierarchical descendant blocks of the block. Because of this feature of heading, hierarchical headings in documents reflect topic structure of the documents. For example, Figure 4.1 is an example web page about computer programming (one of the NTCIR topics) containing hierarchical heading structure. In this figure, each rectangle encloses a block and each emphasized text is a heading. The hierarchical headings in this page reflect its topic structure. For example, its first level topic is computer programming, second level topics are computer programming schools and jobs, and the third level topics are computer programming school courses and degrees.

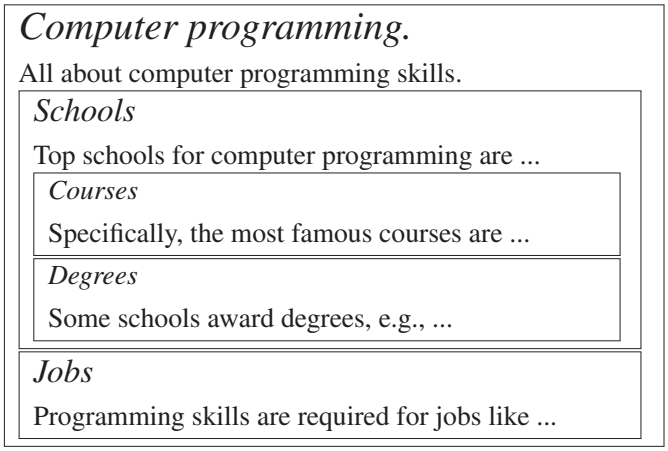


Figure 4.1: Example web page with hierarchical heading structure. Each rectangle encloses block and each emphasized text is heading. Some long texts are replaced by dots.

In this chapter, we propose methods to score hierarchical blocks in documents then subtopic candidates based on the block scores. To the best of our knowledge, this is the first work which discusses the application of detailed hierarchical heading structure of web pages to subtopic mining. Our basic idea is that more contents about a topic suggests more importance of the topic. Our methods score blocks based on the quantity of the contents of the blocks, and our methods compute the score of a subtopic candidate by the summation of the scores of the blocks in corpus whose hierarchical headings match the candidate. To diversify resulting rankings, our methods adopt a subtopic with the best score one-by-one, and every time a subtopic is adopted, our methods re-score all remaining blocks with removing blocks matching with subtopics that have been already adopted. By this approach, the candidates matching the blocks also matching the already-adopted subtopics lose their scores, and resulting subtopic rankings get diversified.

The remainder of this chapter is organized as follows. In the next section, we clarify our research target. After that, we explain our methods in Section 4.3. In Section 4.4, we evaluate our methods on the publicly available NTCIR data set and compare the results with the baselines generated by commercial web search engines. Lastly, Section 4.5 concludes this chapter.

4.2 Definition of Subtopics

In this section, we clarify the definitions of our research target, subtopics of keyword queries.

We focus on subtopics explicitly represented by subtopic strings defined in the NTCIR-10 INTENT-2 task as quoted below [88].

> A subtopic string of a given query is a query that *specializes and/or disam-*

> *biguates* the search intent of the original query. If a string returned in response to the query does neither, it is considered incorrect.

As defined above, each subtopic is associated to the topic behind an original query. In the task and this chapter, a *query* means a keyword query composed of an array of words.

The overview paper lists some example subtopic strings [88]. An original query "harry potter" is specialized by a true subtopic string "harry potter philosophers stone movie." Another string "harry potter hp" is not a subtopic string because it neither specializes nor disambiguates the original query. Another original query "office" is disambiguated by a true subtopic string "office workplace." Another string "office office" is not a subtopic string. Note that true subtopic strings may not include the original queries. For example, "aliens vs predators" is a true subtopic string of the original query "avp."

4.3 Subtopic Ranking Methods Based on Hierarchical Heading Structure

In this section, we propose scoring and ranking methods for subtopic strings. Our proposed methods are based on matching between the subtopic strings and hierarchical heading structure of documents in a document set. We regard that a subtopic string *matches* a block iff all the words in the subtopic string appear either in the heading of the block or in the headings of its ancestor blocks. For example, a subtopic string "computer programming degrees" matches the "degrees" block in Figure 4.1. If a subtopic string matches a block, the block must refer to the subtopic according to the definition of hierarchical heading structure. Because of this definition of matching, if a subtopic string matches a block, the string must also match the hierarchical descendant blocks of the block. However, we do not consider such matching of hierarchical descendants of the already matched blocks. Instead, we score each block considering its descendants.

Formally, the score of a pair of a subtopic string s and a document d is:

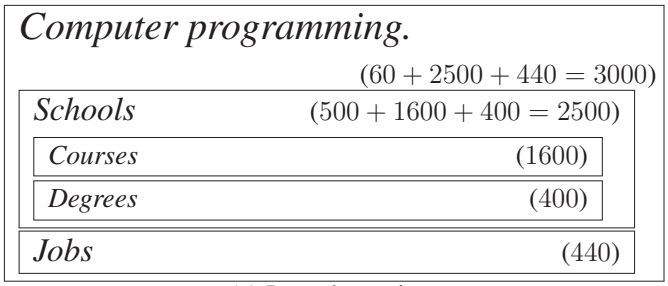
$$\text{docScore}(s, d) = \sum_{b \text{ in } d} \text{match}(s, b)\text{blockScore}(b)$$

where b is each block in d, match(s, b) is 1 iff s matches b and does not match any ancestor block of b and 0 otherwise, and blockScore(b) is the score of b.

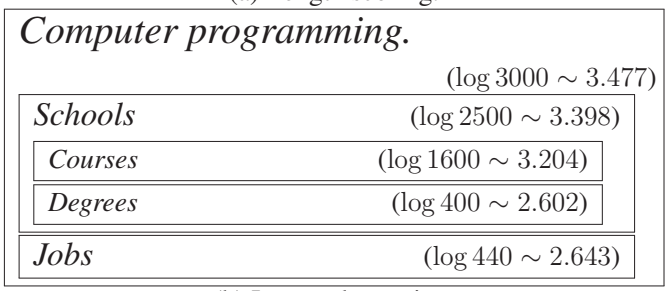
Hereafter in this section, first we discuss the definition of blockScore(b), then integration of subtopic scores on multiple documents, and finally ranking of subtopics into a diversified ranking.

4.3.1 Subtopic Scoring on a Single Page

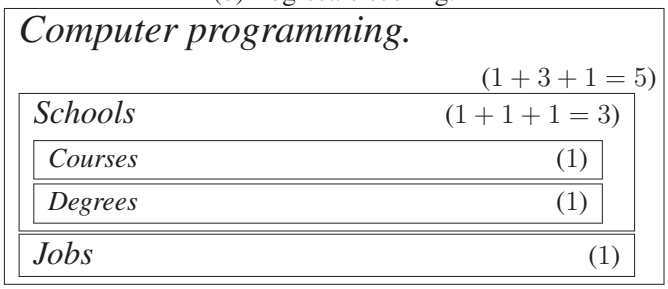
First, we propose four scoring methods for blocks in a document.



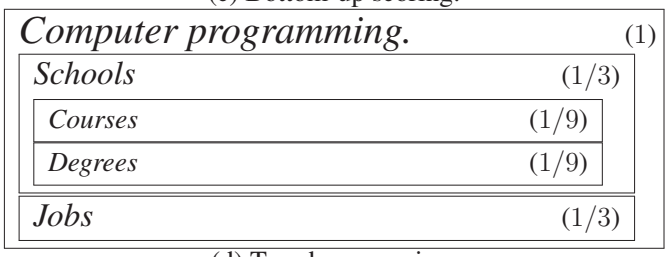
(a) Length scoring.



(b) Log-scale scoring.



(c) Bottom-up scoring.



(d) Top-down scoring.

Figure 4.2: Comparison of scoring results of page in Figure 4.1 by four scoring methods. Scores of blocks are in parentheses. Non-heading components of blocks are omitted.

Scoring by Content Length

Basically, the more description about a subtopic a document contains, the more important the subtopic is for the author of the document. Furthermore, because the author writes the document for readers, the importance of the subtopic for readers (and search engine users) is also reflected by the length of the content. Based on this idea, we can score blocks by the lengths of their contents. The score of a block b is:

$$\text{blockScore}(b) = \text{length}(b)$$

where length(b) is the length of b. We call this *length* scoring. For example, if we score the blocks in Figure 4.1 by this, the result is in Figure 4.2(a). In Figure 4.2, the scores of the blocks are in parentheses and non-heading components of the blocks are omitted.

Scoring by Log-scaled Content Length

As the relevance of a document to a query is assumed not to be direct proportion to the number of query keyword occurrences in the document [87], the importance of a topic may also be not direct proportion to the content length of the block referring to the topic. Based on this idea, we propose another scoring function with logarithmic scaling:

$$\text{blockScore}(b) = \log(\text{length}(b) + 1) \ .$$

We call this *log-scale* scoring. An example result of log-scale scoring is in Figure 4.2(b).

Bottom-up Scoring

In practice, the importance of some topics are not reflected by the content length of their matching blocks. For example, telephone number may be an important subtopic of a place, but blocks under the heading "telephone number" should contain relatively less contents, i.e., only the exact telephone number of the place, than blocks under other headings. Logarithmic scaling in the last section reduces the effect of content length. Furthermore, we can score blocks without using their content lengths. If we assume the even importance for all blocks excluding their child blocks, the score of a block b is formulated as below:

$$\text{blockScore}(b) = 1 + \sum_{c \in b} \text{blockScore}(c)$$

where c is each child block of b. We call this *bottom-up* scoring. An example result of bottom-up scoring is in Figure 4.2(c).

Top-down Scoring

On the other hand, we can assume the even importance for all child blocks of a block. This assumption means that child blocks of a block are used to segment its topic into multiple subtopics of the even importance. Because the original block may have meaningful contents besides its child blocks, we also assign the same importance to the contents. The score of a block b is:

$$\text{blockScore}(b) = \begin{cases} \frac{\text{blockScore}(p)}{1+|p|} & \text{if } \exists p.b \in p \\ 1 & \text{otherwise} \end{cases}$$

where $|p|$ is the number of the child blocks of p and $\exists p.b \in p$ means b has its parent block p. We call this *top-down* scoring. An example result of top-down scoring is in Figure 4.2(d).

4.3.2 Score Integration for Multiple Pages

Next, we explain four ways to integrate the scores of a subtopic string on multiple documents.

Simple Summation

The simplest way to integrate the scores for multiple pages is to sum them up. Such simple summation means that the importance of a subtopic string is reflected by the length of contents (if we adopt length scoring), the number of blocks (if we adopt bottom-up scoring), and so on that refer to the subtopic in a document set.

Formally, the score of a subtopic string s on a document set D is:

$$\text{score}(s, D) = \sum_{d \in D} \text{docScore}(s, d) \ .$$

We call this *summation* integration.

Page-based Integration

By summation integration, documents of more length or including more blocks have more chance to contribute to $\text{score}(s, D)$. However, documents are meaningful information unit as well as blocks are, and scaling of $\text{docScore}(s, d)$ may be useful for effective subtopic scoring. For the scaling, we divide $\text{docScore}(s, d)$ by $\text{blockScore}(\text{root}(d))$ where $\text{root}(d)$ is the root block in d, i.e., the block representing entire d. Because we score each block considering its hierarchical descendant blocks, $\text{blockScore}(b)$ is the maximum in a document when b is the root block of the document, and $\text{docScore}(s, d)$ is the maximum when s matches the root block of d. Therefore, this division scales the $\text{docScore}(s, d)$ to $[0, 1]$. After that, $\text{score}(s, D)$ is defined as below:

$$\text{score}(s, D) = \sum_{d \in D} \frac{\text{docScore}(s, d)}{\text{blockScore}(\text{root}(d))} \ .$$

We call this *page-based* integration.

Note that there is no difference between summation and page-based integration after top-down scoring because blockScore(b) in top-down scoring is already scaled to $[0, 1]$.

Domain-based Integration

Some authors may split a topic into multiple documents in a *domain*, e.g., multiple web pages under a domain name, instead of multiple blocks. Considering such case, domain-based scaling may be more effective than page-based scaling. To formulate such scaling, we introduce a domain Δ which is a subset of the document set D. The new integration function is:

$$\text{score}(s, D) = \sum_{\Delta \subset D} \frac{\sum_{d \in \Delta} \text{docScore}(s, d)}{\sum_{d \in \Delta} \text{blockScore}(\text{root}(d))} .$$

We call this *domain-based* integration.

Combination Integration

Both page-based and domain-based scaling are independently applicable. If we apply both of them, the new integration function is:

$$\text{score}(s, D) = \sum_{\Delta \subset D} \frac{1}{|\Delta|} \sum_{d \in \Delta} \frac{\text{docScore}(s, d)}{\text{blockScore}(\text{root}(d))} .$$

We call this *combination* integration.

4.3.3 Diversifying Subtopic Ranking

To rank multiple subtopic strings into a ranking, we can score each of them once, then simply sort the strings by descending order of their scores. We call this *uniform* ranking method.

However, because search result diversification is one of the most important applications of subtopic ranking, diversity of subtopic ranking is also important. Therefore, we also propose a diversification method of subtopic ranking. Our idea for diversification is that if a subtopic string matches a block, the topic of the block is already referred to by the subtopic string. Therefore, even if the block matches any other subtopic strings, the block should not contribute to the score of the subtopic strings.

Based on this idea, we propose a *diversified* ranking method for subtopic strings based on hierarchical heading structure. In this method, first we score each subtopic string on a document set then put only the string with the best score into the resulting ranking. Second, we remove all the blocks matching with the string from the document set. Third, we re-score the remaining subtopic strings on the remaining blocks then put the string with the best score into the resulting ranking. The second and third steps are repeated until all subtopic strings are ranked.

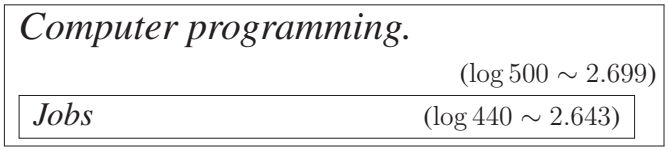


Figure 4.3: Example re-scoring result of page in Figure 4.1 by log-scale scoring after we rank first subtopic string "computer programming schools."

For example, suppose three subtopic strings, "computer programming school", "computer programming course", and "computer programming jobs." If we rank the strings by uniform ranking method and the log-scale scores of the blocks in Figure 4.2(b), the ranks of the strings are in the order above because the strings match the "Schools" (score: 3.398), "Courses" (score: 3.204), and "Jobs" (score: 2.643) blocks, respectively. On the other hand, if we rank the strings by diversified ranking method, "computer programming jobs" achieves the second rank because after "computer programming school" is ranked first, its matching block "School" including its descendant blocks is removed from the re-calculation of the scores (see Figure 4.3). Then the score of "computer programming course" in this page becomes 0 because the block referring to the subtopic in this page has already matched the higher ranked subtopic.

4.4 Evaluation

In this section, we evaluate and compare the baselines and our proposed methods.

We proposed four block scoring methods, four score integration methods, and two subtopic ranking methods. We can arbitrary combine these methods. However, there is no difference between summation and page-based integration and also between domain-based and combination integration when we use top-down scoring as discussed in Section 4.3.2. Therefore, we compare 28 proposed methods in total.

4.4.1 Evaluation Method

Because we do not discuss extraction methods of subtopic candidate strings, we evaluate our ranking methods by re-ranking the baseline subtopic rankings.

We use the official data set (including baselines) and evaluation measures of the NTCIR-10 INTENT-2 task subtopic mining subtask [88]. This is because the dataset of the latest NTCIR-12 IMine-2 task is not available yet, and because first-level and second-level subtopics are distinguished in the second-latest NTCIR-11 IMine task while our proposed methods do not distinguish them. All components of the NTCIR-10 data set is publicly available and most of them are

at the web site of NII[1].

In the subtopic mining subtask, participants are required to return ranked list of top-10 subtopic strings for each query. Subtopic strings are expected to sorted in descending order of their *intent probability*, i.e. the probability that search engine users submitting the given query need information on the subtopics (or *intent*). Multiple subtopic strings may refer to the same intent, but a string refers to one intent at most.

4.4.2 Evaluation Measures

Official evaluation measures of the subtask are intent recall (I-rec), D-nDCG, and D♯-nDCG.

The definition of the I-rec measure is:

$$\text{iRec@10} = |I'|/|I|$$

where I is a set of known subtopics of the original query, and I' is a set of subtopics represented by any of the maximum 10 strings in a ranking to be evaluated. The I-rec measure reflects the recall and diversity of subtopics in rankings.

The definition of the D-nDCG measure for a ranking of maximum 10 strings is:

$$\begin{aligned} \text{DnDCG@10} &= \frac{\text{DDCG@10}}{\text{ideal DDCG@10}} \\ \text{where DDCG@10} &= \sum_{r=1}^{10} \frac{\sum_i Pr(i|q) g_i(r)}{\log(r+1)} \end{aligned}$$

where r is a rank, $Pr(i|q)$ is the intent probability of a subtopic i behind the original query q, and $g_i(r)$ is 1 iff the string at the rank r refers to the subtopic i, and 0 otherwise. The D-nDCG measure reflects the precision and accuracy of rankings.

The integrated measure D♯-nDCG is the weighted summation of I-rec and D-nDCG.

$$\text{D♯nDCG@10} = \gamma \text{iRec@10} + (1-\gamma)\text{DnDCG@10}$$

where γ is the weight of I-rec which is fixed to 0.5 in this chapter and the subtask. In other words, D♯-nDCG is arithmetic mean of I-rec and D-nDCG.

An official evaluation tool is available online[2].

4.4.3 Data Set

The details of the data set is as explained below.

Queries We used 50 keyword queries in the NTCIR data set that are also used in the Text Retrieval Conference (TREC) 2012 Web track [27].

[1] http://www.nii.ac.jp/dsc/idr/en/ntcir/ntcir.html

[2] http://research.nii.ac.jp/ntcir/tools/ntcireval-en.html

Document Sets We used the baseline document rankings generated by default scoring of Indri search engine[3] (including query expansion based on pseudo-relevance feedback) and Waterloo spam filter for TREC 2012 Web track. The baseline rankings are available online[4]. Each ranking consists of 131–837 web pages from ClueWeb09B for a query.

Baseline Results There are the snapshots of query completion/suggestion results by commercial search engines prepared for the NTCIR-10 INTENT-2 task. We used the query completion results by Google and Yahoo because they achieved the best I-rec and D-nDCG scores respectively among the baselines [88]. Because the both results contain only 10 strings at most for each query, re-ranking of them do not affect the I-rec scores. Therefore, we also used our *merged* baseline result which is generated by merging four baseline query completion/suggestion results and sorting them in "dictionary sort" [88]. Because the meaning of dictionary sort is ambiguous, we could not reproduce their evaluation result. We merged the results, decapitalized the strings, removed duplicated strings, and sort the remaining strings in byte order in UTF-8 to generate our merged baseline result.

Known Intents and Intent Probabilities The known intents, subtopic strings referring to them, and intent probabilities of them are prepared for the subtask [88]. They are manually selected from a pool of subtopic candidate strings by multiple assessors according to the definition of subtopic string, and also manually filtered by ten assessors based on their importance. The estimated importance of the strings is also used as their intent probabilities. All the true subtopic strings in the baseline results must be in this data according to their annotation process.

4.4.4 Implementation Details

In this section, we explain the details of our implementation required to evaluate our methods.

Heading Structure Extraction To extract hierarchical heading structure in web pages, we use our heading-based page segmentation (HEPS) method proposed in Chapter 3. It extracts each heading and block in pages as an array of adjoining sibling DOM nodes. For evaluation, we used the reference implementation of HEPS 1.0.0[5].

Text Contents of Headings and Blocks We used the URL and title as the heading of each web page. As the text contents of the other headings, blocks, and entire pages, we use their corresponding raw strings (see Section 3.4.2). Before generating raw strings, each DOM IMG (image) nodes are replaced by its alternate text and URL, i.e., alt and src HTML attribute values.

[3] `http://www.lemurproject.org/indri/`
[4] `https://github.com/trec-web/trec-web-2014`
[5] `https://github.com/tmanabe/HEPS`

Table 4.1: D-nDCG score comparison with query completion by Google. Top-5 proposed methods are listed. For all methods and baseline, I-rec $= 0.3841$.

Scoring	Integration	Ranking	Score
log-scale	domain-based	uniformed	**.4502**
log-scale	combination	uniformed	.4501
log-scale	domain-based	diversified	.4487
log-scale	combination	diversified	.4485
bottom-up	page-based	diversified	.4479
Query completion result by Google			.3735

Content Length For length and log-scaled scoring, we used the number of characters in their raw strings as their length.

Domain For domain-based and combination integration, we distinguished the domains of web pages by the fully qualified domain names in their URLs.

Matching between Subtopic Strings and Headings Before matching subtopic candidates and hierarchical headings, we performed basic preprocessing for retrieval, e.g., tokenization, stop word filtering, and stemming, for both strings. All URLs were split by any non-word characters, and the other strings are tokenized by Stanford CoreNLP toolkit [68]. All tokens were decapitalized, filtered out 33 default stop words of Lucene[6], then stemmed by the Porter stemmer [84].

Subtopic Candidate Strings After preprocessing, duplicated subtopic candidate strings and subtopic candidate strings same as queries were removed.

Ties If we have multiple subtopic candidates of the same score in our unified ranking method or any iteration of our diversified ranking method, we sorted them in the baseline order.

4.4.5 Evaluation Results

Results are listed in Table 4.1, 4.2, and 4.3. Table 4.1 shows the D-nDCG scores achieved by each method when they re-rank the query completion by Google, and Table 4.2 shows the D-nDCG scores achieved by each method when they re-rank the query completion by Yahoo. In Table 4.1 and 4.2, top-5 proposed methods are listed in descending order of their D-nDCG scores. Table 4.3 shows the scores achieved by each method when they re-rank our merged baseline result. In Table 4.3, top-5 proposed methods are listed in descending order of their D♯-nDCG scores. In this comparison, the log-scale/summation/uniformed method achieved the best scores on all the measures among all the proposed methods including ones omitted from Table 4.3.

[6] `http://lucene.apache.org/`

Table 4.2: D-nDCG score comparison with query completion by Yahoo. Top-5 proposed methods are listed. For all methods and baseline, I-rec $= 0.3815$.

Scoring	Integration	Ranking	Score
log-scale	page-based	diversified	**.4617**
bottom-up	domain-based	diversified	.4609
log-scale	page-based	uniformed	.4608
log-scale	summation	diversified	.4601
length	domain-based	diversified	.4587
Query completion result by Yahoo			.3829

Table 4.3: Comparison with our merged baseline result. Top-5 methods are listed in descending order of their D♯-nDCG scores.

Scoring	Integration	Ranking	I-rec	D-nDCG	D♯-nDCG
log-scale	summation	uniformed	**.4009**	**.3997**	**.4003**
log-scale	page-based	uniformed	.3986	.3981	.3984
length	summation	uniformed	.3974	.3945	.3959
log-scale	combination	uniformed	.3956	.3921	.3939
log-scale	domain-based	uniformed	.3956	.3913	.3934
Our merged baseline result			.3310	.3066	.3188

4.4.6 Discussions

In all comparisons, our proposed methods achieved better score than baseline results. This is not because we proposed a number of methods and one of them achieved a better score than each baseline result by chance. For example, we focus on log-scale/page-based/diversified method which achieved the best D♯-nDCG score throughout this chapter by reranking the result by Yahoo. The method also achieved a better D-nDCG score (0.4470) than the result by Google and better I-rec, D-nDCG, and D♯-nDCG scores (0.3840, 0.3695, and 0.3768, respectively) than the merged result. Moreover, according to Student's paired t-test (where each pair consists of the scores of the baseline and our proposed method for a query), all the D-nDCG and D♯-nDCG scores were statistically significantly different from the baseline scores ($p < 0.05$). This fact supports the effectiveness of our proposed subtopic ranking methods. Only the I-rec score was not statistically significant ($p = 0.0656$). Hereafter in this chapter, we discuss statistical significance based on the same test procedure.

Comparison of Block Scoring Methods

Log-scale scoring achieved the best scores in all the three comparisons. This fact may suggest that the importance of a topic is reflected by the content length of the block referring to the topic, but the importance is not direct proportion to the length. Moreover, 11 among the 15 best results shown in Table 4.1, 4.2, 4.3 are using log-scale scoring. This fact may suggest the robustness of log-scale scoring. However, the advantage of log-scale scoring over the others was small. For example, the D-nDCG score of the Yahoo result reranked by the *log-scale*/page-based/diversified method was not statistically significantly different from the scores of the *bottom-up*/page-based/diversified ($p = 0.1481$), *top-down*/page-based/diversified ($p = 0.1204$), and *length*/page-based/diversified ($p = 0.0972$) methods.

The usefulness of content length also suggests a balance between information supply by page authors and information demand by page readers. However, because of the simple estimation method of intent probabilities, i.e., voting by ten assessors, the practicalness of this result is still not clear. More practically, query logs of commertial search engines must reflect intent probabilities of the queries, i.e., subtopic candidate strings. Evaluation on a more practical data set is a future issue.

Comparison of Score Integration Methods

The effect of score integration method selection was small. In the comparison with the Google result (Table 4.1), the log-scale/*domain-based*/uniformed method achieved the best D-nDCG score, but its difference from the second-best score which the log-scale/*combination*/uniformed method achieved was quite small (-0.0001). In the comparison with the Yahoo result (Table 4.2), the log-scale/*page-based*/diversified method achieved the best D-nDCG score, but its difference from the score of the log-scale/*summation*/diversified method was also small (-0.0016). In the comparison with our merged result (Table 4.3), the score differences between the best log-scale/*summation*/uniformed method and the second-best log-scale/*page-based*/uniformed method were also small.

Effect of diversified ranking method

Because I-rec measures the diversity of rankings, we focus on the I-rec score comparison with our merged result (Table 4.3). No method with diversified ranking achieved the top-5 scores. The top-down/combination/*diversified* method achieved the best I-rec score (0.3869) among the methods with diversified ranking. The I-rec score difference between the method and the best log-scale/summation/*uniformed* method was not statistically significant ($p = 0.2759$). The I-rec score difference between the best method and the log-scale/summation/*diversified* method

was also not statistically significant ($p = 0.1028$). As discussed above, our proposed ranking diversification method did neither improve nor worsen resulting rankings.

4.5 Summary

We proposed subtopic ranking methods based on hierarchical heading structure in documents. Our ideas are that hierarchical headings in a document reflect the topic structure of the document and that the length of contents referring to a topic reflects the importance of the topic. Based on these ideas, we proposed methods based on matching between the subtopic candidate strings and the hierarchical headings. We evaluated our methods by using the publicly available NTCIR data set. Our evaluation results indicated that (1) our methods significantly improved the baseline rankings by commercial search engines, that (2) log-scale scoring seems effective and robust, that (3) there is no substantial difference among score integration methods, and that (4) our ranking diversification method was not effective.

CHAPTER 5

HEADING-AWARE PROXIMITY MEASURE AND ITS APPLICATION TO WEB SEARCH

5.1 Introduction

Proximity of query keywords is an important factor for effective query-biased document scoring. It has been widely studied in the context of general document search, and many proximity-aware scoring functions have also been proposed [31, 93, 98]. Most of them consider *distance*, i.e., the difference of their positions, to measure proximity between two keyword occurrences. Less distant occurrences of query keywords in a document suggest more relevance of the document to the query. The distance, and also distance-based existing proximity-aware scoring functions, treat documents as arrays of term occurrences. However, most documents have logical structure, and the structure affects logical proximity between term occurrences. Therefore, we cannot measure logical proximity in structured documents by the distance.

Hierarchical heading structure is one prevalent type of such structure and the structure strongly affects proximity. For example, suppose we have three documents in which a query keyword X occurs n-term-occurrences-distant from the occurrence of another query keyword Y. In document A, X occurs in a heading and Y occurs in the block associated with the heading. In document B, both of the keywords occur in the non-heading component of a block. In document C, X occurs in the non-heading component of a leaf block (a block without child block) and Y occurs in the non-heading component of another leaf block. Among these documents, document A must be the most relevant to the query. This is because term occurrences in a heading describe the topic of the entire corresponding block with less regard for the distance between the occurrences

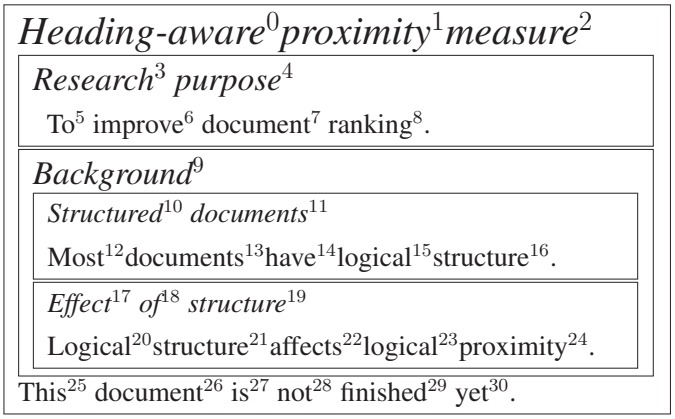


Figure 5.1: Example document with heading structure. Each rectangle encloses block, each text in italic font is heading, and each superscript number is position of term occurrence.

in the heading and in the block. Document C must be the least relevant to the query because the keywords occur in different blocks, i.e., partial documents referring to different topics.

In this chapter, we propose a heading-aware proximity measure which is useful for scoring documents with heading structure. It is based on proportional functions of distance and has different constants for each structural relationship between term occurrences. Our measure can be combined with many existing proximity-aware document scoring functions. In this chapter, we combine the measure with three existing proximity-aware document scoring functions.

We optimized and evaluated our measure and three functions with the data sets prepared for the Text Retrieval Conference (TREC) web tracks. On the training data set, we optimized the various parameter values of the measure and functions. On the test data set, we compared official baseline rankings, rankings generated by the existing proximity-aware scoring functions based on simple distance, and rankings generated by our proximity-aware scoring functions based on our heading-aware proximity measure.

5.2 Proximity Measures

In this section, we explain the simplest proximity measure, distance, and propose a heading-aware proximity measure, *heading-aware semi-distance*. Note that they measure proximity between two term occurrences, not two terms themselves, while some proximity-based functions shown later measure proximity between two terms themselves.

5.2.1 Distance

The simplest measure for proximity between two term occurrences is *distance* between them, that is,

$$dist(o_1, o_2) = |pos(o_1) - pos(o_2)|$$

where o_1 and o_2 are term occurrences in a document and $pos(o)$ denotes the position of o, i.e., serial number for all term occurrences in the document. For example, in Figure 5.1, $dist$(Heading-aware^0, Structured^{10}) $= 10$ and $dist$(proximity^{24}, document^{26}) $= 2$.

5.2.2 Heading-Aware Semi-Distance

As discussed in Section 5.1, distance cannot measure logical proximity between term occurrences in documents containing heading structure. To measure proximity of two occurrences o_1 and o_2 in such documents, we define a new measure with four parameters (namely a_{hc}, b_{hc}, a_{db}, and b_{db}), *heading-aware semi-distance*, denoted by $hasd$, as shown below:

$$hasd(o_1, o_2) = \begin{cases} dist(o_1, o_2) \cdot a_{hc} + b_{hc} & \text{if } hc(o_1, o_2) = \text{true ;} \\ dist(o_1, o_2) \cdot a_{db} + b_{db} & \text{if } db(o_1, o_2) = \text{true ;} \\ dist(o_1, o_2) & \text{otherwise.} \end{cases}$$

Heading and content In this measure, $hc(o_1, o_2)$ is true iff $o_1 \neq o_2$ and either o_1 or o_2 occurs in the hierarchical headings of a block in which the other occurs. In other words, $hc(o_1, o_2)$ is true iff the two occurrences are in relationship of a heading word and its corresponding content word. Note that it includes a case where both of the occurrences are in a heading and also another case where one occurrence is in the heading of a block and the other is in (either the heading or non-heading component of) a hierarchical descendant block of the block. For example, in Figure 5.1, hc(Heading-aware^0, measure^2) = true, hc(Heading-aware^0, Structured^{10}) = true, and hc(measure^2, Most^{12}) = true. Otherwise $hc(o_1, o_2)$ is false. Note that it includes a case where one occurrence is in the heading of a block and the other is in the non-heading component of a hierarchical ancestor block of the block. For example, in Figure 5.1, hc(structure^{19}, document^{26}) = false. According to our assumption, $a_{hc} < 1$ and $b_{hc} = 0$ so that $hasd(o_1, o_2) < dist(o_1, o_2)$ when $hc(o_1, o_2) =$ true.

Division by blocks On the other hand, $db(o_1, o_2)$ is true iff $o_1 \neq o_2$, $hc(o_1, o_2)$ is false, and o_1 and o_2 occurs in different blocks from each other. It includes a case where the blocks are an ancestor and one of its descendant. Otherwise $db(o_1, o_2)$ is false. For example, in Figure 5.1, db(Structured^{10}, Effect^{17}) = true, db(proximity^{24}, document^{26}) = true, db(Heading-aware^0, Structured^{10}) = false, and db(document^{26}, finished^{29}) = false. According to our as-

sumption, $1 < a_{db}$ and $0 \leq b_{db}$, or $1 \leq a_{db}$ and $0 < b_{db}$, so that $dist(o_1, o_2) < hasd(o_1, o_2)$ when $db(o_1, o_2) =$ true.

In this chapter, we assume $0 < a_{hc}$, $0 \leq b_{hc}$, $0 < a_{db}$, and $0 \leq b_{db}$ so that $hasd$ becomes a symmetric positive-definite function (or a semimetric). Note that $hasd$ does not necessarily satisfy the triangle inequality. For example, in Figure 5.1, $hasd$(Most^{12}, Structured^{10}) + $hasd$(Structured^{10}, structure^{16}) < $hasd$(Most^{12}, structure^{16}) when we set $a_{hc} < 0.5$ and $b_{hc} = 0$.

5.3 Proximity-Aware Scoring Methods

We introduce three existing distance-based document scoring methods. For each existing method, we also propose a modified method based on our heading-aware semi-distance.

5.3.1 MinDist

The $mindist$ proximity-based function is proposed by Tao and Zhai [98]. They also proposed four other functions, but concluded in their paper that $mindist$ is their best performing function. In their paper, $mindist$ is defined for a document (or an array of term occurrences) D and a query (or a set of keywords) Q. However, for consistency with the next scoring method P6, we re-define it for D and a pair of different query keywords, κ_1 and κ_2, without changing their scoring results. The function is defined by the formula:

$$mindist(\kappa_1, \kappa_2, D) = \min_{o_1 \in O(\kappa_1, D), o_2 \in O(\kappa_2, D)} dist(o_1, o_2)$$

where $O(\kappa, D)$ is a set of occurrences of κ in D.

Their proximity-based score π of D for Q is defined by:

$$\pi(Q, D) = \log \left[\alpha + \exp \left\{ - \min_{\kappa_1, \kappa_2 \in Q \cap D, \kappa_1 \neq \kappa_2} mindist(\kappa_1, \kappa_2, D) \right\} \right]$$

where α is a free parameter and $Q \cap D$ denotes a set of keywords in Q which also occurs in D.

Their final score of D for Q is the sum of the proximity-based score π and a non-proximity-based score. As the non-proximity-based score, we use a score given by Indri (explained later) with scaling. The final score is $s \cdot indri(Q, D) + \pi(Q, D)$ where s is a free parameter for scaling and $indri(Q, D)$ is the Indri score of D for Q. We call a document ranking method by this score the MinDist method.

5.3.2 P6

Cummins and O'Riordan developed the $p6$ proximity function by combining 12 basic proximity functions (including $mindist$) by using genetic programming [31]. On most data sets in their evaluation, $p6$ achieved the best mean average precision (MAP) scores among them. The $p6$

function is defined by the formula:

$$2 \cdot p6 = \left[\left\{3 \cdot \log\left(\frac{10}{mindist}\right) + \log\left(prod + \frac{10}{mindist}\right) + \frac{10}{mindist} + \frac{prod}{sum \cdot qt}\right\} / qt\right] + \frac{prod}{avgdist \cdot mindist}$$

where $qt = |Q \cap D|$ and $p6$, $mindist$, $prod$, sum, and $avgdist$ are all functions of (κ_1, κ_2, D). The definition of $mindist$ is the one introduced before. The others are:

$$\begin{aligned} prod(\kappa_1, \kappa_2, D) &= |O(\kappa_1, D)| \cdot |O(\kappa_2, D)| \,, \\ sum(\kappa_1, \kappa_2, D) &= |O(\kappa_1, D)| + |O(\kappa_2, D)| \,, \text{and} \\ avgdist(\kappa_1, \kappa_2, D) &= \frac{\sum_{o_1 \in O(\kappa_1, D), o_2 \in O(\kappa_2, D)} dist(o_1, o_2)}{prod(\kappa_1, \kappa_2, D)} \,. \end{aligned}$$

Their proximity-based score of D for Q is defined by:

$$S(Q, D) = \sum_{\kappa_1, \kappa_2 \in Q \cap D, \kappa_1 \neq \kappa_2} p6(\kappa_1, \kappa_2, D) \,.$$

Their final score of D for Q is the sum of the proximity-based score and a non-proximity-based score. We again use Indri score with scaling as the non-proximity-based score. The final score is $s \cdot indri(Q, D) + S(Q, D)$. We call the document ranking method by this score P6.

5.3.3 Expanded Span

Song et al. proposed an efficient method to segment a document into *expanded spans* that never overlap, include as many unique query keywords as possible, and have minimal widths [93]. They also proposed a scoring function f of an expanded span for a query keyword, a scoring function rc of a document for a query keyword, and a scoring function of a document for a query.

In this chapter, an expanded span is treated as a substring of D and denoted by $E = [o_i, ..., o_j]$. Note that both of o_i and o_j are occurrences of different query keywords because of the width-minimality of expanded span. The width of E is:

$$width(E) = \begin{cases} dist(o_i, o_j) + 1 & \text{if } 1 < |E| \text{ and} \\ M & \text{otherwise} \end{cases}$$

where M is a parameter and $0 < M$. The score of a span E for a query keyword κ is:

$$f(\kappa, E) = \begin{cases} \left\{\frac{|Q \cap E|}{width(E)}\right\}^x \cdot |Q \cap E|^y & \text{if } 0 < |O(\kappa, E)| \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

where $|Q \cap E|$ is number of query keywords (not limited to κ) that occur in E, and x and y are free parameters. The score of D for κ is $rc(\kappa, D) = \sum_{E \subset D} f(\kappa, E)$ and the score of D for Q is:

$$\sum_{\kappa \in Q} \frac{(k_1 + 1) \cdot rc(\kappa, D)}{k_1 \cdot \{(1 - b) + b \cdot |D| / avdl\} + rc(\kappa, D)} \log \frac{N - n(\kappa) + 0.5}{n(\kappa) + 0.5}$$

where k_1 and b are parameters, $avdl$ is the arithmetic mean of $|D|$, N is the number of all documents, and $n(\kappa)$ is the number of documents that contain κ. We call document ranking method by this score the Span method.

5.3.4 Heading-Aware Methods

We can easily combine our heading-aware semi-distance with these three methods MinDist, P6, and Span, by replacing $dist$ in them by our $hasd$. We call the heading-aware versions of these three methods, the heading-aware MinDist method (HA-MinDist), the heading-aware P6 method (HA-P6), and the heading-aware Span method (HA-Span).

In HA-Span, a proximity measure for two adjoining query keyword occurrences is also required for the segmentation of a document into expanded spans, and we used our $hasd$ also for this segmentation. See their original paper for the details of this segmentation [93].

5.4 Implementation Details

In this section, we explain some details of our implementation of the proximity-aware document scoring methods. Note that our heading-aware semi-distance is independent from the features of web pages and that it may be applicable to search for general documents containing heading structure, e.g., books and news articles.

To apply the scoring methods to a web page, we need to extract D, i.e., an array of term occurrences in the page. For non-heading-aware methods, we extract all DOM text nodes from each web page, join the contents of the nodes in document order into a string, then replace adjoining whitespace characters in the string by a single space. We ignored descendant text nodes of `STYLE` or `SCRIPT` HTML elements. When joining text contents, we insert a space between them. Finally, we tokenize the string into D by the Treat implementation[1] of the Stanford Penn-Treebank style tokenizer. For heading-aware methods, we split the string at the borders of headings and blocks. This is because headings and blocks are coherent information unit and a term occurrence must not partially overlap headings or blocks. After that, we tokenize the strings. The other processes are same as those for non-heading-aware methods.

We score D for Q after removing 33 default stop words of Apache Lucene[2] from Q, tokenizing Q by the same implementation, and stemming all occurrences in D and Q by the Treat implementation of the Porter stemming algorithm [84].

[1] `https://github.com/louismullie/treat`
[2] `https://lucene.apache.org/`

5.5 Data Sets and Evaluation Measures

For optimization of the parameter values of the methods and also for evaluation of the methods, we used data sets prepared for TREC 2013–2014 web tracks [28, 29]. In this section, we explain the data sets and evaluation measures.

Queries and intents They have 50 keyword queries (topics) for each year. There are one or more intents (subtopics) behind each query. We split the total 100 queries into training and test sets. Because the queries for each year are biased (for example, there are only six intents marked as navigational behind 2014 queries while 38 behind 2013) we split them based on their topic IDs. We used the even-numbered 50 queries for training and the odd-numbered 50 for test.

Document collection They have a huge web collection ClueWeb12, which is crawled by the Lemur Project in 2012. Because of our resource limitation, we used its subset ClueWeb12B, which is also a TREC official collection. The collection contains about 52 million web pages.

Baseline scoring and ranking The 2014 official baseline rankings were generated by default scoring of the Indri search engine (including query expansion based on pseudo-relevance feedback) and Waterloo spam filter. Note that they also include rankings for the 2013 queries. We only used top-200 documents in the rankings because it requires much computation resource to score all the documents in the document collection or even the entire rankings and because we have to repeat it many times for parameter optimization.

Relevance judgment data We used the TREC official graded relevance judgment data of the documents to the intents. Note that it includes relevance judgment for entire ClueWeb12 and this fact affects the values of *ideal* DCG@20 explained below.

Evaluation measures We used four TREC official and well-known evaluation measures, namely ERR-IA@20, α-nDCG@20, NRBP, and MAP-IA, for evaluating a document ranking for one or more intents behind a query. We used the TREC official code, ndeval.c, to calculate scores of these measures[3]. The definition of ERR@20 is:

$$\begin{aligned} \mathrm{ERR@20} &= \sum_{i=1}^{20} \frac{R(g_i)}{i} \prod_{l=1}^{i-1} \{1 - R(g_l)\} \text{ , where} \\ R(g_i) &= \frac{2^{g_i} - 1}{16} \end{aligned}$$

where $g_i \in \{0, 1, 2, 3, 4\}$ is the graded relevance of the ith document in the ranking to an intent. The definition of P@i is the ratio of relevant documents to top-i documents in the ranking. The

[3] https://github.com/trec-web/trec-web-2014/

ith document is considered relevant iff $0 < g_i$. The definition of AP is the arithmetic mean of P@i where ith document is relevant. The *ERR-IA@20* and AP-IA measures are respectively arithmetic mean of ERR@20 and AP scores for all intents (with at least one relevant document) behind the query. The ERR-IA@20 measure is better at measuring effectiveness for navigational intents [22]. The definition of α-*nDCG@20* [23] is:

$$\begin{aligned} \alpha\text{-nDCG@20} &= \frac{\text{DCG@20}}{ideal\ \text{DCG@20}}\ , \text{where} \\ \text{DCG@20} &= \sum_{i=1}^{20} \frac{G(i)}{\log_2(1+i)}\ , \text{and} \\ G(i) &= \sum_{j=1}^{m} \frac{J(i,j)}{2^{C(i,j)}}\ . \end{aligned}$$

The number of intents behind the query is denoted by m, $J(i,j)$ is 1 iff the ith document in the ranking is relevant to the jth intent behind the query, i.e., $0 < g_i$ about j, and is 0 otherwise, $C(i,j)$ is number of documents ranked higher than i and relevant to the jth intent. The definition of *NRBP* is:

$$\text{NRBP} = \frac{m}{4} \sum_i \frac{G(i)}{2^{i-1}}\ ,$$

which is better at measuring effectiveness for ambiguous and underspecified queries [24].

To integrate the evaluation scores for multiple queries, we calculated arithmetic mean of the scores. Especially, we call arithmetic mean of AP-IA scores *MAP-IA*.

5.6 Parameters and Fine Tuning with Training Data

In this section, we explain the optimization method we used for parameter values of the methods, explain optimized parameter values, and discuss meanings of the values.

5.6.1 Optimization Method

For the optimization, we used the 50 training queries and adopted the simple *Coordinate Ascent* algorithm [69], which optimize only one parameter at a time. This is because optimization method is not the main topic of this dissertation. To avoid local maximum, we check three smaller and three larger candidate values than the current value. If the best score among the six candidates is better than the current best score, we adopt the candidate as a new current value then check new six candidates around the value. If not, we optimize the next parameter. Next to the last parameter, we optimize the first parameter again. If we cannot further improve the score by changing any of parameters, we quit the optimization.

The target function is set to MAP-IA. This is because the other measures tend to (almost)

Table 5.1: Optimized parameter values of our $hasd$.

Method	a_{hc}	b_{hc}	a_{db}	b_{db}
HA-MinDist	.450	0	1.50	3
HA-P6	.600	0	1.70	36
HA-Span	.800	3	.800	30

Table 5.2: Optimized values of other parameters.

Method	s	α	M	x	y	k_1	b
MinDist [98]	2.83	.420					
HA-MinDist	2.83	.297					
P6 [31]	256						
HA-P6	256						
Span [93]			54	.250	1.35	3.20	.250
HA-Span			27	.250	.800	.800	.350

ignore lower-ranked documents even though they are important in the middle of the greedy optimization process.

Median of initial values are: $a_{hc} = 0.30$, $b_{hc} = 0$, $a_{db} = 1.00$, $b_{db} = 15$, $\alpha = 1.00$, $s = 1.00$, $M = 45$, $x = 0.25$, $y = 0.30$, $k_1 = 0.40$, and $b = 0.30$. These values are taken from the original papers of the existing methods or reasonably selected by us. Initial values are randomly selected from these median values, five smaller values, and five larger values.

When we change α, s, or k_1 values in the optimization, we multiply it by $2^{-\frac{1}{4}}$ or $2^{\frac{1}{4}}$ because they are expected to be larger than 0 and their optimized values may be dozens of times higher than the initial value. When changing a_{hc}, a_{db}, x, y, or b values, we add -0.05 or 0.05 to the value. When changing b_{hc}, b_{db}, or M values, we add -3 or 3 to the value.

Considering the meanings of parameters, we defined the ranges of some parameters and did not check values outside them. Because $hasd$ must be larger than 0, we let $0.05 < a_{hc}$, $0 \leq b_{hc}$, $0.05 < a_{db}$, and $0 \leq b_{db}$. We also let $0 \leq b \leq 1$. The values of $avdl$, N, and $n(\kappa)$ in (HA-)Span were computed by using the ClueWeb12B and the older ClueWeb09B collections for robust estimation. The value of $avdl$ was about 1,650 and the value of N was about 103 million.

5.6.2 Optimized Parameter Values

Table 5.1 lists the parameter values of our $hasd$ measure optimized for each of our heading-aware method. Optimized values of the other parameters are listed in Table 5.2. They are derived from 64 initial settings for each method.

5.6.3 Discussions

First we discuss the parameter values of $hasd$ (see Table 5.1).

Heading and content About both the HA-Mindist and HA-P6 methods, optimized values of $a_{hc} \leq 0.6$ while b_{hc} was 0. It means logical distance between a pair of term occurrences in a heading and in its corresponding block is evaluated to be smaller than the simple distance. These settings follow our observation introduced in Section 5.1. However, about the HA-Span method, optimized values of a_{hc} was 0.8 while b_{hc} was 3. When $hc(o_1, o_2) =$ true and $a_{hc} = 1$ and $b_{hc} = 0$, $hasd$ is equal to $dist$ and 0.8 is close to 1 and 3 is close to 0. Therefore, it means no significant difference between $dist$ and $hasd$.

Divided by blocks On the other hand, optimized values of a_{db} and b_{db} were inconsistent over three methods. For HA-MinDist, optimized a_{db} was 1.5 but b_{db} was 3. This setting means term occurrences separated by block borders are logically more than 1.5 times distant than the simple distance. For HA-P6, a_{db} was 1.7 and b_{db} was 36. This means term occurrences separated by block borders are logically far more than 1.7 times distant than the simple distance. For HA-Span, a_{db} was 0.8 and b_{db} was 30. Because the value of M (maximum $hasd$ between adjoining term occurrences in the same span), is set to 27, this setting substantially means that term occurrences in different blocks are logically enough distant, and that there is no necessity of considering proximity between occurrences in different blocks. All these settings follows our observation introduced in Section 5.1.

As discussed above, most obtained parameter values of $hasd$ followed our assumptions. This fact supports the validity of our idea and definition of $hasd$ proximity measure.

Next we discuss the other parameter values (see Table 5.2). The values of s for P6 and HA-P6 methods were optimized to 256. This fact means the effect of the $p6$ function was quite limited and the resulting rankings were almost same as the baseline. This should be because the $p6$ function itself is already optimized by using genetic programming for a certain data set [31].

The original papers of the MinDist and Span methods introduced parameter values already optimized for certain data sets. The value of α in the $mindist$ function is 0.3 [98] and the values of M, x, y, k_1, and b in the Span method are respectively 45, 0.25, 0.3, 0.4, and 0.3 [93]. However, according to our preliminary evaluation with test data, our optimized parameter values consistently achieved better scores than the original parameter values on all the evaluation measures. Therefore, hereafter in this chapter, we use our optimized parameter values. This fact also supports usefulness and robustness of the optimization method.

Table 5.3: Comparison of average evaluation scores.

Method	ERR-IA@20	α-nDCG@20	NRBP	MAP-IA
Baseline	.310	.364	.285	.016
MinDist [98]	.298	.360	.270	.017
HA-MinDist	.335	.392	.304	.018
P6 [31]	.307	.367	.277	.017
HA-P6	.309	.368	.280	.017
Span [93]	.402$*$	.440	.383$*$	.020
HA-Span	**.436**$*$	**.470**$*$	**.418**$*$	**.021**$*$

$*$ statistically significantly different from baseline ($p < 0.05$)

5.7 Evaluation with Test Data

Last of all, we explain the evaluation result of web page rankings by the methods.

5.7.1 Evaluation Method

We used the 50 test queries for evaluation. As explained before, we only scored and ranked top-200 documents in the baseline rankings for the queries (total 10,000 tuples of a document, a query, and the baseline score of the document for the query). In other words, we evaluated the methods by re-ranking. All parameters are set by the values listed in Table 5.1 and 5.2.

5.7.2 Evaluation Result

Table 5.3 lists the evaluation scores of all pairs of a measure and a method. In this table, each asterisk means statistically significant difference from the baseline ($p < 0.05$) according to Student's paired t-test where each pair consists of the evaluation scores of two methods for a query. Hereafter in this chapter, we discuss statistical significance based on the same test procedure.

5.7.3 Discussions

Comparison with baseline scores First, we compare each score with the baseline score on the same evaluation measure. As shown in Table 5.3, for 10/12 pairs of one of our heading-aware methods and an evaluation measure, our methods achieved the better scores than the baseline scores. For 7/12 pairs of an existing method and an evaluation measure, existing methods also improved the baseline scores. However, according to the t-test procedure, only the improvements by the existing Span and our heading-aware Span methods were statistically significant on some of the measures. Especially, our HA-Span method statistically significantly improved the baseline rankings on all the measures. Moreover, our HA-Span method also achieved the best scores

among all the methods on all the measures. These facts strongly supports the validity of our assumptions about the effects of heading structure to logical proximity and the effectiveness and robustness of our $hasd$ measure and HA-Span method for query-biased document ranking.

Comparison with the existing methods Next, we compare the scores of our heading-aware methods with their corresponding existing methods on one of the evaluation measures. As shown in Table 5.3, all our heading-aware methods consistently achieved the even or better scores than the corresponding existing methods on all the evaluation measures. In details, our heading-aware methods improved the scores for 11/12 tuples of an existing method, its corresponding heading-aware method, and an evaluation measure. This fact supports the effectiveness and robustness of our $hasd$ measure. However, because of the small number of queries, the difference between the scores of our methods and the corresponding existing methods were not statistically significant except for ERR-IA@20 and α-nDCG@20 of HA-MinDist and MinDist.

5.8 Summary

In this chapter, first we explained our assumptions about the effect of heading structure to logical proximity. When compared to the simple distance, a term occurrence in a heading is logically closer to another term occurrence in the block corresponding to the heading, and in other cases, a term occurrence in a block is logically more distant from another term occurrence in another block. Based on the assumptions, we proposed a heading-aware proximity measure and our heading-aware variants of three existing proximity-aware document scoring methods. We then optimized the parameters of the methods, and finally evaluated all the methods by using TREC data sets. Most of optimized parameter values of the functions and evaluation scores supported the validity of our assumptions and the robustness and usefulness of our $hasd$ measure and heading-aware Span method. Especially, HA-Span achieved the best scores among all the methods on all the evaluation measures. Moreover, the score differences from the baseline scores were all statistically significant. These facts also support the effectiveness and robustness of HA-Span for query-biased document ranking.

CHAPTER 6

HEADING-AWARE BLOCK-BASED WEB SEARCH

6.1 Introduction

Web pages are not just plain text data but they contain various types of regional structures. Some types of regions are expected to refer to just one topic. For example, document object model (DOM) nodes specified by certain HTML tags (e.g., P tags for marking up paragraphs) and most panes within web pages separated by margins are considered to be such a region. Since a web page very often refers to multiple topics and only a few of them are relevant to users' search intents, region-based retrieval is expected to improve the accuracy of web page search systems. Such region-based retrieval is usually called passage retrieval. Typical passage retrieval systems extract documents containing at least one region including most query keywords then score the documents focusing only on that region.

In this chapter, we continuously focus on a type of regional structure, called *hierarchical heading structure*. This structure is composed of nested logical blocks associated with *headings*. Each heading is a highly summarized description of the block's topic. We show the hierarchical heading structure in the example page in Figure 6.1, in Figure 6.2. In Figure 6.2, each block (including the entire page) is enclosed by a rectangle, and its heading is assigned a subscript representing its logical level.

Hierarchical heading structure has three features important for passage retrieval. First, blocks in our definition are expected to be better units for passage retrieval than other regions. In our definition, a block must have an associated heading. It means that we only consider blocks that

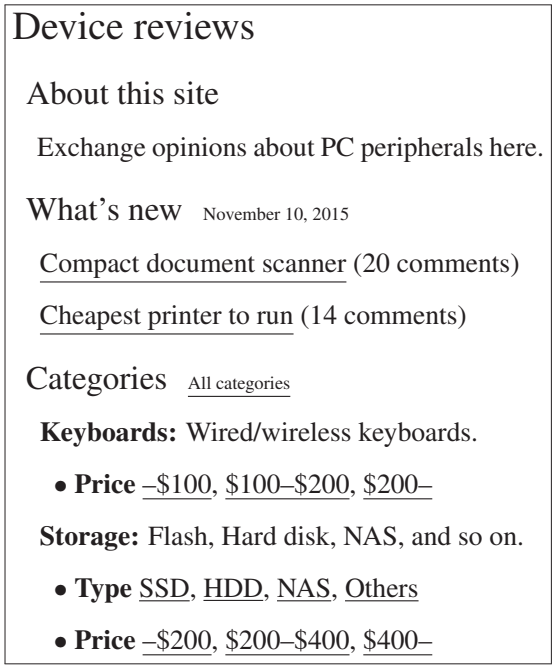


Figure 6.1: Page with hierarchical heading structure.

are large enough and yet coherent enough to be summarized into the headings.

Second, keyword occurrences in headings indicate the relevance of their associated blocks to the user's intent more strongly than occurrences in other components of the blocks. This is because authors tend to include important words for blocks into their headings so that the headings become highly summarized descriptions of the topics of the blocks.

Third, a heading describes the topic of any component of its associated block, regardless how distant the component is from the heading. Specifically, a heading of a block restricts topics of all the descendant blocks in the hierarchical document structure. For example, the entire web page shown in Figure 6.1 is also a block about its topic "Reviews of devices", and it contains blocks about some sub-topics, such as "Categorized reviews of devices" and "Reviews of devices in the keyboard category." As shown in this example, the topic of each block is hierarchically described by its own heading and all its hierarchical ancestor blocks' headings.

Based on the features of hierarchical heading structure explained above, we developed a heading-aware search system for web pages. Our system produces a search result in three steps. First, we extract implicit hierarchical heading structures from the explicit DOM tree structures of web pages by using our extraction method proposed in Chapter 3. Second, we retrieve blocks whose headings, contents, and ancestor blocks' headings seem relevant to the query intent. We

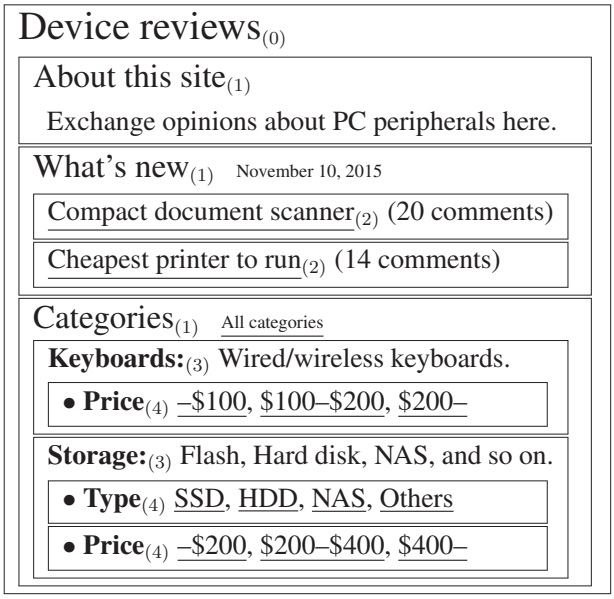


Figure 6.2: Hierarchical heading structure of page in Figure 6.1. Each rectangle encloses block, each text with subscript is heading and each subscript number represents logical level.

retrieve blocks instead of pages because each block has its individual topic as each page has. Third, we score and rank the retrieved blocks. When scoring a block, we consider not only its contents but also its ancestors' headings. Moreover, we assign different weights to its own heading, its parent heading, and the other ancestor headings because they have different logical distances from the block.

The remainder of this chapter is organized as follows. We explain our four proposed methods for block-level Boolean retrieval in Section 6.2, and our method for heading-aware block ranking in Section 6.3. In Section 6.4, we explain the results of our experiments evaluating our proposed methods. Finally, we conclude this chapter in Section 6.5.

6.2 Block-Level Boolean Retrieval Methods

As explained in Section 6.1, our search system produces a query result in two steps: it first retrieves blocks that seem relevant to the query intent then scores them. In this section, we explain how to determine whether each block in web pages is relevant to the query intent when the hierarchical heading structure of the pages is given. This block-level Boolean retrieval method is also effective in evaluating each page because a page without retrieved blocks should not be retrieved as an entire page.

We first consider the retrieval of relevant blocks by simple Boolean AND search, i.e., we

retrieve blocks that contain all the query keywords. In addition to the simple AND search of blocks, we consider two extensions that use the hierarchical heading structure of pages: removal of the contents of child blocks and inclusion of keyword occurrences in the headings of ancestor blocks. Because we can independently apply these two extensions, we compare simple block-level Boolean AND search with three extended methods.

6.2.1 Block-Level Boolean AND Search

We retrieve relevant blocks by using Boolean AND search based on the following assumption.

Assumption 5
Since each block in a heading structure is a coherent information unit, readers do not need to read other blocks to understand the contents of a block.

Generally, readers do not always read an entire page. They may scan only headings in the page, which describes the topics of blocks in the page, then only read blocks that seem relevant to their informational needs. In fact, some popular web sites, e.g., Wikipedia[1], provide tables of contents at the beginning of long pages. A table of contents of such a page consists of all headings in the page, and when a reader clicks on a heading in it, the browser window immediately scrolls to the block associated with the heading. Such a function helps readers scan the headings to only read relevant blocks. The authors of such pages are aware of such reader behaviors; therefore, they tend to write self-contained blocks. Of course, some types of documents are not designed in that manner, but we assume that many web pages are.

When a user inputs a query composed of multiple keywords into a search system, the user expects the system to return documents including the keyword occurrences related to each other. With the ordinary page-level Boolean AND search method, therefore, it is assumed that there is a relationship between any pair of terms in a page. As discussed above, however, web pages may be structured by headings, and each keyword may occur in unrelated blocks and may have no logical relationship to each other. For example, if a user submits the query “compact printer” to find information about compact printers, a typical search engine returns the page in Figure 6.1. However, this page contains no blocks or information about compact printers. To eliminate such irrelevant blocks and pages, we use block-based Boolean AND search in the retrieval step. The simplest way is to retrieve blocks containing all keywords in their contents, but we consider two extensions as explained in the following sections.

[1] Wikipedia `http://www.wikipedia.org/`

6.2.2 Removal of Child Blocks

The *full contents* of a block contains the contents of its child blocks. A reader of the block, however, may skip reading some child blocks if the reader can determine that they are irrelevant by reading only their headings. There may also be a reader who reads a child block without reading the *direct contents* of its parent block (i.e., its contents excluding the contents of its child blocks) if the reader can locate the child block including the necessary information only by scanning the headings. Based on these observations, we introduce another assumption below.

Assumption 6
Some blocks do not require the readers to read even their parent or child blocks.

This assumption means that terms in the direct contents of some blocks have no relationship to those in their child blocks. If there are many such blocks, we should only use terms in the direct contents of blocks when executing block-level Boolean AND search. That is, we should only retrieve blocks whose direct contents contain all the keywords.

In the web page in Figure 6.2, there is no block whose direct contents contain both "compact" and "printer." Therefore, our heading-aware search system can correctly eliminate all blocks (including the entire page) from the result to the example query, "compact printer."

6.2.3 Addition of Ancestor Headings

According to our observation, headings are an exception of the two assumptions above.

Assumption 7
To understand the contents of a block, readers need to read the headings of its hierarchical ancestor blocks, although the headings are located outside the block.

This is because a block is included in its ancestor blocks; therefore, the headings of the ancestor blocks also restrict the topic of the block. In many cases, the terms occurring in the headings of the ancestor blocks are completely omitted from the contents of the block. Note that this assumption does not contradict with Assumption 6. Even when a reader skips reading the direct contents of a block, the reader reads the block's heading; therefore, the reader can understand the contents of its child blocks.

For example, in the page shown in Figure 6.1, there are two blocks with the same heading term "Price." However, we can determine that the first one is about prices of keyboards and the other is about prices of storage, although there is no description of the product category within the "Price" blocks. This is because the headings of their parent blocks, "Keyboards:" and "Storage:", restrict the topics of their descendant blocks.

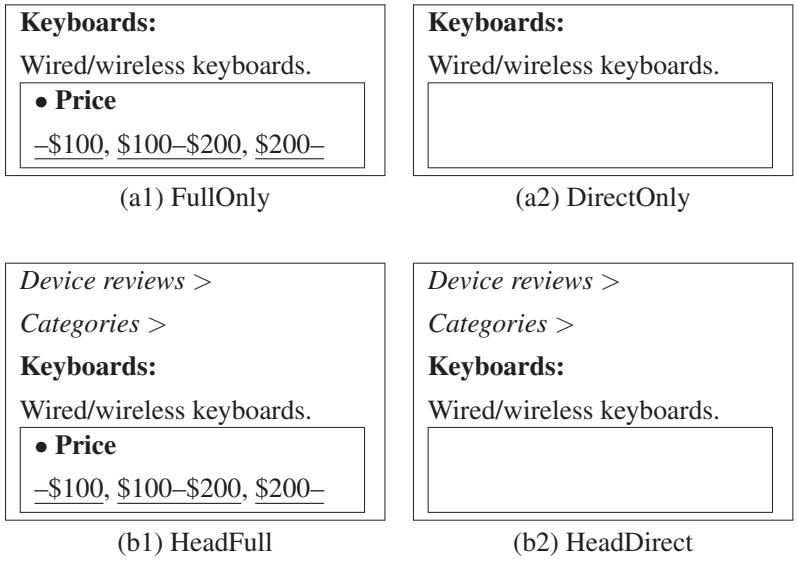


Figure 6.3: Target text of "Keyboards" block in Figure 6.2 generated with each method for block-level Boolean AND search. Ancestor headings are in italic and are delimited by ">."

If we assume the assumption above, we should add the headings of the ancestor blocks to the contents of each block. They include the headings of a page, namely its title and URL.

6.2.4 Four Proposed Block-Level Boolean Retrieval Methods

Because the two extensions above can be applied independently, we can consider four block-level Boolean retrieval methods listed below: *FullOnly* retrieves blocks whose full contents include all the keywords, *DirectOnly* retrieves blocks whose direct contents include all the keywords, *HeadFull* retrieves blocks such that all keywords occur either in the headings of their ancestor blocks or their full contents, and *HeadDirect* retrieves blocks such that all keywords occur either in the headings of its ancestor blocks or their direct contents. For example, in Figure 6.3, we show the target text of Boolean AND search generated from the "Keyboards:" block in Figure 6.2 with each of the four methods for block-level Boolean retrieval. For good readability, we omit the page URL from the top-level heading, indicate ancestor headings in italic, and separate each heading by the delimiter ">."

6.3 Heading-Aware Block Ranking Method

In this section, we discuss our heading-aware block scoring and ranking method that improves web page scoring and ranking by exploiting the hierarchical heading structure.

6.3.1 Basic Block Ranking

Most existing keyword-based scoring functions score documents mainly based on the number of keyword occurrences in the documents. However, if a search system simply counts the number of occurrences, documents including more term occurrences have more possibility to gain higher scores than shorter documents. Many scoring functions normalize scores by document length to avoid this. This normalization, however, may give a falsely lower score to documents with many topics than documents with a few topics, as explained below.

In general, page authors may discuss multiple and/or hierarchical topics in a single web page. For example, the page in Figure 6.1 includes hierarchical blocks discussing reviews of devices, newly arrived reviews, and reviews of devices in the keyboard category. In addition, in real web pages, there are many regions not related to the page topic, such as menus and advertisements.

Suppose we have another page whose contents are the same as those in the page in Figure 6.1 but without a "Categories" block. For the query "compact scanner", this page gains a higher score than the former because each keyword occurs once in both pages, and the former page is longer than the latter. These pages, however, should achieve the same score because only the "Compact document scanner" blocks in them are relevant to the query and they are exactly the same.

To solve this problem, Cai et al. [17] proposed a block ranking method called *block rank* (BR). Their method extracts logical blocks (in their definition) from web pages, scores each block by its full contents, then ranks the documents by the best score of the blocks in the document. Because each logical block corresponds to one topic, BR can score each page by focusing only on one topic in the page. We also adopt the basic idea of BR.

6.3.2 Heading-Aware Block Ranking

As discussed in Section 6.2.4, to understand the contents of a block, we need to read the headings of its ancestor blocks. Therefore, we also use the ancestor headings to score a block.

To score blocks considering their ancestor headings, we adopt a variant of BM25F [83]. With BM25F, we can score documents composed of multiple *fields*, which can be any partial content or metadata of the documents. For example, in the original paper on BM25F, the authors used the page title, anchor text of in-links, and body text as fields of web pages [86]. The BM25F ranking function calculates the score of a block β for keyword query q by the following formula:

$$\mathrm{BM25F}(q, \beta) = \sum_{\kappa \in q} \frac{\mathrm{weight}(\kappa, \beta)}{k_1 + \mathrm{weight}(\kappa, \beta)} \log \frac{N - \mathrm{bf}(\kappa) + 0.5}{\mathrm{bf}(\kappa) + 0.5}$$

where κ is a keyword in q, k_1 is a parameter to modify the scaling of term frequency, N is the number of all blocks, and $\mathrm{bf}(\kappa)$ is the number of blocks containing κ in any field. The weighted

term frequency, weight(κ, β), is defined as

$$\text{weight}(\kappa, \beta) = \sum_{f \in \beta} \frac{\text{occurs}(\kappa, f, \beta) \cdot boost_f}{\left((1 - b_f) + b_f \cdot \frac{\text{length}(f,\beta)}{\text{avgLength}(f)}\right)}$$

where f is a field in β, occurs(κ, f, β) is the number of occurrences of κ in f of β, $boost_f$ is the weight of keyword occurrences in f, b_f is the parameter in $[0, 1]$ to modify the strength of length normalization for f, length(f, β) is the length of f of β, and avgLength(f) is the arithmetic mean of the length of f.

As shown in the formula above, we can assign different weights to keyword occurrences in different fields. This has been found to be effective for some data sets. For example, Robertson et al. [86] achieved the best Precision@10 score for the TREC 2002 web track topic distillation task by $boost_{\text{title}}$ = 50 and $boost_{\text{anchor-text}}$ = 20 while $boost_{\text{body-text}}$ = 1.

We also use ancestor headings as fields of a block because all headings describe the topics of their descendant blocks. The strength of the relationship between a heading and a descendant block may be different depending on the difference in their depth. Therefore, we split ancestor headings of a block into the following four fields: page title, page URL, its parent's heading, and the other ancestors' headings.

We also treat the own heading of a block as a field because keyword occurrences in it strongly indicate the relevance of the block, as discussed in Section 6.1. Note that the terms in the own heading are included both in this new field and also in the body text field.

We could also split the full contents of a block into direct contents and contents of its child blocks. In addition, based on the hierarchical heading structure, we could define many more fields such as the direct contents of the parent block, full contents of the preceding sibling block, and headings of the child blocks. However, we only included five fields discussed above because the optimization of parameters for more fields becomes more difficult.

6.3.3 Parameters and Fine Tuning

The BM25F function has the following parameters: $boost_f$ for each field, b_f for each field, and k_1. Because the value of $boost_f$ are relative to each other, we can fix one of them. Therefore, we need to optimize these $2n$ parameters when we have n fields.

To optimize such a number of parameters, we adopt the simple *Coordinate Ascent* algorithm [69]. The target function for the optimization method is nDCG@20 (explained later in Section 6.4.3). When we revise k_1 or $boost_f$ in the algorithm, we multiply it by $2^{-\frac{1}{4}}$ or $2^{\frac{1}{4}}$ because their optimized values may be dozens of times higher than the initial value [86]. When revising b_f, we add -0.1 or 0.1 to the value. Because of our time limitation, we limit the values of k_1 and $boost_f$ in (0, 128) and b_f in [0, 1]. To avoid local maximum, we check the three smaller

and three larger values than the current value. If the best value of nDCG@20 among the six candidate parameter values is higher than the current best value, we adopt the candidate as a new parameter value. If not, we check the next parameter. Next to the last parameter, we check the first parameter again. If we cannot further improve the value of nDCG@20 by revising any of the parameters, we quit the optimization. The pseudo code of the entire optimization method is shown below.

Input: initial parameter values
$[v_{k_1}, v_{b_{f_1}}, \ldots, v_{b_{f_n}}, v_{boost_{f_1}}, \ldots, v_{boost_{f_{n-1}}}]$
1: $cur := k_1$, $last := k_1$
2: $candidates := [v_{-3}, \ldots, v_{-1}, v_{cur}, v_{+1}, \ldots, v_{+3}]$
3: $results := [n_{-3}, \ldots, n_{-1}, n_{cur}, n_{+1}, \ldots, n_{+3}]$
where n_i is the nDCG@20 value we obtain if we adopt v_i
4: jump to 8 if n_{cur} is the maximum value in $results$
5: $v_{cur} := v_i$ where n_i is the maximum value in $results$
6: $last := cur$
7: jump to 2
8: $cur :=$ next parameter of cur, or k_1 for $boost_{f_{n-1}}$
9: jump to 2 if $last \neq cur$
Output: optimized parameter values

We set k_1 in 2.0 and b_f in 0.5 and randomly chose the values of $boost_f$ from [1/8, 8] initially.

6.3.4 Ranking Combination

Cai et al. [17] argued that simple combination of page-level and block-based rankings produces a new ranking significantly better than the original two. The new ranking is obtained by sorting documents in ascending order of the simple weighted summation of the original ranks of the document. The new score of document d, $\text{score}_{\text{BR+DR}}(d)$, is defined as

$$\text{score}_{\text{BR+DR}}(d) = (1 - \alpha) \cdot \text{rank}_{\text{BR}}(d) + \alpha \cdot \text{rank}_{\text{DR}}(d),$$

where $\text{rank}_{\text{BR}}(d)$ is the rank of d with the block-based method, $\text{rank}_{\text{DR}}(d)$ is the rank of d with the page-level method, and $\alpha \in [0, 1]$ is the relative weight of $\text{rank}_{\text{DR}}(d)$. In the original paper, the authors tested α in [0.0, 1.0] with 0.1 increments and obtained the best improvement of precision for the TREC 2001 data set at $\alpha = 0.5$. Because the value seems reasonable and discussion on α value is not the main topic of this dissertation, we fix α to 0.5.

6.4 Evaluation

We measured the performance of our block-level Boolean retrieval methods and heading-aware block ranking method.

6.4.1 Evaluation Method

To measure the performance of our block-level Boolean retrieval methods and heading-aware block ranking method, we used the data set and evaluation measures used in the TREC 2009–2012 web tracks. The data set consists of a document collection, queries, and the relevance assessments between them. Because the document collection is huge and we did not have enough computation resources needed for applying our heading extraction method to all the documents in it, we instead applied the baseline retrieval method to the entire collection, selected the top-100 pages for each query, and extracted a hierarchical heading structure only from them. We then eliminated irrelevant blocks with our block-level Boolean retrieval methods and ranked the pages based on the best score of the relevant blocks in the page with our heading-aware block ranking method. We explain the details of the data set and our evaluation procedure in the following.

Document Collection

The document collection in the data set is ClueWeb09, which is a huge web snapshot crawled in 2009. Because of our resource limitation, we used its subset, ClueWeb09B. Because the collection does not contain external files, such as images and style sheets, which affect our heading extraction method, we downloaded such files from the Internet Archive when they were available.

Queries

There are 200 queries (50 per year) in the data set. To avoid bias, we split them into two sets, *training* and *test* query sets. Since original query sets were biased for each year (for example, the number of multi-keyword queries is 27 for 2010 while 50 for 2011), we classified the odd-numbered 100 queries into the training query set, and the remaining even-numbered 100 queries into the test query set.

Relevance Assessment Data

As the ground truth of the document-query relevance, we used relevance assessment data provided by NIST. The data cover multiple query intents per query, which includes both informational and navigational intents. We, however, only used the first informational intent for each query. This is because well-known page-level metadata, such as page titles, are much more important than hierarchical headings in navigational intents where a user seeking a specific page or site is assumed [27].

Baseline Method

As the baseline method, we extracted pages including all the keywords, ranked them by using a variant of BM25F [83], then selected the top-100 pages for each query as the *initial ranking*. We obtained 19,934 pages in total. Note that the baseline method extracted less than 100 pages for some queries. The baseline method is much simpler than state-of-the-art methods, but as discussed later in Section 6.4.3, it achieved performance comparable to the best TREC methods after the fine tuning explained in Section 6.3.3.

When we retrieve blocks by the baseline method, we used keywords occurring in any of the following five fields: the three fields used in [86], i.e., title, anchor text of in-links, and body text, and two more fields: the URL of the page and text explicitly specified as headings by HTML tags (H1 to H6 and DT). We added them because the URL can be considered as the heading of a page, and explicitly tagged headings may also be useful for scoring a page. With our heading-aware block ranking method, we also used these fields of the page, but we did not use explicit headings because we instead used headings extracted with HEPS. We obtained external anchor texts from the ClueWeb09 web site. To avoid the effect of spam links, when a page has multiple in-links with the same anchor text, we included only one of them in the anchor text field of the page.

Block extraction

Our block extraction method extracted 637,584 blocks from the 19,891 unique pages in the initial rankings. When we calculated $\mathrm{bf}(\kappa)$ in the BM25F function, we used these 637,584 blocks. Therefore, N in the function is 637,584.

Implementation details

Because our research target is text-based retrieval, we converted each IMG node into a text node whose contents are the alternate text and the URL in the original IMG node. After that, we only regarded text nodes as contents of headings and blocks. We used Solr 4.0.0[2] as the search engine. We pre-processed documents and queries with Solr's default settings for English text. It removed 33 stop words and stemmed all terms by using the Porter stemming algorithm [84].

6.4.2 Evaluation of Block-Level Boolean Retrieval

First, we evaluated the accuracy of our block-level Boolean retrieval methods. Because block-level relevance assessment data were unavailable while page-level data were available, we measured the accuracy of page-level Boolean retrieval based on our block-level Boolean retrieval methods. The page-level retrieval is based on the following simple idea: If a page does not con-

[2] Apache Solr `http://lucene.apache.org/solr/`

Table 6.1: Comparison of number of retrieved blocks, number of retrieved pages, page-level mean precision, recall and F-score by our block-level Boolean retrieval methods.

Method	$\|Blocks\|$	$\|Pages\|$	P	R	F
All (baseline)	637,584	19,934	.149	**.2325**	.154
HeadFull	379,204	19,807	.150	.2320†	.154
HeadDirect	367,735	18,837	.157‡	.230†	**.158**‡
FullOnly	199,036	19,521	.151‡	.231‡	.154
DirectOnly	179,334	18,035	**.158**‡	.223‡	.155

†statistically significant difference from All ($p < 0.05$)

‡statistically significant difference from All ($p < 0.005$)

tain blocks retrieved by our block-level Boolean retrieval methods, page-level methods should not also retrieve the page. Based on this idea, we eliminated such pages from the initial page set then compared the resulting new page set with the initial page set.

Evaluation Measures

We used the *balanced F-score* for comparison, which is defined as $2 \cdot Precision \cdot Recall/(Precision + Recall)$ where

$$\begin{aligned} Precision &= |extracted\ relevant\ pages|/|all\ extracted\ pages| \text{ and} \\ Recall &= |extracted\ relevant\ pages|/|all\ relevant\ pages| \,. \end{aligned}$$

We used the F-score because the block-level Boolean retrieval methods do not rank the extracted pages. To integrate the scores for multiple queries, we calculated their arithmetic mean.

Evaluation Results and Discussion

The results are listed in Table 6.1. The methods are listed in descending order of the number of retrieved blocks. Even with HeadFull, 40.5% of the blocks in the initial page set was eliminated, and with DirectOnly, 71.9% was eliminated.

While the methods eliminated many blocks, as we removed only the pages that include no retrieved blocks, all the methods had little effect on page-level evaluation measures. In particular, HeadFull and FullOnly extracted almost all pages because the full contents of the top-level block correspond to the entire page body. Note that a few pages were not extracted because the full contents of the blocks do not contain metadata (URL, page title, and anchor text of in-links), while the baseline method used the term occurrences in them. Both HeadDirect and DirectOnly significantly improved precision (+0.008 and +0.009, respectively). HeadDirect, however, retained higher recall than DirectOnly. As a result, according to Student's paired t-test (where each

pair is composed of the evaluation scores of the baseline and the extended method for a query), only HeadDirect resulted in a statistically significant improvement in F-score. This supports the importance of the ancestor headings of blocks. Hereafter in this chapter, we discuss the statistical significance based on the same test procedure.

6.4.3 Evaluation of Heading-Aware Ranking

Second, we evaluated the accuracy of our heading-aware block ranking method. As explained in Section 6.4.1, we evaluated this method by comparing the results of its page-level re-ranking with the initial rankings.

Re-ranking Methods

We compared the rankings produced with the following methods. *IR* is the initial ranking method. *HR* is our heading-aware block ranking method. *EBR* is an extended version of Cai et al.'s BR (Block Rank) [17]. The EBR method extracts blocks from pages by using VIPS and ranks the blocks by using BM25F by taking into account their full contents and page-level metadata. We added three new fields, URL, title, and anchor text, to BR as explained above. We did not test the original BR separately. This is because, if the newly added fields are useless for the ranking, their weight must be optimized to zero and EBR would become the same as the original BR.

In addition, we compared all combinations of the four block-level Boolean retrieval methods explained in Section 6.2.4 with EBR and our HR. Because VIPS does not extract the headings of blocks, we did not combine our HeadFull or HeadDirect with EBR. Note that the field for body text used in HR includes the full contents of the block even if we combine HR with HeadDirect or DirectOnly.

We also tested the ranking combination explained in Section 6.3.4. We combined each block-based ranking with IR.

Evaluation Measures

As the main measure for comparing our heading-aware block ranking method and the baseline, we used nDCG@20, one of the modern standard evaluation measures of TREC [26, 27]. We calculated the nDCG@20 score of a ranking with the following formula:

$$\mathrm{nDCG@20} = \frac{\mathrm{DCG@20}}{\mathrm{ideal\ DCG@20}}$$

$$\text{where } \mathrm{DCG@20} = \sum_{i=1}^{20} \frac{2^{g_i} - 1}{\log_2(1+i)} \,,$$

where $g_i \in \{0, 1, 2, 3, 4\}$ is the relevance grade of the i-th document in the ranking.

To compare our methods with other TREC methods, we also adopted three other evaluation measures. Expected reciprocal rank (ERR) is the other modern official evaluation measure suited for evaluating rankings for navigational intents [22]. We calculated other well-known measures, Precision@k (P@k, precision of top-k ranking) and mean average precision (MAP). MAP is the arithmetic mean of the average precision for each query, and average precision is the average of Precision@k where k is the rank of each retrieved and relevant document. In the same way as MAP, we calculated their arithmetic mean to integrate the scores for multiple queries.

Optimized Parameter Values and Discussion

First, we show BM25F parameter values that maximized the nDCG@20 score of the ranking by each method for the training queries in Table 6.2. In this table, $boost_f$ (the weight of a field f) is expressed as the ratio against the weight of the full contents of the block (or the entire page for IR). The values in parentheses are b_f.

For IR, the optimal setting shown in Table 6.2(a) is the best of 3,692 settings derived from 16 randomly chosen initial settings. The trend of the obtained optimal values is similar to that in the original paper of BM25F [86]. The title is the most important, and the anchor text is also much more important than body content. This result suggests our optimization process worked well.

For HR, we separately optimized parameters for each block-level Boolean retrieval method and also for all the blocks in the initial page set (*All* in Table 6.2(b)). For each method, we tested 5,760–6,769 settings derived from 16 randomly chosen initial settings.

For the methods other than HR/HeadDirect, the weight of ancestors' headings and/or parent's heading were close to the upper limit of the weights we set. This may be because pages well structured by headings tend to contain information on diverse topics; therefore, such pages have a higher possibility to be relevant to search intents. Note that the existence of ancestors' headings in a page implies that the page contains at least three levels of headings, and similarly, parent's headings imply two levels of headings. In fact, when we extracted only pages containing at least one block whose ancestors' headings include at least one keyword (excluding stop words), the precision was 0.265. Similarly, it was 0.244 for the parent's heading. Both are significantly higher than those for the block's own heading (0.224) and the page title (0.223). In addition, the ratio of the most relevant pages (i.e., $g_i = 4$) in all extracted relevant pages was 34.3% for ancestors' headings and 26.4% for the parent's heading, while 20.8% for the own heading and 16.7% for the page title. The weights obtained for HR/HeadDirect agree with these results. This strongly supports the validity of this method.

The precision of structure extraction may also affect the weights of ancestors' and parent's headings. As discussed in Section 3.4.4, our HEPS method extracts ancestor and parent headings

Table 6.2: Values of BM25F parameters k_1, $boost_f$, and b_f (in parentheses), optimized for each method and training queries.

Ranking by	Retrieval by	Training Queries	k_1	URL	Title	Anchor text	Ancestor headings	Parent heading	Own* heading	Full contents
(a)IR	All	Odd numbered	0.21	0.84 (0.7)	22.6 (0.7)	9.51 (0.1)	-	-	1.19 (0.5)	1 (0.1)
(b)HR	All		0.84	1.19 (0.8)	4.76 (0.8)	1.19 (0.0)	108 (1.0)	108 (0.7)	0.42 (0.7)	1 (0.2)
	HeadFull		0.84	0.42 (0.0)	3.36 (1.0)	2.83 (0.0)	0.50 (0.2)	108 (1.0)	8.00 (0.1)	1 (0.4)
	HeadDirect		0.11	0.50 (0.5)	4.00 (0.6)	1.00 (0.0)	22.6 (1.0)	26.9 (0.7)	8.00 (0.5)	1 (0.8)
	FullOnly		1.00	0.84 (0.8)	6.96 (0.8)	3.37 (0.0)	90.6 (1.0)	3.35 (0.3)	0.50 (0.5)	1 (0.2)
	DirectOnly		0.30	0.30 (0.1)	1.98 (0.7)	0.99 (0.0)	0.25 (0.5)	108 (0.6)	1.42 (0.0)	1 (0.3)
(c) EBR	All		11.3	6.72 (0.7)	13.5 (0.8)	5.66 (0.0)	-	-	-	1 (0.1)
	FullOnly		1.40	3.37 (0.4)	11.3 (0.8)	16.0 (0.0)	-	-	-	1 (0.1)
	DirectOnly		32.0	1.00 (0.7)	13.4 (0.2)	19.0 (0.0)	-	-	-	1 (0.1)
(d)IR	All	TREC 09,10,12	0.60	0.14 (0.3)	16.0 (0.7)	16.0 (0.1)	-	-	4.74 (0.4)	1 (0.5)
(e)HR	HeadDirect		0.01	1.68 (0.9)	3.37 (1.0)	1.66 (0.1)	19.0 (1.0)	8.00 (0.4)	2.00 (0.0)	1 (0.7)

$\star$ For initial ranking (IR) method, weight of explicitly tagged headings is shown in this column.

Table 6.3: Comparison of mean nDCG for test queries.

Ranking by	Retrieval by	W/o ranking combination	Ranking combi. w/IR (α = 0.5)
(a)IR	All	.173	
(b)HR	All	.205 (+18.3%)†	.210 (+21.1%)‡
	HeadFull	**.206** (+18.8%)†	.217 (+25.2%)‡
	HeadDirect	.193 (+11.2%)	**.218** (+25.6%)‡
	FullOnly	.196 (+13.1%)	.193 (+11.1%)†
	DirectOnly	.197 (+13.7%)	.193 (+11.7%)†
(c) EBR	All	.175 (+1.07%)	.174 (+0.70%)
	FullOnly	.164 (-5.23%)	.172 (-0.57%)
	DirectOnly	.163 (-5.84%)	.175 (+1.07%)

†statistically significant difference from IR ($p < 0.05$)

‡statistically significant difference from IR ($p < 0.005$)

in far better precision (0.810 and 0.808, respectively) than the average (0.638).

In summary, these results suggest that information on headings are important and can improve the rankings of related blocks.

Comparison of Proposed Methods

Next, we show the nDCG@20 scores of the baseline and our HR applied to the test query set. Note that the parameters were optimized for the training query set to avoid bias. The baseline was 0.173 (Table 6.3(a)). All our HR improved the score (Table 6.3(b)) regardless of the choice of Boolean retrieval method and the use of ranking combination method. Without combination with IR, HR/HeadFull resulted in the best score, and HR with all blocks followed with a slight difference. Improvement with these two methods was statistically significant with a significance level of 5%. These results show that our HR is quite useful for improving ranking.

When combined with IR, HR/HeadDirect resulted in the best improvement. Its score was the best of all the methods including those without being combined with IR. For the remainder of this chapter, we call this method *our best method*. This fact means that HR/HeadDirect produced a ranking more complementary to page-level ranking with IR than the other methods. The HR/HeadFull method followed the best method by a slight difference. On the other hand, HR/All with IR resulted in only a slight improvement compared with HR/All without IR. These three scores were statistically significant with a better significance level, 0.5%. Two retrieval methods without considering the headings of ancestor blocks resulted in worse scores when combined

Table 6.4: Comparison with top-3 (by ERR) cat. B runs for TREC '11 web track adhoc task [26].

Run	ERR	nDCG	P	MAP
srchvrs11b	**.131**	**.233**	.298	**.110**
DFalah11	.122	.204	.275	.079
UAmsM705tiLS	.119	.202	.273	.085
(d) Our baseline	.129	.216	.284	.105
(e) Our best method	.130	.225	**.303**	.109

with IR. These results show that our HR/HeadFull and HR/HeadDirect are especially useful for complementing the page-level ranking.

Comparison with VIPS and Block Rank

Next, we compared our HR with EBR, which uses VIPS for block extraction. In the same way as we did for our HR, we optimized parameters for each block-based Boolean retrieval method starting from 16 randomly chosen initial settings, and checked 3,124–3,272 settings for each method. The numbers of checked settings were smaller than those for HR, but it is mainly because there are a smaller number of parameters for the EBR method.

The nDCG@20 scores for EBR are shown in Table 6.3(c). Regardless of the choice of the Boolean retrieval method and the combination with IR, their scores exhibited only slight improvement or even worsened. These results suggest that page segmentation with VIPS may not be effective for today's web pages. This weakness was also reported in [35] regarding today's data-intensive web pages including access to many types of content and rich sets of blocks.

Comparative Evaluation with TREC Runs

Finally, we compared our rankings with past TREC runs. We focused on the TREC 2011 web track adhoc task because all the query intents for the task are informational. Note that we could not compare our rankings with the runs of other years directly because the data sets of the adhoc tasks in the other years contain both informational and navigational intents, and we evaluated methods focusing on only informational intents (see Section 6.4.1).

Because our pre-defined training data set contains 25 queries from the data set, we re-optimized the BM25F parameter values for the others, namely the TREC 2009, 2010, and 2012 data sets to avoid bias. The optimization results are listed in the bottom of Table 6.2. The rows (d) and (e) are for the initial ranking method and for our best method, respectively. The evaluation results are listed in Table 6.4. The three results on top, which are cited from the overview paper of the track [26], are those that achieved the top-3 ERR@20 scores among category B runs, which used ClueWeb09B document collection, which we also used for our evaluation. There were also

some category A runs that achieved better ERR@20 scores, but they used the entire ClueWeb09 collection. Scores of other measures, nDCG@20, Precision@20, and MAP are also shown.

The results show that both our baseline and best methods achieved scores comparable to the other three runs for all four evaluation measures. Even our baseline method achieved the second best score among the top-3 TREC runs for all measures. This means that the superiority of our best method over the baseline is not because of the weakness of the baseline. Our best method achieved the best Precision@20 score. For ERR@20 and MAP, our best method achieved the second best result, and the differences from the top (*srchvrs11b*) are small (-0.8% and -0.9%, respectively). As these results suggest, our best method is comparable to the current best methods.

6.5 Summary

We proposed four block-level Boolean retrieval methods and a heading-aware block ranking method. According to our evaluation results, our HR/HeadDirect method significantly improved the resulting rankings and achieved evaluation scores comparable with the current ranking methods. In particular, our evaluation results indicated that keyword occurrences in the ancestor headings strongly support the relevance of the block to the query intent, and that pages containing a hierarchical block structure have more possibility to be relevant to the query intent. To the best of our knowledge, no existing web search system uses a hierarchical heading structure. Therefore, our heading-aware block ranking method has the potential to complement the current best web page search systems. Specifically, application of our methods to state-of-the-art learning to rank [63] systems is for future work.

CHAPTER 7

HEADING-AWARE SNIPPET GENERATION FOR WEB SEARCH

7.1 Introduction

Most web pages contain hierarchical heading structure. The structure is composed of nested logical blocks and each block is associated with a heading which briefly describes the topic of the block. Because of this feature of headings, to fully understand sentences in web pages, readers should first read the contextual headings of the sentences. The contextual headings (or merely headings) of a sentence are the headings associated with either the block containing the sentence or its hierarchical ancestor blocks.

Therefore, the contextual heading words (or merely heading words), i.e. the words in the contextual headings, are important for understanding their associated sentences.

For this reason, there have been several studies on heading-aware snippet generation [81,101]. These methods assign higher scores to headings themselves [101] or sentences containing their heading words [81].

However, contextual heading words are very often omitted from their associated sentences because human readers can recognize the topic of the sentences by reading their headings first. For example, in the example page in Figure 7.1 containing the hierarchical heading structure in Figure 7.2, by the sentence “It has risks as well as big benefits”, the author is writing about swimming without the word swimming. For a query “swimming risks”, the existing methods cannot assign higher relevance scores for such sentences.

To solve the problem, we develop a new method of heading-aware snippet generation that

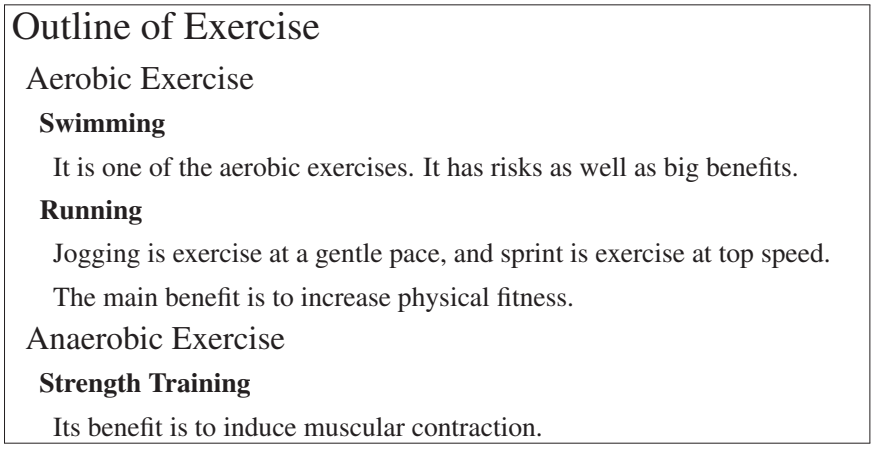


Figure 7.1: Example web page with hierarchical heading structure.

takes the omission of heading words into account. Our method assign higher scores to sentences that either include query keywords within themselves or have contextual headings including the query keywords. Our new approach does not conflict with the existing approach that uses heading-word occurrences in sentences. Therefore, we also consider another method which combines the two types of evidences, namely heading-word occurrences in sentences themselves and query keyword occurrences in contextual headings of sentences.

7.2 Snippet Generation Methods

In this section, we explain four snippet generation methods for web search.

7.2.1 Basic Snippet Generation Method

Generally, the quality of document summaries relies on three factors [2]. The *readability* of a summary is how easy it is for humans to read [25, 52], its *representativeness* is how well it represents the contents of its original document [61], and its *judgeability* is to what extent it helps humans to judge the relevance of its original document to the humans' informational needs [61]. Among the three factors, judgeability is the most important for search result snippets.

Basically, search result snippets are generated from web pages by search systems in three steps as described below [81, 101, 109]. First, the system splits the page into text fragments. To generate readable snippets, many systems split it into semantically coherent fragments such as sentences. Second, the system scores the fragments based on numbers of occurrences of important words in the fragments. The occurrences of the query keywords directly indicate the relevance of the original page to the humans' intent behind the query. Therefore, almost all sys-

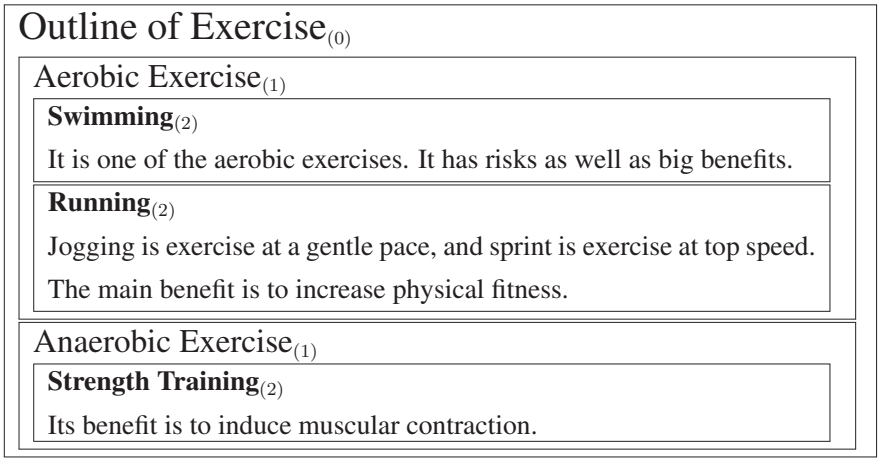


Figure 7.2: Hierarchical heading structure in Figure 7.1. Each rectangle encloses block, each text with subscript is heading and each subscript number represents depth of block in the hierarchy.

tems take keyword occurrences into account for judgeability. On the other hand, other important words (see Section 2.5) in a document represent the contents of the original document better than other words. Therefore, many systems take important-word occurrences into account for higher representativeness. Third, the system selects the top-ranked sentences into the summary. In this step, the system selects the sentences in descending order of their scores until the length of the summary reaches the limit. Our *baseline* method also consists of these three steps. The method is described as below.

Input A web page with its DOM tree structure. Note that we consider only documents in English throughout this chapter. This is merely because we use some language-dependent libraries for sentence segmentation and stemming. Note that our heading-aware sentence scoring methods are language-independent.

Sentence Segmentation First, the method extracts the text contents of the page, then segments the contents into sentences. As the text contents of a page, we extract the text contents of all text and IMG (image) nodes under the BODY (content body) node of the page, and concatenate them in the document order. As the text contents of IMG nodes, we extract their alternate text. From IMG nodes without alternate text, we extract the URLs of the images. We split the text of the page into sentences by the Treat implementation[1] of the Stanford sentence segmentation.

Sentence Scoring We score the sentences based on number of keyword occurrences in them by a variant of the BM25 function [87]. The function calculates the score of a sentence s for

[1] `https://github.com/louismullie/treat`

Outline of Exercise

... It has risks as well as big benefits. ... Running ... The main benefit is to increase physical fitness. ... Its benefit is to induce muscular contraction.

Figure 7.3: Example baseline snippet for example query "benefits running."

keyword query Q by the following formula:

$$\text{score}(Q, s) = \sum_{\kappa \in Q} \frac{\text{weight}(\kappa, s)}{k_1 + \text{weight}(\kappa, s)} \log \frac{N - \text{sf}(\kappa) + 0.5}{\text{sf}(\kappa) + 0.5}$$

where κ is a keyword in Q, k_1 is a parameter to modify the scaling of occurrence frequency, N is the number of all sentences, and $\text{sf}(\kappa)$ is the number of sentences containing κ in the page. The $\text{weight}(\kappa, s)$ is defined as

$$\text{weight}(\kappa, s) = \frac{\text{occurs}(\kappa, s)}{\left((1 - b) + b \cdot \frac{\text{length}(s)}{\text{avgLength}}\right)}$$

where $\text{occurs}(\kappa, s)$ is the number of occurrences of κ in s, b is the parameter to modify the strength of length normalization, $\text{length}(s)$ is the length of s in number of words, and avgLength is the average length of sentences in the page. We count $\text{occurs}(\kappa, s)$ after the basic pre-processing, i.e. stemming by the Porter stemming algorithm [84] and removal of 33 default stop words of Lucene library.

Sentence Selection To select sentences, we simply scan sentences in descending order of their scores, and if there still remains the space to include the sentence into the snippets, we include it. We can also adopt advanced methods for the selection, such as Maximal Marginal Relevance [19, 118], however, we adopt this simple method because this step is not the main topic of this dissertation.

Output The generated snippet and the title of the input page. If there is no page title specified, we output the page URL. Figure 7.3 is an example output.

7.2.2 Occurrences of Heading Words in Sentences

Heading words are important to represent their associated blocks because the words are selected by the authors to describe the block topics of the blocks briefly. As discussed in Section 7.1, we consider contextual headings and heading words of sentences. For example, in Figure 7.2, the heading words of the sentence "It is one of the aerobic exercises" are outline, of, exercise, aerobic, and swimming.

Outline of Exercise

> **Aerobic Exercise** > **Swimming**

It is one of the aerobic exercises. ...

> **Aerobic Exercise** > **Running**

Jogging is exercise at a gentle pace, and sprint is exercise at top speed. The main benefit is to increase physical fitness.

Figure 7.4: Example heading-aware snippet for example query "benefits running."

One promising way to generate representative snippets is to extract sentences containing many occurrences of their contextual heading words. Pembe and Güngör [81] proposed such a method. We also use this idea for our *existing* method, which is based on summation of the BM25 scores for two types of words, namely query keywords and heading words. However, weighted summation of BM25 scores produces worse rankings in case that they count occurrences of the same words [86]. Therefore, we split the words into three types, namely narrow query keywords (NK-words), narrow heading-words (NH-words), and heading keywords (HK-words). The NK-words are query keywords which are not heading words, and the NH-words are heading words which are not query keywords. The HK-words are the words which are heading words and also query keywords. We modify the baseline method as described below.

Sentence Segmentation Because headings and blocks are semantically coherent fragments and no sentence should overlap the boundaries of them, we segment the text contents of pages into text fragments by all their boundaries, and then segment the fragments into sentences by the Treat implementation explained before. Because we show headings in a different way from other components of snippets (as discussed later), we separately extract headings and other sentences.

Sentence Scoring The new $\mathrm{score}(Q, s)$ and $\mathrm{weight}(w, s)$ are calculated by:

$$\begin{aligned}
\mathrm{score}(Q, s) &= \sum_{w \in Q \cup H(s)} \frac{\mathrm{weight}(w, s)}{k_1 + \mathrm{weight}(w, s)} \log \frac{N - \mathrm{sf}(w) + 0.5}{\mathrm{sf}(w) + 0.5}, \\
\mathrm{weight}(w, s) &= \frac{\mathrm{occurs}(w, s) \cdot boost^{\mathrm{typeof}(w)}}{\left((1 - b) + b \cdot \frac{\mathrm{length}(s)}{\mathrm{avgLength}}\right)}
\end{aligned}$$

where $H(s)$ is heading words of s, w is a word in Q or $H(s)$, and $\mathrm{typeof}(w) \in \{\text{NH-words}, \text{NK-words}, \text{HK-words}\}$ is the type of w. The parameter $boost^{\mathrm{typeof}(w)}$ represents importance of the occurrences of w whose type is $\mathrm{typeof}(w)$.

Output The generated text snippets and their headings including the title or URL of the input page. In case that heading structure of documents are given, we can improve readability of snip-

pets by showing sentences and their headings separately [81]. We also adopt this idea. Figure 7.4 shows an example output.

The other steps of this method are same as those of the baseline method. We call this method the *existing* method.

7.2.3 Keyword Occurrences in Headings

Our observation is that the heading words are very often omitted from sentences. Despite such omission, heading words are important to clarify the topic of their associated sentences. Therefore, to select sentences that well represent its original document considering such omission, we must count keyword occurrences in the contextual headings of the sentences as well as in the sentences themselves.

Sentence Scoring Based on this idea, we regard that each sentence comprises two fields, namely the contents of the sentence itself and its contextual headings, and adopt a variant of BM25F, a scoring function for documents comprising multiple fields [86]. The function calculates the score of a sentence S comprising two fields for keyword query Q by the formulas:

$$\begin{aligned}
\mathrm{score}(Q, S) &= \sum_{\kappa \in Q} \frac{\mathrm{weight}(\kappa, S)}{k_1 + \mathrm{weight}(\kappa, S)} \log \frac{N - \mathrm{sf}(\kappa) + 0.5}{\mathrm{sf}(\kappa) + 0.5}, \\
\mathrm{weight}(\kappa, S) &= \sum_{f \in S} \frac{\mathrm{occurs}(\kappa, f, S) \cdot boost_f}{\left((1-b) + b \cdot \frac{\mathrm{length}(f,S)}{\mathrm{avgLength}(f)}\right)}
\end{aligned}$$

where f is a field in S, $\mathrm{occurs}(\kappa, f, S)$ is number of occurrences of κ in f of S, $boost_f$ is the weight of query keyword occurrences in f, $\mathrm{length}(f, S)$ is the length of f of S, and $\mathrm{avgLength}(f)$ is the average length of f. The other steps of this method are same as those of the existing method. We call this method *our* method.

7.2.4 Combination of Two Advanced Methods

Above two modifications can be applied independently. Therefore, we can consider the fourth method which adopts both of them.

Sentence Scoring We calculate the combined score and weight by:

$$\begin{aligned}
\mathrm{score}(Q, S) &= \sum_{w \in Q \cup H(S)} \frac{\mathrm{weight}(w, S)}{k_1 + \mathrm{weight}(w, S)} \log \frac{N - \mathrm{sf}(w) + 0.5}{\mathrm{sf}(w) + 0.5}, \\
\mathrm{weight}(w, S) &= \sum_{f \in S} \frac{\mathrm{occurs}(w, f, S) \cdot boost_f^{typeof(w)}}{\left((1-b) + b \cdot \frac{\mathrm{length}(f,S)}{\mathrm{avgLength}(f)}\right)}
\end{aligned}$$

where $boost_f^{typeof(w)}$ is weight of occurrences of w in f. The other steps are same as those of our method. We call this method the *combination* method.

Table 7.1: Boost for occurrence of words of each type in each field.

Parameter name	Value	Parameter name	Value
$boost_{\text{headings}}^{\text{HK-words}}$	3.0	$boost_{\text{sentence}}^{\text{HK-words}}$ ($boost^{\text{HK-words}}$)	4.0
$boost_{\text{headings}}^{\text{NH-words}}$	0	$boost_{\text{sentence}}^{\text{NH-words}}$ ($boost^{\text{NH-words}}$)	1.0
$boost_{\text{headings}}^{\text{NK-words}}$ ($boost_{\text{headings}}$)	3.0	$boost_{\text{sentence}}^{\text{NK-words}}$ ($boost^{\text{NK-words}}$, $boost_{\text{sentence}}$)	3.0

7.2.5 Parameters and Fine Tuning

These scoring functions require three types of parameters: The saturation factor k_1 controls scaling of weighted term frequency, b controls the strength of length normalization, and $boost$ controls the weights of term occurrences of each type of words in each field. Because the scaling and normalization are not the main topic, we use the default values 2.0 for k_1 and 0.75 for b [86].

The setting of $boost$ is important for effective heading-aware snippet generation. According to the observation by Pembe and Güngör [81], occurrences of query keywords are three times more important than those of heading words. Therefore, we use 3.0 for all $boost_{\text{sentence}}^{\text{NK-words}}$ in Section 7.2.4, $boost_{\text{sentence}}$ in Section 7.2.3, and $boost^{\text{NK-words}}$ in Section 7.2.2 while we use 1.0 for all $boost_{\text{sentence}}^{\text{NH-words}}$ in Section 7.2.4 and $boost^{\text{NH}-\text{words}}$ in Section 7.2.2. Because there is no existing observation about the balance of weights of the keyword occurrences in sentences and in their contextual headings, we simply use 3.0 (same as $boost_{\text{sentence}}^{\text{NK-words}}$) for $boost_{\text{headings}}^{\text{NK-words}}$ in Section 7.2.4 and $boost_{\text{headings}}$ in Section 7.2.3. Because heading words always occur in headings, we use 0 for $boost_{\text{headings}}^{\text{NH-words}}$. As the weight of HK-words, we use the summations of the weights of NH-words and NK-words. All the $boost$ values are listed in Table 7.1 for reference.

7.3 Evaluation

In this section, we evaluate each snippet generation method.

7.3.1 Evaluation Method

As discussed in Section 7.2.1, judgeability is the most important property of effective search result snippets. Therefore, to measure the effectiveness of snippet generation methods, we measure the judgeability of their output snippets. To measure the judgeability, in the INEX snippet retrieval track [102], the results of relevance judgments under two different conditions are compared. One judgment is performed based on the entire documents while the other is only based on their snippets. If they agree, the snippets provided high judgeability and the snippet generation method was effective. We use this measure and also their length limit of snippets, which is 180 letters for a page.

However, the target of INEX is XML documents while our target is web pages. Therefore, we used a data set for the Text Retrieval Conference (TREC) 2014 web track ad-hoc task [29].

7.3.2 Data Set

Queries and intents Fifty keyword queries and their intent descriptions.

Document collection ClueWeb12B, a web snapshot crawled in 2012. We extracted top-20 pages for each query (total 1,000 pages) from the official baseline search result for the TREC task. The result is generated by the default scoring by Indri search engine (including query expansion based on pseudo-relevance feedback) and filtered by Waterloo spam filter.

Page-based relevance judgment data The TREC official graded relevance of the entire pages to the intents. We simply regarded that documents whose grades are more than 0 as relevant to the intent, and the others are irrelevant.

Snippet-based relevance judgment data We carried out a user experiment with four participants. They are all non-native English readers familiar with web search. In each period of the experiment, each participant is required to read the intent description behind a query first. Next, he is required to scan top-20 search result items including the snippets generated by a method and to judge whether each original page is relevant to the intent. We broke out the search results to participants by Graeco-Latin square, therefore each snippet was not judged more than once, and each participant did not judge a page more than once and used all methods almost evenly. As described above, we adopted binary relevance. It is because the user of a real web search engine must decide to read or not for each original page based on its snippets and there is no intermediate choice.

7.3.3 Evaluation Measures

Evaluation measures We use three evaluation measures from the INEX track: Recall, negative recall (NR), and the geometric mean (GM) of them. Recall is the ratio of pages correctly judged as relevant on their snippets to pages relevant as a whole. It is calculated by $|\textit{Correctly judged pages relevant as a whole}|/|\textit{Pages relevant as a whole}|$. On the other hand, NR is the ratio of pages correctly judged as irrelevant on their snippets to pages irrelevant as a whole. It is calculated by $|\textit{Correctly judged pages irrelevant as a whole}|/|\textit{Pages irrelevant as a whole}|$. GM is the primary evaluation measure of the INEX track and our evaluation. It is calculated by $\sqrt{Recall \cdot NR}$. To integrate the evaluation scores for multiple queries, we calculated the arithmetic mean of them.

Table 7.2: Comparison of average evaluation scores of four methods.

Method	Recall	NR	GM
Baseline	**.475**	**.828**	**.512**
Exist.	.373	.780	.386
Ours	.438	.777	.456
Combi.	.396	.776	.401

Table 7.3: Average evaluation scores of four methods for each type of queries.

(a) For 24 *faceted* queries.

Method	Recall	NR	GM
Baseline	**.524**	**.806**	**.539**
Exist.	.416	.737	.378
Ours	.509	.723	.470
Combi.	.290	.737	.257

(b) For 24 *single* queries.

Method	Recall	NR	GM
Baseline	.431	**.837**	.488
Exist.	.336	.804	.392
Ours	.375	.816	.443
Combi.	**.491**	.795	**.530**

7.3.4 Evaluation Results and Discussions

Comparison of snippet generation methods First, we compared the average evaluation scores of four snippet generation methods. Table 7.2 lists the results. The baseline method achieved the top scores by all the evaluation measures. Our heading-aware method achieved the second GM score. The existing heading-aware method achieved the worst GM score and its difference from the baseline method was statistically significant ($p < 0.05$) according to Student's paired t-test where each pair is composed of the evaluation scores of the baseline and heading-aware methods for a query. Hereafter in this chapter, we discuss statistical significance based on the same test procedure. There was no statistically significant difference from the baseline to the other methods. As shown in this result, the heading-aware methods were not effective for general queries. The difference of the GM scores was mainly caused by the difference of the recall scores. In fact, the best method improved the recall score by 27.3% from the worst while the NR score by only 6.70%. In other words, the effectiveness of the methods mainly depends on how many relevant pages its output snippets can indicate to the users. This tendency was seen through all evaluations.

Effect of query type For detailed evaluation, TREC splits queries into several types. *Faceted* queries are underspecified, while there are clear and focused intents behind *single* queries [29]. The data set contains 24 queries of each type. It also contains only two *ambiguous* queries, however we ignored them. The scores for the faceted queries are listed in Table 7.3 (a) and the

Table 7.4: Average GM scores of four methods and query length excluding stopwords.

$\|Keywords\|$	$\|Queries\|$	Baseline	Exist.	Ours	Combi.
2	25	**.585**	.406	.543	.393
3	10	**.503**	.387	.378	.388
4 or more	15	.394	.350	.362	**.425**

scores for the single queries are listed in (b). As shown in these tables, the baseline method achieved the best scores for faceted queries while the combination method achieved the best recall and GM scores for single queries. Only the GM score difference between the baseline and combination methods for faceted queries was statistically significant. This fact suggests that heading-aware snippet generation methods may be effective for clearly specified intents. To indicate the relevance of a page to a clearly specified intent, small number of sentences and their rich contextual information, i.e. their headings, may be important. In the other cases, it may be important to show a larger number of sentences in the page. For further discussion, another evaluation with more queries is needed.

Effect of query length When a user inputs multiple keywords, the user is probably requesting pages in which all the keywords occur in relation to each other. On the other hand, as discussed in Section 7.2.3, contextual heading words have semantic relationship to their associated sentences. Therefore, the heading-aware methods must be more useful for queries containing more keywords. In other words, there are usually less sentences containing more different keywords directly. However, considering the contextual headings of the sentences, heading-aware methods can detect more of relevant sentences. Based on this idea, we classified the queries by their numbers of keywords excluding stopwords. Table 7.4 lists the numbers of queries in each class and the GM scores of each method for each class. For the queries with two keywords, the baseline method achieved the best GM score and its differences from the existing and combination methods were statistically significant. However, only the combination method retained its score for the longer queries while the other three methods lost their scores. The correlation coefficient of the GM score and the number of pairs of different query keywords for each query was 0.247 for the combination method while -0.105 for the baseline. Especially, for four or more keywords, the combination method achieved the best score. It supports the above discussion about longer queries. For further discussion, another evaluation with more queries is needed.

Query type and query length Query type depends on query length because more query keywords specify the intents of the query more clearly. In fact, the average length of single queries was 3.54 words while that of faceted queries was 2.25 words. This dependence might affect our

Table 7.5: Median amount of required time in second to check snippets of 20 pages for one query.

(a) By each method.

Method	Time in sec.
Baseline	411.5
Exist.	**308.5**
Ours	315.5
Combi.	349.0

(b) By each participant.

Participant	Time in sec.
A	297.5
B	429.0
C	347.5
D	293.0

Table 7.6: Comparison of average evaluation scores of four participants.

Participant	Recall	NR	GM
A	.367	.787	.376
B	.482	.783	.498
C	.459	.815	.451
D	.375	.776	.430

evaluation results.

Required time analysis We also measured the median required time for checking 20 pages for a query. Table 7.5 (a) lists the results. Intuitively, the assessors took much more time for our evaluation tasks than practical search tasks. It may be because they are non-native English reader, and/or because they read snippets more carefully for more accurate judgment than usual. Generally, heading-aware snippets significantly reduced the required time. It must be because the users can read structured text more easily than plain text.

Effect of assessors We also compared the required time and evaluation scores for each assessor. Table 7.5 (b) lists the median time in second required for checking 20 pages and Table 7.6 lists the average evaluation scores for each assessor. As shown in Table 7.5 (b), the required times are quite different for each assessor. The difference also affects the average GM scores of them, that is, the most and second-most careful assessors, B and C, achieved the best and second-best GM scores respectively. Note that the effect of the assessors for the other comparative evaluations is limited because each assessor uses each methods almost evenly.

7.4 Summary

We introduced a novel idea for heading-aware snippet generation and compared one baseline and three heading-aware snippet generation methods. The idea is that sentences whose contextual headings contain query keywords provide judgeability as well as sentences containing query

keywords directly. Our evaluation result indicated that the heading-aware methods were not effective for general queries. Only for queries representing its intents clearly or containing four or more keywords, the heading-aware combination method achieved the best score. This fact suggests that heading-aware snippet generation is useful for such queries. However, to discuss the statistical significance of the result, an additional evaluation with more queries is needed.

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

8.1 Conclusions

In this dissertation, we first proposed an extraction method for hierarchical heading structure in web pages, then discussed the four methods that use the information on the structure. Conclusions of each topic are as listed below:

Extracting Logical Hierarchical Structure of HTML Documents Based on Headings

We developed a method that extracts hierarchical heading structure in web pages. Our method first extracts hierarchical headings based on several assumptions on their design, and segment a document into hierarchical blocks by using these headings. This approach is expected to work as long as the document is appropriately designed so that human readers can recognize its hierarchical structure based on its visually prominent headings. We evaluated our method with a standard Web page corpus, and our experiment shows that our method outperforms the existing page segmentation method. The existing methods, however, focus on top-level page layout structure. The existing methods and our method are, therefore, complementary to each other. Our analysis on the experimental results suggests that we can improve our method by introducing more visual features and some clustering techniques for correctly classifying nodes.

Subtopic Ranking Based on Hierarchical Headings

We proposed subtopic ranking methods based on hierarchical heading structure. Our ideas are that hierarchical headings in a document reflect the topic structure of the document and that the length of contents referring to a topic reflects the importance of the topic. Based on these ideas, we proposed methods based on matching between the subtopic candidate strings and the

hierarchical headings. We evaluated our methods by using the publicly available NTCIR data set. Our evaluation results indicated that (1) our methods significantly improved the baseline rankings by commercial search engines, that (2) log-scale scoring seems effective and robust, that (3) there is no substantial difference among score integration methods, and that (4) our ranking diversification method was not effective.

Heading-Aware Proximity Measure and Its Application to Web Search

First we explained our observations about the effect of heading structure to logical proximity. Based on the observations, we proposed a heading-aware proximity measure and our heading-aware variants of three existing proximity-aware document scoring functions. We then optimized the parameters of the methods, and finally evaluated all the functions by using TREC data sets. Most of the optimized parameter values of the functions followed our assumptions. Moreover, our heading-aware Span method with our heading-aware semi-distance generated rankings significantly better than the existing Span method with simple distance.

Heading-Aware Block-Based Web Search

We proposed four block-level Boolean retrieval methods and a heading-aware block ranking method. According to our evaluation results, our HR/HeadDirect method significantly improved the resulting rankings and achieved evaluation scores comparable with the current state-of-the-art ranking methods. In particular, our evaluation results indicated that keyword occurrences in the ancestor headings strongly support the relevance of the block to the query intent, and that pages containing a hierarchical block structure have more probability to be relevant to the query intent. To the best of our knowledge, no existing web search system uses a hierarchical heading structure. Therefore, our heading-aware block ranking method has the potential to complement the current best web page search systems. Specifically, application of our methods to state-of-the-art learning to rank systems is for future work.

Heading-Aware Snippet Generation for Web Search

We introduced an idea for heading-aware snippet generation and compared one baseline and three heading-aware snippet generation methods. Our idea is that sentences whose contextual headings contain query keywords provide judgeability as well as sentences containing query keywords directly. Our evaluation result indicated that the heading-aware methods were not effective for general queries. Only for queries representing its intents clearly or containing four or more keywords, the heading-aware combination method achieved the best score. This fact suggests that heading-aware snippet generation is useful for such queries. However, to discuss the statistical significance of the result, an additional evaluation with more queries is needed.

8.2 Future directions

Last of all, we suggest two interesting future directions of our research.

8.2.1 Incompleteness of Our Assumptions and Observations

Of course, all assumptions and observations in this dissertation do not cover all web pages. For example, Kyoto Aquarium is sometimes called Umekoji Aquarium, because it is in Umekoji park. It is a paraphrasing problem and many methods for solving this type of problems, e.g., construction of thesauruses and word embedding, have been proposed. As well as paraphrasing, there are many types of problems for building practical search engines. There also exists many methods for solving them, however. It is still unclear in this dissertation whether the existing methods are applicable to our heading-aware search system.

On the other hand, there may be some types of problems which are specific to heading-aware search systems. For example, suppose a block which hierarchical headings "Kyoto Aquarium", "Nearby aquariums", and "Kaiyukan Aquarium." The block includes the term "Kyoto Aquarium" in its hierarchical headings. However, it must be about Kaiyukan Aquarium, which is not Kyoto Aquarium. Like this, our assumptions and observations do not apply to some pages.

In this dissertation, we indicated that statistical use of hierarchical heading structure improves precision of web search. However, we focused on neither rare information nor minority opinion. To search them effectively, we have to solve the problems discussed above.

8.2.2 Data Structure and Algorithms for Hierarchical Heading Structure

In this dissertation, we did not discuss amount of resources, namely memory size and computation time, required for our methods. Of course, our methods are more complicated and require more resource than the baseline methods, and therefore, we have to lower the amount of resources so as to put a heading-aware search system into practical use.

The first problem is to estimate the increase of necessary resources. For example, indexing of all hierarchical blocks requires more memory size than indexing of documents. For another example, scoring of all blocks requires more computation time than scoring of documents. Complexity analysis of our methods is needed.

The second problem is to propose data structure and algorithms for lowering the increase of necessary resources. For example, Yoshii et al. proposed a method which lowers increase of computation time for scoring all elements in XML documents by clustering the elements [114]. We should check whether such existing methods are applicable to our system, and if not, we need to propose alternative practical methods.

BIBLIOGRAPHY

[1] M. D. Adelfio and H. Samet. Schema extraction for tabular data on the web. *VLDB*, 6(6):421–432, 2013.

[2] M. Ageev, D. Lagun, and E. Agichtein. Towards task-based snippet evaluation: Preliminary results and challenges. In *MUBE, SIGIR*, pages 1–2, 2013.

[3] S. Amer-Yahia and M. Lalmas. XML search: Languages, INEX and scoring. *SIGMOD Rec.*, 35(4):16–23, 2006.

[4] A. Anjewierden. Aidas: Incremental logical structure discovery in pdf documents. In *ICDAR*, pages 374–378, 2001.

[5] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD*, pages 337–348, 2003.

[6] P. Arvola, M. Junkkari, and J. Kekäläinen. Generalized contextualization method for XML information retrieval. In *CIKM*, pages 20–27, 2005.

[7] P. Arvola, J. Kekäläinen, and M. Junkkari. Contextualization models for XML retrieval. *Inf. Process. Manage.*, 47(5):762–776, 2011.

[8] A. Bah, B. Carterette, and P. Chandar. Udel @ NTCIR-11 IMine track. In *NTCIR*, 2014.

[9] M. Beigbeder, M. Géry, and C. Largeron. Using proximity and tag weights for focused retrieval in structured documents. *Knowl. Inf. Syst.*, 44(1):51–76, 2015.

[10] M. Bendersky and O. Kurland. Utilizing passage-based language models for document retrieval. In *ECIR*, pages 162–174, 2008.

[11] A. Bouchoucha, J. Nie, and X. Liu. Université de montréal at the NTCIR-11 IMine task. In *NTCIR*, 2014.

[12] A. Broschart and R. Schenkel. Proximity-aware scoring for xml retrieval. In *SIGIR*, pages 845–846, 2008.

[13] J. T. Brudvik, J. P. Bigham, A. C. Cavender, and R. E. Ladner. Hunting for headings: Sighted labeling vs. automatic classification of headings. In *ASSETS*, pages 201–208, 2008.

[14] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR*, pages 621–622, 2006.

[15] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Accordion summarization for end-game browsing on PDAs and cellular phones. In *CHI*, pages 213–220, 2001.

[16] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. VIPS: a vision-based page segmentation algorithm. Technical Report MSR–TR–2003–79, Microsoft, 2003.

[17] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Block-based web search. In *SIGIR*, pages 456–463, 2004.

[18] J. P. Callan. Passage-level evidence in document retrieval. In *SIGIR*, pages 302–310, 1994.

[19] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[20] D. Chakrabarti, R. Kumar, and K. Punera. A graph-theoretic approach to webpage segmentation. In *WWW*, pages 377–386, 2008.

[21] H. Chao and J. Fan. Layout and content extraction for PDF documents. In *DAS*, pages 213–224, 2004.

[22] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*, pages 621–630, 2009.

[23] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666, 2008.

[24] C. L. Clarke, M. Kolla, and O. Vechtomova. An effectiveness measure for ambiguous and underspecified queries. In *ICTIR*, pages 188–199, 2009.

[25] C. L. A. Clarke, E. Agichtein, S. Dumais, and R. W. White. The influence of caption features on clickthrough patterns in web search. In *SIGIR*, pages 135–142, 2007.

[26] C. L. A. Clarke, N. Craswell, I. Soboroff, and E. M. Voorhees. Overview of the TREC 2011 web track. In *TREC*, 2011.

[27] C. L. A. Clarke, N. Craswell, and E. M. Voorhees. Overview of the TREC 2012 web track. In *TREC*, 2012.

[28] K. Collins-Thompson, P. N. Bennett, F. Diaz, C. Clarke, and E. M. Voorhees. TREC 2013 web track overview. In *TREC*, 2013.

[29] K. Collins-Thompson, C. Macdonald, P. N. Bennett, F. Diaz, and E. M. Voorhees. TREC 2014 web track overview. In *TREC*, 2014.

[30] E. Cortez, D. Oliveira, A. S. da Silva, E. S. de Moura, and A. H. Laender. Joint unsupervised structure discovery and information extraction. In *SIGMOD*, pages 541–552, 2011.

[31] R. Cummins and C. O'Riordan. Learning in a pairwise term-term proximity framework for information retrieval. In *SIGIR*, pages 251–258, 2009.

[32] M. Cutler, Y. Shih, and W. Meng. Using the structure of html documents to improve retrieval. In *USITS*, 1997.

[33] N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *SIGIR*, pages 95–104, 2011.

[34] S. Das, P. Mitra, and C. L. Giles. Phrase pair classification for identifying subtopics. In *ECIR*, pages 489–493, 2012.

[35] E. S. de Moura, D. Fernandes, B. Ribeiro-Neto, A. S. da Silva, and M. A. Gonçalves. Using structural information to improve search in web collections. *JASIST*, 61(12):2503–2513, 2010.

[36] S. Debnath, P. Mitra, N. Pal, and C. L. Giles. Automatic identification of informative sections of web pages. *IEEE Trans. on Knowl. and Data Eng.*, 17(9):1233–1246, 2005.

[37] L. Denoyer, P. Gallinari, and H. Zaragoza. Hmm-based passage models for document classification and ranking. In *ECIR*, pages 126–135, 2001.

[38] G. Dias, G. Cleuziou, and D. Machado. Informative polythetic hierarchical ephemeral clustering. In *WI*, pages 104–111, 2011.

[39] Z. Dou, S. Hu, Y. Luo, R. Song, and J.-R. Wen. Finding dimensions for queries. In *CIKM*, pages 1311–1320, 2011.

[40] M. A. El-Shayeb, S. R. El-Beltagy, and A. A. Rafea. Extracting the latent hierarchical structure of web documents. In *SITIS*, pages 385–393. 2006.

[41] J. Fleiss, B. Levin, and M. Paik. *Statistical Methods for Rates and Proportions (3rd edition)*. Wiley, John and Sons, Inc., 2003.

[42] L. Gao, Z. Tang, X. Lin, Y. Liu, R. Qiu, and Y. Wang. Structure extraction from pdf-based book documents. In *JCDL*, pages 11–20, 2011.

[43] L. Gao, Z. Tang, X. Lin, X. Tao, and Y. Chu. Analysis of book documents' table of content based on clustering. In *ICDAR*, pages 911–915, 2009.

[44] J. Graupmann, R. Schenkel, and G. Weikum. The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents. In *VLDB*, pages 529–540, 2005.

[45] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. DOM-based content extraction of HTML documents. In *WWW*, pages 207–214, 2003.

[46] K. Hatano, H. Kinutani, M. Yoshikawa, and S. Uemura. Information retrieval system for XML documents. In *DEXA*, pages 758–767, 2002.

[47] K. Hatano, T. Shimizu, J. Miyazaki, Y. Suzuki, H. Kinutani, and M. Yoshikawa. Ranking and presenting search results in an RDB-based XML search engine. In *INEX*, pages 156–169, 2007.

[48] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya. Robust web page segmentation for mobile terminal using content-distances and page layout information. In *WWW*, pages 361–370, 2007.

[49] J. He, V. Hollink, and A. de Vries. Combining implicit and explicit topic representations for result diversification. In *SIGIR*, pages 851–860, 2012.

[50] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *SIGIR*, pages 305–314, 2012.

[51] D. Jiang and W. Ng. Mining web search topics with diverse spatiotemporal patterns. In *SIGIR*, pages 881–884, 2013.

[52] T. Kanungo and D. Orr. Predicting the readability of short web summaries. In *WSDM*, pages 202–211, 2009.

[53] M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *SIGIR*, pages 178–185, 1997.

[54] M. Keller and H. Hartenstein. GRABEX: A graph-based method for web site block classification and its application on mining breadcrumb trails. In *WI*, pages 290–297, 2013.

[55] M. Keller and M. Nussbaumer. MenuMiner: Revealing the information architecture of large web sites by analyzing maximal cliques. In *WWW*, pages 1025–1034, 2012.

[56] S.-J. Kim and J.-H. Lee. Subtopic mining using simple patterns and hierarchical structure of subtopic candidates from web documents. *Inf. Process. Manage.*, 51(6):773–785, 2015.

[57] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[58] C. Kohlschütter and W. Nejdl. A densitometric approach to web page segmentation. In *CIKM*, pages 1173–1182, 2008.

[59] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174, 1977.

[60] L. Leal, F. Scholer, and J. Thom. RMIT at INEX 2011 snippet retrieval track. In *INEX*, pages 300–305, 2011.

[61] S. Liang, S. Devlin, and J. Tait. Evaluating web search result summaries. In *Adv. in Info. Retrv.*, pages 96–106. 2006.

[62] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from web documents. In *KDD*, pages 588–593, 2002.

[63] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009.

[64] Y. Liu, R. Song, M. Zhang, Z. Dou, T. Yamamoto, M. P. Kato, H. Ohshima, and K. Zhou. Overview of the NTCIR-11 IMine task. In *NTCIR*, 2014.

[65] C. Lu, L. Bing, and W. Lam. Structured positional entity language model for enterprise entity retrieval. In *CIKM*, pages 129–138, 2013.

[66] W. Lu, S. Robertson, and A. MacFarlane. Field-weighted XML retrieval based on BM25. In *INEX*, pages 161–171, 2006.

[67] C. Luo, X. Li, A. Khodzhaev, F. Chen, K. Xu, Y. Cao, Y. Liu, M. Zhang, and S. Ma. THUSAM at NTCIR-11 IMine task. In *NTCIR*, 2014.

[68] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60, 2014.

[69] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Inf. Retr.*, 10(3):257–274, 2007.

[70] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the web using tag path clustering. In *WWW*, pages 981–990, 2009.

[71] Y. Mizuuchi and K. Tajima. Finding context paths for Web pages. In *HT*, pages 13–22, 1999.

[72] C. Monz. Minimal span weighting retrieval for question answering. In *IR4QA, SIGIR*, pages 23–30, 2004.

[73] J. G. Moreno and G. Dias. HULTECH at the NTCIR-10 INTENT-2 task: Discovering user intents through search results clustering. In *NTCIR*, 2013.

[74] J. G. Moreno and G. Dias. HULTECH at the NTCIR-11 IMine task: Mining intents with continuous vector space models. In *NTCIR*, 2014.

[75] P. Ogilvie and J. Callan. Hierarchical language models for xml component retrieval. In *INEX*, pages 224–237, 2004.

[76] P. Ogilvie and J. Callan. Hierarchical language models for XML component retrieval. In *INEX*, pages 224–237, 2005.

[77] H. Okada and H. Arakawa. Automated extraction of non <h>-tagged headers in webpages by decision trees. In *SICE*, pages 2117–2120, 2011.

[78] E. Oomoto and K. Tanaka. OVID: design and implementation of a video-object database system. *IEEE Trans. on Knowl. and Data Eng.*, 5(4):629–643, 1993.

[79] S. Oyama and K. Tanaka. Query modification by discovering topics from web page structures. In *APWeb*, pages 553–564, 2004.

[80] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

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

[82] F. C. Pembe and T. Güngör. A tree learning approach to web document sectional hierarchy extraction. In *ICAART*, pages 447–450, 2010.

[83] J. Pérez-Iglesias, J. R. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein. Integrating the Probabilistic Models BM25/BM25F into Lucene. *CoRR*, abs/0911.5046, 2009.

[84] M. F. Porter. An algorithm for suffix stripping. In *Readings in Information Retrieval*, pages 313–316. Morgan Kaufmann Publishers, 1997.

[85] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *ECIR*, pages 207–218, 2003.

[86] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM*, pages 42–49, 2004.

[87] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.

[88] T. Sakai, Z. Dou, T. Yamamoto, Y. Liu, M. Zhang, and R. Song. Overview of the NTCIR-10 INTENT-2 task. In *NTCIR*, 2013.

[89] H. Sano, S. Shiramatsu, T. Ozono, and T. Shintani. A web page segmentation method based on page layouts and title blocks. *IJCSNS*, 11(10):84–90, 2011.

[90] K. Simon and G. Lausen. ViPER: augmenting automatic information extraction with visual perceptions. In *CIKM*, pages 381–388, 2005.

[91] H. Sleiman and R. Corchuelo. A survey on region extractors from web documents. *TKDE*, 25(9):1960–1981, 2013.

[92] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning important models for web page blocks based on layout and content analysis. *SIGKDD Explor. Newsl.*, 6(2):14–23, 2004.

[93] R. Song, M. J. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. In *ECIR*, pages 346–357, 2008.

[94] R. Song, M. Zhang, T. Sakai, M. P. Kato, Y. Liu, M. Sugimoto, Q. Wang, and N. Orii. Overview of the NTCIR-9 INTENT task. In *NTCIR*, 2011.

[95] K. M. Svore, P. H. Kanani, and N. Khan. How good is a span of terms?: Exploiting proximity to improve web retrieval. In *SIGIR*, pages 154–161, 2010.

[96] K. Tajima, Y. Mizuuchi, M. Kitagawa, and K. Tanaka. Cut as a querying unit for WWW, Netnews, and E-mail. In *HT*, pages 235–244, 1998.

[97] S. Takami and K. Tanaka. Web-snippet generation suitable for search purpose in web search results. *IPSJ Journal*, 49(4):1648–1656, 2008 (in Japanese).

[98] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *SIGIR*, pages 295–302, 2007.

[99] Y. Tatsumi and T. Asahi. Analyzing web page headings considering various presentation. In *WWW*, pages 956–957, 2005.

[100] C. Tian, T. Tezuka, S. Oyama, K. Tajima, and K. Tanaka. Improving web retrieval precision based on semantic relationships and proximity of query keywords. In *DEXA*, pages 54–63, 2006.

[101] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR*, pages 2–10, 1998.

[102] M. Trappett, S. Geva, A. Trotman, F. Scholer, and M. Sanderson. Overview of the INEX 2013 snippet retrieval track. In *CLEF*, 2013.

[103] M. Z. Ullah and M. Aono. Query subtopic mining for search result diversification. In *ICAICTA*, pages 309–314, 2014.

[104] M. Z. Ullah, M. Aono, and M. H. Seddiqui. SEM12 at the NTCIR-10 INTENT-2 english subtopic mining subtask. In *NTCIR*, 2013.

[105] C. Wang, M. Danilevsky, N. Desai, Y. Zhang, P. Nguyen, T. Taula, and J. Han. A phrase mining framework for recursive construction of a topical hierarchy. In *KDD*, pages 437–445, 2013.

[106] C.-J. Wang, Y.-W. Lin, M.-F. Tsai, and H.-H. Chen. Mining subtopics from different aspects for diversifying search results. *Inf. Retr.*, 16(4):452–483, 2013.

[107] J. Wang, G. Tang, Y. Xia, Q. Zhou, T. F. Zheng, Q. Hu, S. Na, and Y. Huang. Understanding the query: THCIB and THUIS at NTCIR-10 intent task. In *NTCIR*, 2013.

[108] Q. Wang, Y. Qian, R. Song, Z. Dou, F. Zhang, T. Sakai, and Q. Zheng. Mining subtopics from text fragments for a web query. *Inf. Retr.*, 16(4):484–503, 2013.

[109] S. Wang, Y. Hong, and J. Yang. Pku at inex 2011 xml snippet track. In *INEX*, pages 331–336, 2011.

[110] T. Watanabe. Experimental evaluation of usability and accessibility of heading elements. In *W4A, WWW*, pages 157–164, 2007.

[111] Y. Xia, X. Zhong, G. Tang, J. Wang, Q. Zhou, T. F. Zheng, Q. Hu, S. Na, and Y. Huang. Ranking search intents underlying a query. In *NLDB*, pages 266–271, 2013.

[112] Y. Xue, F. Chen, A. Damien, C. Luo, X. Li, S. Huo, M. Zhang, Y. Liu, and S. Ma. THUIR at NTCIR-10 INTENT-2 task. In *NTCIR*, 2013.

[113] T. Yamamoto, M. P. Kato, H. Ohshima, and K. Tanaka. KUIDL at the NTCIR-11 IMine task. In *NTCIR*, 2014.

[114] M. Yoshii, J. Miyazaki, K. Hatano, and H. Kato. A study on XML information retrieval using partial document clustering. In *DEIM*, 2009 (in Japanese).

[115] H. Yu and F. Ren. TUTA1 at the NTCIR-11 IMine task. In *NTCIR*, 2014.

[116] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR*, pages 210–217, 2004.

[117] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW*, pages 76–85, 2005.

[118] L. Zhang, Y. Zhang, and Y. Chen. Summarizing highly structured documents for effective search interaction. In *SIGIR*, pages 145–154, 2012.

[119] W. Zheng, H. Fang, H. Cheng, and X. Wang. Diversifying search results through pattern-based subtopic modeling. *Int. J. Semant. Web Inf. Syst.*, 8(4):37–56, 2012.

[120] W. Zheng, X. Wang, H. Fang, and H. Cheng. An exploration of pattern-based subtopic modeling for search result diversification. In *JCDL*, pages 387–388, 2011.

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to Prof. Keishi Tajima, the mentor of my undergraduate thesis, master's thesis, and this doctoral dissertation. There is no doubt that most of my research methodology and skills are inherited from him through his six-year mentoring. Because of the persistent and detailed mentoring, our weekly man-to-man meetings sometimes reached to three hours. I would like to express my gratitude to him for spending such a long time only for educating me. He also taught me academic writing in both English and Japanese by revising our papers again and again in spite of his other hard work. For example, our VLDB paper was revised 12 times before submission without fundamental change. I am also indebted to him for his countless number of comments on our TEXsources.

Next I would particularly like to thank the reviewers of this dissertation, Prof. Katsumi Tanaka and Prof. Masatoshi Yoshikawa. Prof. Tanaka pointed out some severe, but interesting problems for making our proposed methods more practical. I appended discussion on the problems to the last of this dissertation.

Prof. Yoshikawa, the other reviewer, is also an adviser of my master's thesis and this dissertation. He is an expert of XML retrieval and his questions and advises always reminded me the area of XML retrieval, close to but different from the area of web search. Thanks to him, many citations of XML papers were made for improving the degree of perfection of this dissertation.

I would like to offer my special thanks to the other adviser of this dissertation, Prof. Tetsuya Sakai at Waseda University. He pointed out many flaws of my experimental evaluations from many points of view, e.g., topic set size, agreement of assessors, evaluation measures, baselines, fine tuning, and so on. His insightful comments and recently published handbook statistically significantly improved this dissertation.

I would also like to thank the other adviser of my master's thesis, Prof. Yasushi Sakurai at Kumamoto University. He gave me considerable inspirations about applications of our heading-based page segmentation method, which contributed to my master's thesis and also to this dissertation, of course.

Acknowledgments

I have greatly benefited from all other (former or current) professors, staffs, and students in the Digital Library group of the Department of Social informatics, Kyoto University. Especially, Prof. Adam Jatowt taught me the basis of information retrieval and of oral communication in English in IR *rinkoh* (reading society), which is held just after I joined the group. Furthermore, he recently helped me to improve my presentation at an international conference. Prof. Hiroaki Ohshima taught us students the first things about laboratory life and made noteworthy effort for building better research environment. Prof. Yusuke Yamamoto also supported the IR *rinkoh*. Prof. Makoto P. Kato, Prof. Takehiro Yamamoto, and Dr. Kosetsu Tsukuda supported and encouraged our research, my laboratory life, our papers, and so on. I also received generous support of the secretaries of the group, Ms. Sato, Ms. Shiraishi, and Ms. Ashiwa, for filling out complicated paperwork. The senior students under Prof. Tajima, Mr. Tanaka and Mr. Inoue, particularly supported the beginning of my laboratory life. Dr. Yoshiyuki Shoji (a one-year senior student), Mr. Kazutoshi Umemoto (a same-year student), and I helped each other to finish our doctoral dissertations. I especially thank some of the other students, namely Mr. Takemura, Mr. Ohshige, Mr. Kadowaki., Mr. Tanaka, Mr. Takeda, Mr. Nakano, and Mr. Hashimoto for participating the experiments we carried out.

I would also like to express my gratitude to the Japan Society for the Promotion of Science (JSPS) for their financial support. This dissertation was supported by JSPS *KAKENHI* Grant Number 13J06384.

Last of all, I would like to thank my parents, Dr. and Ms. Manabe, my sister and brothers, and my fiance, Ms. Takesue.

Publications and First Appearances

Journal Papers

1. Tomohiro Manabe and Keishi Tajima.
   Extracting Logical Hierarchical Structure of HTML Documents Based on Headings.
   The Proceedings of the VLDB Endowment, vol. 8, no. 12, pp. 1606–1617, 2015.
   (the first appearance of Section 2.1 and Chapter 3)

2. Tomohiro Manabe and Keishi Tajima.
   Heading-Aware Proximity Measure and Its Application to Web Search.
   DBSJ Journal, vol. 14, article no. 2, 2016.
   (the first appearance of Section 2.3 and Chapter 5)

International Conference Papers

1. Tomohiro Manabe and Keishi Tajima.
   Heading-Aware Snippet Generation for Web Search.
   In *Proceedings of the 11th Asia Information Retrieval Societies Conference*, Lecture Notes in Computer Science, vol. 9460, pp. 188–200, 2015.
   (the first appearance of Section 2.5 and Chapter 7)

2. Tomohiro Manabe and Keishi Tajima.
   Subtopic Ranking Based on Hierarchical Headings.
   In *Proceedings of the 12th International Conference on Web Information Systems and Technologies*[1], 2016 (to appear).
   (the first appearance of Section 2.2 and Chapter 4)

[1] WEBIST `http://www.webist.org/`

International Workshop Paper

1. Tomohiro Manabe, Kosetsu Tsukuda, Kazutoshi Umemoto, Yoshiyuki Shoji, Makoto P. Kato, Takehiro Yamamoto, Meng Zhao, Soungwoong Yoon, Hiroaki Ohshima, and Katsumi Tanaka.
   Information Extraction based Approach for the NTCIR-10 1CLICK-2 Task.
   In *Proceedings of the 10th NTCIR Conference*, pp. 243–249, 2013.

Domestic Symposium and Workshop Papers

1. 真鍋知博, 田島敬史:
   “Web ページ中のノード間の論理的関係の発見”,
   第 3 回データ工学と情報マネジメントに関するフォーラム, 2011.

2. 飯村結香子, 真鍋知博, 塩原寿子, 内山匡:
   “EC サイトからの商品情報抽出ルールの自動生成”,
   情報処理学会研究報告, vol. 2012-IFAT-105（または 2012-NL-105）, no. 13, pp. 1–7, 2012.

3. 真鍋知博, 田島敬史:
   “Web ページ中の階層的見出し構造の発見”,
   第 4 回データ工学と情報マネジメントに関するフォーラム, 2012.

4. 真鍋知博, 田島敬史:
   “階層的見出し構造に着目した Web ページ検索システム”,
   第 5 回データ工学と情報マネジメントに関するフォーラム, 2013.