Website Summarization:

A Topic Hierarchy Based Approach

LIU Nan

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Systems Engineering and Engineering Management

August 2006

Thesis/Assessment Committee

Professor Lam Wai (Chair)
Professor Yang Chuen Chi (Thesis Supervisor)
Professor Yu Xu (Committee Member)
Professor Wang Ke (External Examiner)

Abstract

Web users often need to look for particular pages containing some specific information within a large Website. Traditionally, this process is accomplished through clicking a series of links to move from page to page, which can be very time consuming when faced with Websites consisting of hundreds or even thousands of pages. Unlike the World Wide Web, most Websites are developed by a single party and the content of a Website is usually well organized. Providing an overview of the Website's content structure such as a sitemap can help a user to locate useful information much more efficiently. In addition to assisting Website navigation, modeling Website's content structure is also essential to various Website mining applications such as Website classification and summarization. Hierarchical models are most commonly used to organize a Website's content. A Website's content structure can be represented by a topic hierarchy, which is a directed tree rooted at a Website's homepage whose vertices and edges correspond to web pages and hyperlinks. In this thesis, we first studied the problem of automatically generating a Website's topic hierarchy. We modeled a Website's link structure as a weighted directed graph and proposed methods for estimating the edge weights. Several graph algorithms were adapted to generate the topic hierarchy based on the graph model. We have tested the model and algorithm on real Websites and achieved very promising results. We then studied algorithms for generating keywords for each Web page in the topic hierarchy. In particular, we proposed features that take into account a Web page's location in the topic hierarchy. The quality of the extracted keywords was evaluated by human judges and the results showed that utilizing topic hierarchy information could result in significant improvement in terms of the quality of the extracted keywords.

摘要

在互联网的日常应用中，用户经常需要在大型网站中查找含有某些具体信息的个别网页。在目前的条件下，人们主要通过点击一系列的链接实现在网页之间的移动。在面对大型网站时，此种浏览方式极其耗时。与万维网有所分别的是，大部分网站通常由个别组织或个人开发，因而有良好的内容规划。提供网站内容结构的纵览可以有效提高用户在网站中查找信息的效率。此外，网站内容结构对于如网站分类，网站摘要等网站数据挖掘应用都有重要价值。现有的大部分网站均采用树形结构进行内容规划。在本文中，我们提出使用主题树作为网站内容结构的模型。在网站的主题树中，网站主页将作为树的根，而其他网页和链接则对应为树的节点及边。我们首先讨论了自动生成网站主题树的方法。我们提出利用有向图模拟网站的链接结构，并使用不同的图算法生成主题树。通过使用真实网站对提出的模型及算法进行测试，我们取得了良好的试验结果。本文接下来进一步讨论了识别网页关键词的算法并提出了基于网页在主题树中的位置的识别关键词特征。而对生成的关键字质量的评估结果证明利用网页在主题树中的位置可以显著提高关键词识别算法的性能。

Acknowledgements

I would like to express my gratitude to the following organizations and people for the support and contribution:

The Chinese University of Hong Kong for offering the scholarship and the opportunity to pursue both my bachelor and master degrees.

Professor Yang Chuen-Chi, Christopher, my supervisor, for his invaluable guidance, encouragement, understanding and sharing of his knowledge.

Professors and students at the Department of Systems Engineering and Engineering Management, for their generous sharing of ideas, suggestions, and discussions.

Thank you!

Contents

List of Figures

List of Tables

Chapter 1 Introduction

With rapid growth of the World Wide Web (WWW), enormous amount of information can now be accessed from the web, making it the largest source of information. Nevertheless, finding and extracting information from the Web, whether manually or automatically can be extremely difficult. The invention of search engines helps alleviate this problem by enabling a user to find pages by typing in a set of keywords. However, to use a search engine effectively, a user needs to be able to express his information need through a query, which poses a challenge to inexperienced web users or those who has an ambiguous information need that is hard to express.

Users looking for information on the web frequently need to navigate within a certain Website to locate particular web pages satisfying his needs. However, effective website exploration is not an easy task. Since most current web browsers are able to display only one web page at a time, users have to follow the hyperlinks to move back and forth on the Website's link graph in hopes of finding interesting pages, which is a tedious and time consuming process and even prohibitive when user is accessing the web using handheld devices which have very limited bandwidth. From a user's perspective, such problems can be greatly eased if a "table-of-content" of the Website is available since a user can quickly narrow down his search scope to particular group(s) of pages about topics relevant to the user's need. Some well designed websites provide sitemaps as a kind of navigational support. A sitemap provides an overview of a website's content structure by listing a few most important pages that correspond to the major topics of the website (Figure 1-1). Through examining the site map, a user can quickly locate relevant information without having to visit many pages. Despite the usefulness of sitemaps, they are not provided by most

Websites. Moreover, most existing sitemaps cover only a limited number of pages, which account for only a very small proportion of the Website. This is because most sitemaps have to be constructed manually, and thus are very difficult to be expanded to include hundreds or even thousands of Web pages. Thus a sitemap can only help a user locate Web pages about very broad topics. To look for more specific information, the user could only use the broad topics on the sitemap as entry point and continue exploring by following hyperlinks. For example, to look for Prof. Johnson's publications, one need to first go to the "Academic Staff" page, then find the page correspond to "Homepage of Prof. Johnson" and finally find "Prof. Johnson's Publication".

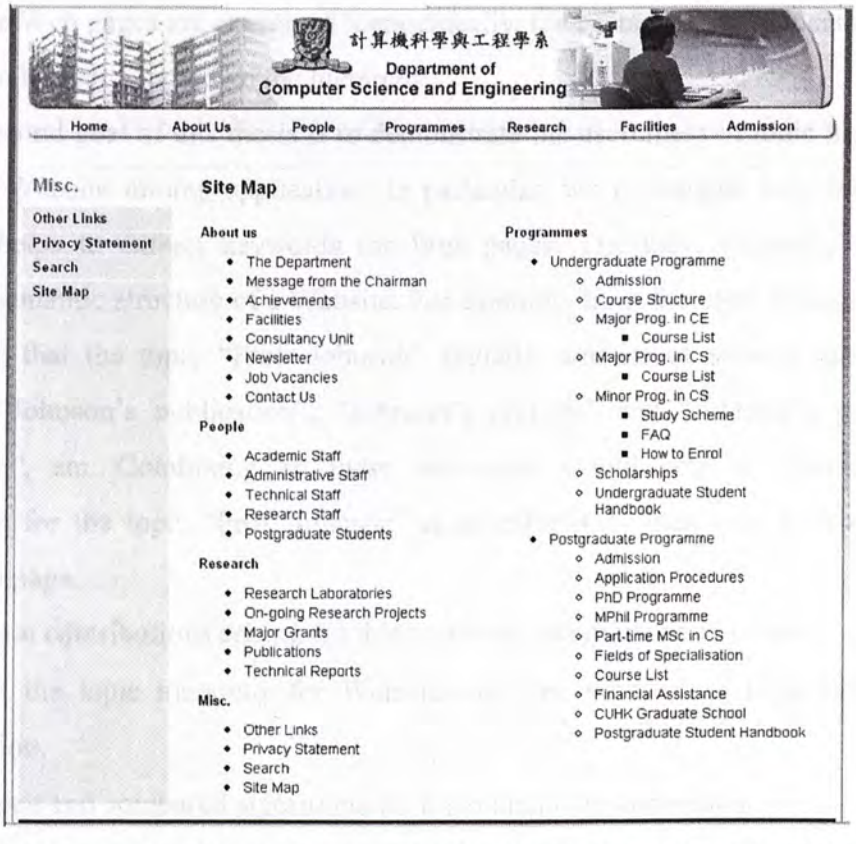

Figure 1-1 Example sitemap for a computer science department's Website

A Website's content is usually divided into a set of different topics. Each topic may in turn be further divided into more sub-topics. A web designer uses many different approaches to organize the large amount of information, such as connecting related content through hyperlinks and grouping related files through folders. The first goal of this thesis is to investigate how the underlying content organization of a general Web site can be automatically discovered. In particular, we proposed algorithms to automatically generate the topic hierarchy for a given Website. A topic hierarchy models the topic/sub-topic relationships between Web pages. For example, "Prof. Johnson" is a sub-topic of "Academic Staff", and "Prof. Johnson's publication" is a sub-topic of "Prof. Johnson". By automatically generating the topic hierarchy, we can build a complete sitemap covering every page in a Website. And because the Web pages are organized hierarchically, the users can easily locate useful information by searching down the hierarchy.

The second goal of this thesis is to demonstrate the usefulness of topic hierarchy in general Website mining application. In particular, we investigate how the topic hierarchy helps to extract keywords for Web pages. The topic hierarchy reveals important semantic structure of a Website. For example, from the topic hierarchy, we may notice that the topic "Prof. Johnson" actually consists of several sub-topics including "Johnson's publication", "Johnson's projects" and "Johnson's personal information", etc. Combining all these sub-topics would help us identify key information for the topic "Prof. Johnson" more effectively than only looking at a single homepage.

The main contributions arising fro this thesis are summarized as follows:

- Defined the topic hierarchy for Websites and the problem of topic hierarchy generation.
- Developed and compared algorithms for topic hierarchy generation.
- Developed and compared algorithms for extracting keywords from Web page based on the topic hierarchy.

The outline of the thesis is organized as follows.

- In Chapter 2, we survey the existing works closely related to this research.
- In Chapter 3, we describe the topic hierarchy generation algorithms and the results of evaluation.
- In Chapter 4, we present the method for extracting keywords from Web pages based on the topic hierarchy and assess its performance.

Chapter 2 Related Work

The ever increasing volume of unstructured text and hypertext data on the Web has aroused much interest from diverse research areas such as information retrieval, data bases and natural language processing. The problem of generating topic hierarchy is part of the more general web mining research. Most existing web mining research focused on models and techniques for dealing with either the entire World Wide Web or individual Web pages, which can be partially attributed to the success of general web search engines such as Google, etc. In recent years, there have been an increasing number of works about Website mining, which focus on handling particular Websites instead of the entire WWW or individual Web pages. In this chapter, we will examine several closely related works in web mining, in particular, web structure mining and Website mining.

2.1 Web Structure Mining

In web structure mining, one discovers knowledge from the connectivity of web. The nature of web structure mining is similar to the social network analysis and citation network analysis. By analyzing the incoming and outgoing links of web pages, the importance of Web pages and other information, such as authorities and hubs, can be identified (Kleinberg 1998). Authority pages are web pages containing large amount of information relevant to a given user query, while the hub pages are the ones providing links to authorities. Therefore, authority and hub pages are the most important pages answering user queries. Identifying the most authoritative pages for a user query is one of the research topics in web structure mining (Chakrabarti 2000). Several algorithms have been proposed for mining the authoritative pages. The

most notable among them are the Hyperlink Induced Topic Search (HITS) and PageRank algorithms.

2.1.1 HITS Algorithm

Kleinberg (1998) developed an algorithm to discover authority and hub pages from a large collection using the links among them including in-links and out-links. The assumption adopted by the HITS algorithm is that a good authority page should be pointed by good hubs while a good hub should point to good authorities. By linking a page p to another page q, the creator of p confers some authority on q. In other words, there would not be a link from p to q (denoted by $p \rightarrow q$) if q is not important p. Thereafter, authorities and hubs exhibit a mutually reinforcing relationship.

A query to HITS is forwarded to a search engine, which retrieves a sub-graph of the web whose nodes match the query. Pages citing or cited by these pages are also included. Each node p in this expanded graph has two associated scores $h(p)$ and $a(p)$, initialized to 1. HITS then iteratively assigns

$$h(p) = \sum_{q:p->q} a(q) \quad \text{and} \quad a(p) = \sum_{q:q->p} h(q) \tag{2.1}$$

where $h(p)$ and $a(p)$ are normalized to 1 after each iteration. The hub and authority scores converge respectively to the measure of a page being an authority and the measure of a page being a hub.

2.1.2 PageRank Algorithm

Unlike the HITS algorithm which finds the authorities and hubs from a collection of web pages, the PageRank algorithm assigns a score to each web page indicating how important the page is (Brin and Page 1998). The assumption is that: (1) a page is important if it contains links to many important pages; and (2) a page is important if it has high in-degree. Therefore, PageRank $R(q)$ of an arbitrary page q can be defined as follows:

$$R(q) = \sum_{p:p->q} \frac{R(p)}{N_p} \tag{2.2}$$

where N_p is total number of out-links on p. However the web graph is irreducible and there may exist some dangling links created for no reason. The problem is solved by a random jump to a new chosen from the entire web. In particular, if the probability of picking a link from the current page is c, the probability of picking other pages in the web will be $1-c$. The PageRank can then be redefined as:

$$R(q) = c \sum_{p:p->q} \frac{R(p)}{N_p} + \frac{1-c}{V} \tag{2.3}$$

where V is the total number of pages. The higher the PageRank score, the more important is the web page.

2.2 Website Mining

A Website is an organized collection of pages on a specific topic maintained by a single person or group. The link structure of a Website is quite distinct from the Web. Both HITS and PageRank focus on the intra-domain links and a hyperlink $p \rightarrow q$ is generally considered as conferring authority to q by p. However, within the same Website, the hyperlinks are mainly created to enable navigation instead of recommending pages. Therefore, new techniques exploiting the special characteristics of Websites have been developed in a variety of applications including Website classification, Web unit mining and logical domain extraction.

2.2.1 Website Classification

Compared to Web page classification, there have been much less work on classifying Websites. In Website classification, the objects to be classified are entire Websites. The classification of Websites is quite different from that of Web pages because Websites may contain a large number of pages.

In the superpage approach, a Website is represented as single virtual Webpage consisting of the union of all pages from the Website. In other words, each Website

can be treated as one document and the normal document classification methods can be directly applied. The advantage of the superpage approach is that it is simple to implement. However, it ignores the structural information about an entire Website. Because of its simplicity, it is generally considered as baseline methods for Website classification.

Ester et al (2002) conducted experiments on a dataset of corporate Websites belonging to two different industries (IT-Service and Florist). Using the superpage approach, the classification accuracy of 55.6% was achieved using Naïve Bayes classifier.

The keyword vector approach represents a Website by a vector of keyword frequencies. In this representation, the content of each Webpage is summarized as a keyword taken from a pre-defined vocabulary. In other words, a keyword is assigned to each page. The predefined list of keywords may be constructed based on the nature of the Website. The assignment of keywords to Webpages can be achieved using Webpage classification techniques. In the experiments using the same corporate Website datasets, the keyword approach using Naïve Bayes classifier achieved 78.7% in accuracy. (Ester et. al. 2002)

In the tree-based approach, a Website tree is constructed using web pages and the links among them to capture the Website's structure. Starting with the Website's homepage, a Website tree is built by performing a breadth-first search. The idea is that most Websites have hierarchical structure (Ester et. al. 2002). They normally present information from homepage which contains general information to the pages that carry more specific information and need to be reached through several links. Each web page is represented by a keyword and the Website tree is therefore a tree of keywords. By classifying the Website tree using a method developed based on Markov tree model, 86% accuracy was achieved on corporate datasets.

2.2.2 Web Unit Mining

Sun and Lim (2003) showed that information about a particular concept is usually not presented on a single Webpage but a set of semantically related pages connected by hyperlinks Within a Website. For example, a professor's home page would contain links to pages describing his research, curriculum vitae and education etc. These pages together represent a professor instance. This leads to the definition of Web unit, which is a set of web pages representing the same concept instance. Web unit mining is to determine the set of web pages constituting web units and classify these web units into different concepts in a given ontology. The Iterative Web Unit Mining (iWUM) algorithm carries out web unit construction and web unit classification iteratively (Sun and Lim 2003). Initially, a set of web units are generated using the Website's directory structure. In each iteration, the web units are classified into a set of domain-specific concept labels and then recombined to form larger web units. This process continues until there are no further changes to the constructed web units and their labels. To apply this method to a Website of a new domain, one needs to have a domain specific ontology defining the different concepts and also classifiers for classifying web units into each concept.

2.2.3 Logical Domain Extraction

The logical domain extraction problem (Li et al. 2000) is very similar to web unit mining. A logical domain is a group of pages that have a specific semantic relation and are related a syntactic structure in a Website. A logical domain is essentially a mini site with a large Website, such as a professor's personal Website within his department's Website.

Li et al. assumes that there exists an entry page for each logical domain that is supposed to be the first page to be visited by users navigating the logical domain. Once the entry page was identified, the other pages in the domain can be extracted based on the link structure and folders. The key problem in logical domain extraction becomes the identification of the entry pages (Li et al. 2000). Li et al. developed a

rule-based approach for identifying entry pages based on Web page metadata including titles, URL, anchors, contents, link structure and citation. The technique was able to identify logical domain entry pages with high precision. However, because the method relies on very specific rules, the recall was rather low with many entry pages being left out. In addition, the extracted logical domains tend to be small and correspond to very specific topics such as a professor, a class, etc. The result is therefore a very long list of different types of logical domains, which is very hard to browse for a user.

2.2.4 Web Thesaurus Construction

An important feature of the web's link structure is topic locality (Davison 2000). Topic locality means that the web pages connected by hyperlinks are more likely to be about the same topic than those unconnected. Chen et al (2003) suggest that by replacing each web page by their anchor texts, a Website's link structure can be treated as a semantic network, in which words appeared in the anchor text are nodes and semantic relations are edges. Hence, it is possible to construct a thesaurus by using this semantic network information.

Chen et at (2003) pointed out that there are two functions for a hyperlink in a Website: one for navigation convenience and the other for connecting semantically related web pages together. To use the link structure as a semantic network, the navigational links should be removed while the semantic links should be retained. A link is classified as a navigation link if it is one the following:

1. The target page is in a parent folder of the source page
2. The link is in a navigation bar and the target page is not in a subdirectory of the source page.
3. The link occurs in many web pages.

Experimental results showed that the proposed rules can identify navigation links with high precision. However, the recall was not measured, so the quality of the cleansed link structure is not assured. Moreover, this simple approach doesn't

consider the topology of resulted link structure. After the removal of recognized navigation link, the link structure remains to be a complicated graph instead of a well defined hierarchy that we are looking for.

Chapter 3 Website Topic Hierarchy Generation

The content of World Wide Web is highly unstructured due to its totally decentralized growth. In contrast, Websites are much more structured as their development usually involves careful planning and design. A necessary step in developing a Website is content organization. The hierarchical model is frequently used for the organization of complex bodies of information on websites for its simplicity and clarity (Lynch and Horston 2002). A large Website usually consists a number of major topics, which may be recursively divided into a number of subtopics. For example, the major topics on Stanford Database Group's Website include "Members", "Projects", "Classes", etc. Within the topic "Projects", each individual projects such as "Data Stream" form its subtopics. Similarly, sub topics can be further divided into sub subtopics. In a website, all the topics and subtopics correspond to Web pages. In topic hierarchy generation, we attempt to organize a Website's pages into a tree structure so that the topic-subtopic relations between the Web pages are reflected by the parent-child relations in the tree.

3.1 Problem Definition

Our approach for generating topic hierarchy is based on analyzing the Website's link structure, which is typically a densely connected graph as shown in the left part of Figure 3-1. To formally define topic hierarchy, we first define two types of hyperlinks based on their purpose. When designing a Website, there are always hyperlinks pointing from a topic to its subtopics, as these are the essential links for browsing a Website. We refer to the hyperlinks connecting a topic to its subtopics as ***aggregation links***, which are indicated by the solid arrows in Figure 3-1. In the mean time, there exists a large number of hyperlinks not used to connect topics to subtopics, but are created to provide a quick way for moving from page to page, which we refer to as

short-cut links. These are indicated by the dashed arrows in Figure 3-1. For example, a professor's page may link to a class he teaches or a project he is involved in. Also, when developing large sites, a common practice is to use templates for creating groups of similar Web pages, which often contains a navigation bar linking to a few most important pages. The existence of these short-cuts poses the challenge in generating a topic hierarchy based on the link structure, which is to distinguish aggregation links from short-cut links. Figure 3-1 illustrates the original link structure with both aggregation and short-cut links, as well as the topic hierarchy formed by only the aggregation links. Notice that there are as many as 10 short-cut links versus only 5 aggregation links in the original link structure.

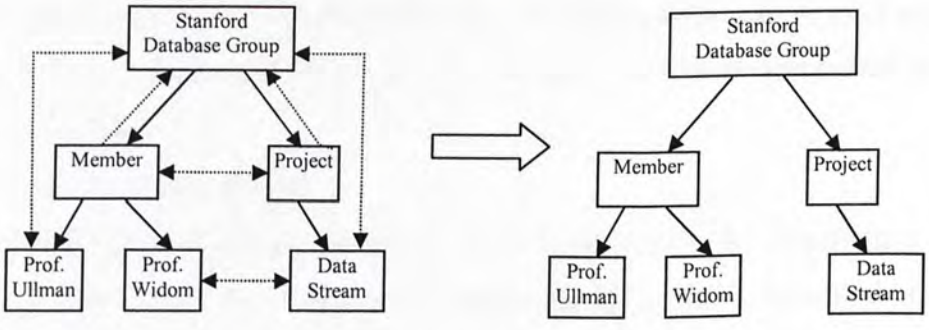

Figure 3-1 Link Structure vs. Topic Hierarchy

The topic hierarchy for a Website is formally defined as a directed tree with all of the following properties:

- It is rooted at the homepage of the Website.
- Its vertices include all the pages reachable from the Website's homepage by following a sequence of hyperlinks.
- Each edge in the tree corresponds to an aggregation link pointing from the parent node (topic) to the child node (subtopic).

3.2 Graph Based Algorithms

Our algorithms for generating topic hierarchy takes a Website's link structure, which is a general graph, as input and extracts from it a sub-graph, which has to be a tree structure. Several graph algorithms have been adapted for solving this problem, including breadth first search, shortest-path search and minimum directed spanning tree(Liu and Yang 2005a, 2005b). All three algorithms are capable of extracting a tree from an input graph, but are different in terms of the input graph representation and criteria used to evaluate edges for adding to the tree. Breadth first traversal works on unweighted graph and minimizes the number of hops from the root to other nodes. Shortest path search and minimum directed spanning tree handle weighted graph and minimizes the weight of the paths from the root to other nodes and the total weight of all the edges respectively. In the next few sections, we will describe each of these algorithms in detail.

3.2.1 Breadth First Search

Breadth-first search is one of the simplest algorithms for searching a graph. Given a graph $G = (V, E)$ and a distinguished ***source*** vertex s, breadth-first search systematically explores the edges of G to "discover" every vertex that is reachable from s. It computes the distance (smallest number of edges) from s to each reachable vertex. It also produces a "breadth-first tree" with root s that contains all reachable vertices. For any vertex v reachable from s, the path in the breadth-first tree from s to v corresponds to a "shortest path" from s to v in G, that is, a path containing the smallest number of edges.

Breadth-first search is so named because it expands the frontier between discovered and undiscovered vertices uniformly across the breadth of the frontier. That is, the algorithm discovers all vertices at distance k from s before discovering any vertices at distance $k + 1$. Algorithm 4-1 shown below is a variation of the

Algorithm 3-1 Breadth-First Search

BFS(G, s)
Input:
    G (V, E): *the Website's link structure*
    s: *source vertex*
Output:
    $G'(V', E')$: *the subgraph of G corresponding to the topic hierarchy*

```
1  V' ← Ø
2  E' ← Ø
3  for each vertex u ∈ V - {s}
4     do color[u] ← WHITE
5        d[u] ← ∞
6  color[s] ← GRAY
7  d[s] ← 0
8  Q ← Ø
9  ENQUEUE(Q, s)
10 while Q ≠ Ø
11    do u ← DEQUEUE(Q)
12    V' ← V' ∪ {u}
13       for each v such that (u, v) ∈ E
14          do if color[v] = WHITE
15                then color[v] ← GRAY
16                     d[v] ← d[u] + 1
17                     E' ← E' ∪ {(u, v)}
18                     ENQUEUE(Q, v)
19       color[u] ← BLACK
20 return (V', E')
```

breadth-first search algorithm in (Cormen et al. 2003), which outputs the breadth-first tree as the topic hierarchy.

Breadth-first search constructs a breadth-first tree, initially containing only its root, which is the source vertex s. Whenever a new vertex v is discovered in the course of scanning the adjacency list of an already discovered vertex u, the vertex v and the edge (u, v) are added to the tree. And u would become the ***parent*** of v in the breadth-first tree. Since a vertex is discovered at most once, it has at most one parent. When the graph is stored as an adjacency list, the time for performing breadth-first

search is proportional to the time spent in scanning the adjacency list, so the time complexity for this algorithm is $O(|E|)$.

3.2.2 Shortest Path Search

In a ***shortest-paths problem***, we are given a weighted, directed graph $G = (V, E)$, with weight function $w : E \rightarrow \mathbf{R}$ mapping edges to real-valued-weights. The ***weight*** of path $p = \langle v_0, v_1, ..., v_k \rangle$ is the sum of the weights of its constituent edges:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) .$$

We define the ***shortest-path weight*** from u to v by

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{if there is a path from u to v.} \\ \infty & \text{otherwise} \end{cases}$$

A ***shortest path*** from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$. For generating topic hierarchies, we shall focus on the ***single-source shortest-paths problem*:** given a graph $G = (V, E)$, we want to find a shortest path from a given ***source*** vertex $s \in V$ to each vertex $v \in V$.

A famous algorithm for single-source shortest-path search is the Dijkstra's algorithm, which takes as input a weighted directed graph $G = (V, E)$ with nonnegative weight function (i.e. $w(u, v) \geq 0$ for each edge $(u, v) \in E$). Dijkstra's algorithm maintains a priority queue of vertices whose final shortest-path hasn't been determined. The algorithm repeatedly selects the vertex u in the queue with the minimum shortest-path estimate and updates the shortest-path estimate for all nodes adjacent to u. When the weight function w is nonnegative, Dijkstra's algorithm is guaranteed to find the shortest-paths for all vertices and the subgraph formed by the vertices and edges on the shortest-paths is a tree rooted at the source vertex (Cormen et al. 2003).

Algorithm 3-2 shows the Dijkstra's algorithm for building topic hierarchy. The running time of the algorithm depends on how the priority queue is implemented.

Algorithm 3-2 Shortest-Path Search

SPS(G, s)
Input:
G (V, E): *the Website's link structure*
s: *source vertex*
w: *edge weight function*
Output:
$G'(V', E')$: *the subgraph of G corresponding to the topic hierarchy*

```
1  V' ← Ø
2  E' ← Ø
3  for each vertex u ∈ V
4     do d[u] ← ∞
5        π[u] ← NIL
6  d[s] ← 0
7  Q ← V
8  while Q ≠ Ø
9     do u ← EXTRACT-MIN(Q, d)
10    V' ← V' ∪ {u}
11    E' ← (π[u], u)
12    for each v such that (u, v) ∈ E
13       do if d[v] > d[u] + w(u, v)
14       then  d[v] = d[u] + w(u, v)
15             π[v] = u
16 return (V', E')
```

Consider first the case in which we maintain the min-priority queue by taking advantage of the vertices being numbered 1 to $|V|$. We simply store $d[v]$ in the vth entry of an array. Each INSERT and DECREASE-KEY operation takes $O(1)$ time, and each EXTRACT-MIN operation takes $O(V)$ time (since we have to search through the entire array), for a total time of $O(|V|^2+|E|) = O(|V|^2)$.

3.2.3 Minimum Directed Spanning Tree

Minimum spanning tree is a well known problem in graph theory. We firstly review the more well known case when the input graph is undirected. For an

undirected weighted graph $G(V, E)$ with weight function w, we wish to find an acyclic subset T of E that connects all of the vertices and whose total weight

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

is minimized. Since T is acyclic and connects all of the vertices, it must form a tree, which we call a spanning tree since it "spans" the graph G. The problem of determining the tree T is the ***undirected minimum-spanning-tree problem***. There exists very efficient algorithm for determining undirected minimum spanning tree, such as Kruskal's algorithm and Prim's algorithm (Cormen et al. 2003).

The ***directed minimum-spanning tree problem*** takes as input a directed weighted graph $G(V, E)$ and a root vertex s and finds a subset T of E whose total weight is minimized and such that the root s has only outgoing edges while all the other vertices has only one incoming edge. Therefore, there is a unique path from the root to every other vertex in the directed spanning tree.

The topic hierarchy for a Website is exactly a directed spanning tree rooted at the homepage, therefore we could generate the topic hierarchy by finding a minimum directed spanning tree from the Website's link graph. Efficient algorithms for finding directed minimum spanning tree are less well known, but they exist. We will use here the Chu-Liu-Edmonds algorithm (Chu and Liu 1965; Edmonds 1967), sketched in Algorithm 3-3 following the presentation in (Leonidas 2003). Informally, the algorithm has each vertex in the graph greedily select the incoming edge with lowest weight. If a tree results, it must be the minimum directed spanning tree. Otherwise, there must be a cycle. The algorithm identifies a cycle and contracts it into a single dummy vertex and recalculates the weights of edges into and out of the cycle. It can be proved that a directed minimum spanning tree in the contracted graph is equivalent to a minimum spanning tree in the original graph (Leonidas 2003). Hence the algorithm can recursively invoke itself on the new graph. Naively this algorithm runs in $O(|V|^3)$ time since each recursive call takes $O(|V|^2)$ to find the lightest incoming

edge for each word and to contract the graph. There are at most $O(|V|)$ recursive calls since we cannot contract the graph more than n times.

Algorithm 3-3 Directed Minimum Spanning Tree

Chu-Liu-Edmonds(G, s)
Input:
G (V, E): *the Website's link structure*
s: *source vertex*
w: *edge weight function*
Output:
$G'(V', E')$: *the subgraph of G corresponding to the topic hierarchy*

$E' \leftarrow \{(u, v) : v \in V, \mathrm{u} = \arg\max_u w(u, v)\}$
$V' \leftarrow V$
do if $G'=(E', V')$ has no cycles
  then return G'
  else find a cycle C in G'
$G_c \leftarrow$ subgraph of G excluding vertices in C
$x \leftarrow$ dummy vertex representing C
for each $v \in V - C : \exists u \in C \wedge (u,v) \in E$
  add (x, v) to G_c
  $w(x, v) \leftarrow \min_{u \in C} w(u,v)$
for each $u \in V - C : \exists v \in C \wedge (u,v) \in E$
  add (u, x) to G_c
  $w(u, x) \leftarrow \min_{v \in C}[w(u,v) - w(\pi(v),v) + w(C)]$ where $\pi(v)$ is the predecessor of v in C and $w(C) = \sum_{v \in C} w(\pi(v),v)$
$(V', E') \leftarrow$ Chu-Liu-Edmonds(G_c, s)
do if $\exists (x,v) \in E'$
  then find a vertex u in C s.t. $(u,v) \in E$, $(u,v') \in C$ and $w(u,v) = w(x,v)$
    $E' \leftarrow E' \cup \{(u,v)\} \cup C - \{(u,v')\}$
do if $\exists (u,x) \in E'$
  then find a vertex v in C s.t. $(u,v) \in E$, $(u',v) \in C$ and $w(u,v) - w(\pi(v),v) + w(C) = w(x,v)$
    $E' \leftarrow E' \cup \{(u,v)\} - \{(u',v)\}$
$V' \leftarrow V' \cup C - \{x\}$
return (V', E')

3.2.4 Discussion

All of the three graph algorithms presented above are able to extract a tree structure from the link graph. Shortest-path search bears some similarity to breadth-

first search. Both algorithms generate the tree by iteratively expanding the frontier of the tree by selecting a node closest to the root so far. Shortest-path search maintains a min-priority queue Q which is like the set of white vertices in a breadth-first search; just as vertices in Q have their shortest-path estimate, so do black vertices in a breadth-first search have their correct breadth-first distances. However, the shortest-path estimate is based on the real function w, which is more precise than the breadth-first distances which simply count number of edges. In a dense graph like a Website's link graph, it is easy to encounter multiple paths with equal breadth-first distance which is less likely when using shortest-path estimate with a properly designed weight function. Therefore, shortest-path search is more capable of distinguishing aggregation links from short-cut links for generating topic hierarchy from the link graph. Directed minimum spanning tree is different from the other two algorithms in that it tries to minimize the weight of the very last edge on the path from the root to a node whereas breadth-first search and shortest-path search evaluate the whole path based the breadth-first distance and shortest-path estimate.

Notice that both shortest-path search and directed minimum-spanning tree need the edge weight function w as input, which plays an important role in selecting edges in the topic hierarchy. Therefore, the effectiveness of these two algorithms heavily depend the edge weight function. We will give details for designing the weight function in following section.

3.3 Edge Weight Function

Recall that there are two types of hyperlinks between Web pages within a website: aggregation links, which connects topics and topics, and short-cut links, for which there is no hierarchical relationship between the two connected Web pages. Clearly, in topic hierarchy generation, the aggregation links should be selected to form the tree structure. The graph algorithms described in the previous sections apply different criteria to identify the aggregation links. Breadth-first search doesn't use

edge weight and selects the edges which minimize the number of links connecting the root to each vertex. Shortest-path search and directed minimum spanning tree considers the weights of edges. The former selects the edges which form a path from the root to a node with the smallest total weight. The later tries to make the weight on edges into each node as small as possible while maintaining the spanning tree topology. Both algorithms tend to favor light edges over heavy edges. Therefore, a proper edge weight function should assigns smaller weight to aggregation links and large weight to short-cut links. In this section, we describe two approaches to designing a weight function which maps a link (u, v) to a value in the range [0, 1].

3.3.1 Relevance Method

Given a hyperlink (u, v), the relevance method computes the $w(u, v)$ by estimating the relevance of v to u. The assumption is that v is more relevant to u for aggregation links than for short-cut links. For an aggregation link, the two pages u and v either have a **ISA** relationship such as when u is the page "Projects" and v is the page for "Data Stream", or **Part-of** relationship such as when u is "J. Ullman" and v is "J. Ullman's Books". Both **ISA** and **Part-of** mean u and v are semantically related (i.e. relevant), whereas a short-cut link usually connects two loosely related entities such as a professor and his department. Therefore, the relevance of v to u is a useful measure for distinguishing aggregation and short-cut links. We compute the relevance between u and v by comparing their content and path.

3.3.1.1 Content Relevance

The content relevance reflects the similarity between the texts on u and v. To compare the textual content of Web pages, we first preprocess the Web pages by removing all HTML mark-ups. The remaining text is then used to build a vector representation of the content, $[w_{1,k}, w_{2,k}, \ldots w_{t,k}]^T$, where $w_{i,k}$, the weight of each term i in document k, is its $tf*idf$ value, where tf is the frequency of the term i in the k, idf is its inverted document frequency (Yates and Neno 1999). The content relevance is calculated by the cosine similarity between the two document vectors:

$$r_{content}(u,v) = \frac{\sum_{i=1}^{t} w_{i,u} \cdot w_{i,v}}{\sqrt{\sum_{i=1}^{t} w_{i,u}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,v}^2}} \quad (1)$$

3.3.1.2 Path Relevance

The path relevance is based on the URL of u and v. A URL string, such as `www.cs.cmu.edu/people/index.html` consists of a domain name `www.cs.cmu.edu` and a path `/people/index.html`. As we are dealing Web pages of particular Website, the host name portion of their URLs must be the same. The URL's path portion reveals two useful features: One is the location of the folder where the page is stored and the other is the name of the file corresponding to this page. Sometimes a URL doesn't include the file name but ends with `/`, in which case it generally refers to the "index.html" file within the folder.

Folders are commonly used for organizing large number of documents. Related files are usually grouped within the same folder and may be further divided by subfolders. Given the paths of all the pages, we can easily build a directory tree to represent the directory structure of the Website, in which the leaf nodes correspond to individual files and the internal nodes represent folders, such as shown in Figure 3-2.

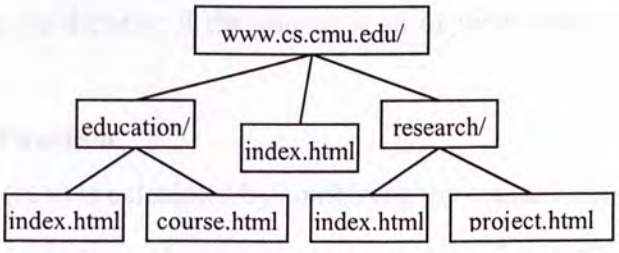

Figure 3-2 Partial directory tree for www.cs.cmu.edu

Given the locations of page s and t in the directory tree, we can easily measure their **directory distance** $dist_{dir}(s, t)$ using the number of edges on the path between the two documents. So, the distance between `www.cs.cmu.edu/education/index.html` and

`www.cs.cmu.edu/research/project.html` is 4 as can be seen from Figure 3-2. Measuring the distance between pairs of pages in a tree structure is straightforward and can be done using the following 2-step procedure: (1) find the lowermost common ancestor of the two leaves (2) count the number of links to go down from this ancestor to each leaf.

The file name for a Web page is a useful feature for telling whether a page is some topic's **entry page**, which generally serves as the table-of-content for a topic and is meant to be the first page to be visited when this user browse into this topic. We identify all pages whose file name contain `index` as a substring as an **explicit entry page**. Given a link (u, v), there is a chance that it is an aggregation link if u is an explicit entry page. We therefore introduce the following function to:

$$\delta_{index}(u) = \begin{cases} -1 & \text{if s is an explicit index page} \\ +1 & \text{otherwise} \end{cases} \tag{2}$$

which is added to $dist_{dir}(u,v)$ to provide path relevance

$$r_{path}(u,v) = dist_{dir}(u,v) + \delta_{index}(u) \tag{3}$$

The function $\delta_{index}(u)$ may be viewed as a refinement to $dist_{dir}(u, v)$. So when the directory distances are equal the explicit entry page feature could further distinguish links by reducing the distance if the source is an explicit entry page and increase it otherwise.

3.3.1.3 Weight Function

Finally, the $w(u, v)$ is calculated by combining the content and path relevance.

$$w(u,v) = \lambda \cdot r_{content}(u,v) + (1-\lambda) \cdot \frac{r_{path}(u,v)}{\max\limits_{\forall u',(u',v)\in E} r_{path}(u',v)} \tag{4}$$

where $0<\lambda<1$ controls the weight on content and path relevance. Note that because r_{path} is not in the range [0,1], normalization is necessary to convert it to the same range as $r_{content}$, which is done through dividing it by the maximum over all links into v.

3.3.2 Machine Learning Method

The relevance method for edge weighting described in the previous section utilizes two numeric features: content relevance and path relevance, and calculate the edge weight by linearly combine the two values. In this section, we introduce the machine learning method for estimate edge weight, which is capable of incorporating a large number of features, both numeric and categorical. The idea is to treat the edge weighting problem as a standard classification problem, in which we attempt to classify each link into one of two classes: aggregation links vs. short-cut links.

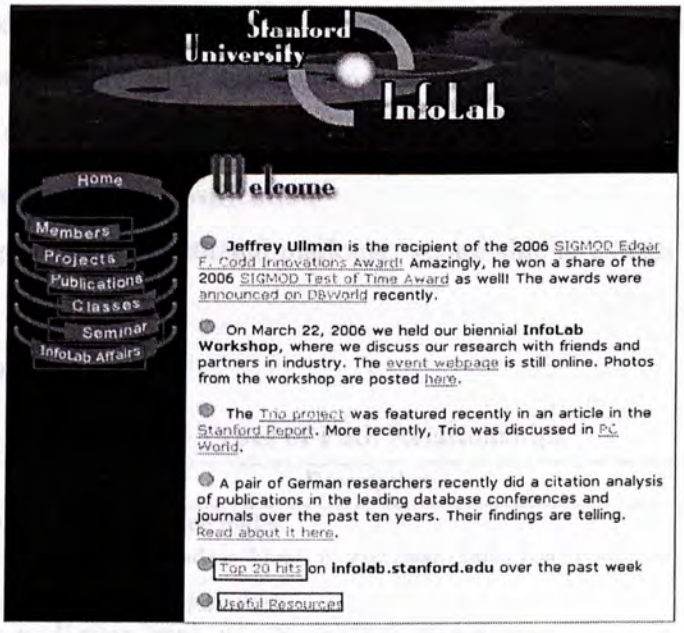

Figure 3-3 A Web page with aggregation being links highlighted

A Web page contains much more information than a normal text document. The HTML markups on a Web page could reveal a variety of information such as the presentation and layout of different elements in the document. These markups are processed by a browser to produce a visual presentation of the Web page to the user. A web designer usually utilizes different visual hints on a Web page to facilitate navigation. Figure 3-3 shows the Stanford Database Group's homepage, on which we

have highlighted the aggregation links. Using only this example, we could notice several patterns about aggregation link and short-cut links:

1. Links in a navigation bar are more likely to be navigation links.
2. Aggregation links is usually associated with an anchor image or short anchor text.
3. Links that occurs in the middle of a sentence are less likely to be aggregation links.

All these patterns are associated with the presentation and layout of the Web page. Through examining a large number of typical Web pages, we identified a set of features which are based the content, path, presentation and layout of the Web pages, that are useful for distinguishing aggregation and short-cut links. The details about these features are described in the next subsection.

3.3.2.1 Features for Link Classification

Using the webpage's content, path and markups, we are able to extract the following features for a link (u, v).

- **Path Relationship**

Table 3-1 Types of Path Relationships

Type	Description
I	u and v are in the same folder. e.g. u : `/ullman/index.html`, v : `/ullman/publication.html`
II	u is in the parent folder of the v e.g. u : `/index.html`, v : `/db_pages/members.html`
III	u is in the v's grandparent or higher folder e.g. u : `/index.html`, v : `/cs104/proj/proj1.html`
IV	u is in some subfolder within v's folder e.g. u : `/cs104/proj/proj1.html`, v : `/cs104/index.html`
V	Otherwise e.g. u : `/cs104/index.html`, v : `/cs202/index.html`

Similar to the directory distance (see section 3.3.1.2), the path relationship feature is based on the locations of u and v in the website's directory. However, we decide to make the path relationship is categorical feature instead of numeric. The

reason for this is that the directory distance is a symmetric measure, so it will be the same no matter if u is in a parent folder of v or u is in a subfolder of v. However, a link is much more likely to be aggregation link when u is in a parent folder of v than in a subfolder of v (Chen et al. 2002). We defined 5 types of relationships for this feature which are described in Table 3-1.

- **Explicit Entry Page**

Each sub tree within a topic hierarchy corresponds to a particular topic of a web site. We define the page sitting at the root of a sub tree as the **entry page** for the topic of that sub tree. It is quite often that the entry page of a particular topic is named `index.html`, e.g. `http://www.db.stanford.edu/~ullman/index.html`. However, an entry page is not necessarily named `index.html`, such as `http://www.db.stanford.edu/db_pages/members.html`, which serves as the entry page for the topic "group members" and uses a meaningful word "members" as its name. We refer to Web pages with name `index.html` as explicit entry pages, as they are easily recognizable by the file name.

For a link (u, v), if u is as explicit entry page, it is more likely to be an aggregation link than if u is not. Hence, we let the explicit entry page to be a categorical feature which can take on one of two values, "yes" and "no", correspond to whether page u is an explicit entry page.

- **Content Relevance**

This is the same as in the relevance based method. We use it here again as a numeric feature for the machine learning method.

- **Inside Navigation Bar Feature**

Navigation bar is a prevailing element in today's web design. A designer uses navigation bars to highlight lists of hyperlinks in a contiguous area on Web pages such as the top, side and bottom. For Websites of enormous size, the use of navigation bars enable users to move around in the websites more efficiently. Very often, links in the navigation bar are pointing to subtopics of the current page.

Therefore, whether a link is located inside a navigation bar can be a useful feature for telling if it is aggregation link.

Our method for identifying the navigation bars on a Web page relies on analyzing the document structure of the HTML file for the Web page. In order to analyze the HTML document, we pass Web pages through an open source HTML parser, openXML (http://www.openxml.org), which corrects the markup, so we do not need to worry about error resilience, and create a Document Object Model (DOM) tree. The Document Object Model (www.w3c.org/DOM) is a standard for creating and manipulating in-memory representations of HTML (and XML) documents. The different elements of a HTML document (e.g. table, table rows, table cells, paragraphs, links, etc) are organized hierarchically in a DOM tree, which reveals useful information about the webpage's layout, such as which element is contained within which element (See Figure 3-4).

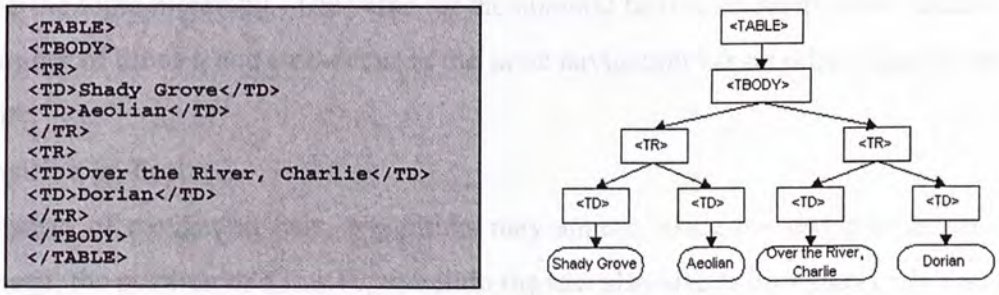

Figure 3-4 A fragment of HTML for simple table and the corresponding graphical DOM tree representation

A navigation bar is supposed to match some sub-tree within the DOM tree of the corresponding documents. To identify such sub trees, we traverse the DOM tree recursively. For each node, we calculate the ratio of the length of linked text (i.e.

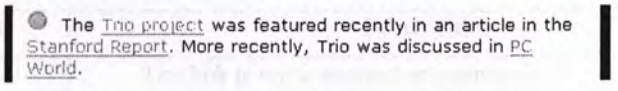

Figure 3-5 Example for Link Text Ratio Computation

archor text for hyperlinks) to the length of all the text under the node. For example, in Figure 3-5, the lengths of all the anchor text sum to 35 and the length of all text is 122, which gives a link text ratio of 35/122=0.287. If this ratio exceeds a threshold, which we set as 0.8, the node is classified as a **link node**. And the root node for a navigation bar is a node which is a link node and whose parent node is not a link node. Upon detecting all the sub trees corresponding to navigation bars, we can decide the value of the inside navigation bar feature as either "yes" or "no" according to whether the link is located within a navigation bar in the DOM structure.

- **Co-reference**

This feature is the counterpart of the inside navigation bar feature. The observation that links within a navigation bar are usually sub topics of the page with the navigation bar also implies that if two pages frequently co-occur in the same navigation bar on other pages, they are more likely to be siblings instead of parent and child in the topic hierarchy. The value for the numeric feature co-reference is equal to the number of times u and v co-occur in the same navigation bar on other pages in the Website.

- **Position in Text**

Instead of navigation bars, hyperlinks may appear inside the text portion on a Web page, the position of a link (u, v) within the text also affects how likely v is a sub topic of u. We first preprocess the text in a Web page by splitting blocks of text into sentences using the Brill Tagger (Brill 1995). The position of a link in the text is then classified as one of the four types listed in Table 3-2 and examples for each type of link position is shown in Figure 3-6.

Table 3-2 Types of Link Positions

Type	Description
I	The link is not a segment in a sentence.
II	The link is at the beginning of a sentence.

III	The link is in the middle of a sentence.
IV	The link is at the end of a sentence.

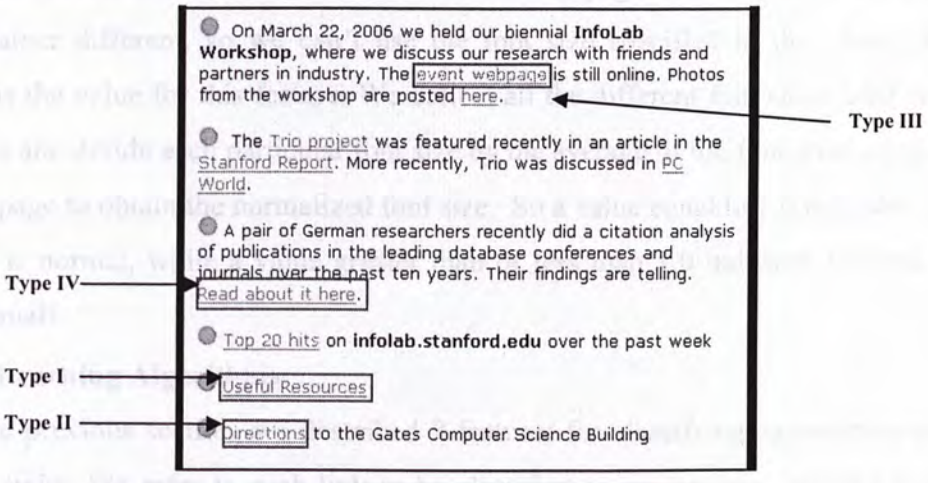

Figure 3-6 Examples of Link Positions

- **Anchor Text Length**

We observed the anchor text for sub topics are usually described by some short phrases, such as "people", "projects", etc., which provide a good abstraction of the nature of the sub topics. In contrast, long anchor text like sentences are usually used highlight certain news or events, such as "IBM scientists discovers new way to explore and control atom-scale magnetism", which are often not sub topics of the page. In the anchor text length feature, we measure the number of non-stop words inside the anchor text (a list of stop words such as "and","to","on", etc. are excluded).

- **Anchor Text Font Size**

HTML allows an author to decorate text using font and colors so as to emphasis certain text and distinguish different types of text. Our assumption is that the font used for anchor text of aggregation links is supposed to emphasize the link, therefore, a larger font size or more distinctive color should be used for the anchor text. However, as the property of a "distinctive" color is quite ambiguous and may involve

image processing techniques, which will hurt efficiency, we focus here on the font size associated the Web page.

Notice that the scales of the font size used on Web pages from different web sites can be rather different, so we can't use the font size specified in the <font> tag directly as the value for this feature. We collect all the different font sizes used on a Web page and divide each particular font size by the average of the font sizes used on the Web page to obtain the normalized font size. So a value equal to 1.0 indicates the font size is normal, while a value greater than or less than 1.0 indicates the font is large or small.

3.3.2.2 Learning Algorithms

In the previous section, we described 8 features for classifying aggregation and short-cut links. We refer to each link to be classified as an *instance*, which can be described by a feature vector $\mathbf{x} = \langle x_1, x_2 \ldots x_8 \rangle$, where x_i is the value of the *i-th* feature for this link. Let c_1 and c_2 denote the two categories: aggregation link and short-cut links. A classifier is a function $f(\mathbf{x})$ that outputs the corresponding category for an instance. In this section, we will describe the specific classification algorithm being used for this classification task. In this work, we've chosen and tested three classifiers, which are Decision Tree, Naïve Bayes and Logistic Regression respectively.

- **Decision Tree**

It is natural and intuitive to classify an instance through a sequence of question, in which the next question asked depends on the answer to the current question. Such a sequence of question can be represented by a decision tree as shown in Figure 3-7. The classification of a particular instance begins at the root node. The questions asked at each node concern a particular feature of the instance, which may be about which case the value of the feature belongs if it's categorical to or what range the value of the feature is in if it's numeric. To determine the category for the instance, we follow

the downward links until a terminal or leaf node is reached, which contains the category decisions.

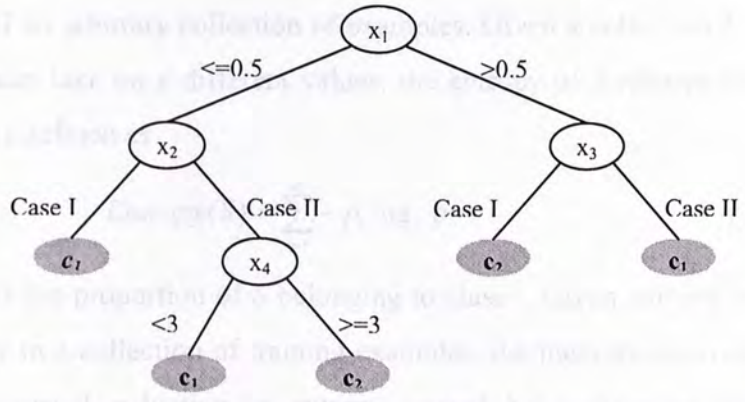

Figure 3-7 Classification in a basic decision tree

Assume that we have a set *D* of labeled training data, clearly any decision tree will progressively split the set of training examples into smaller and smaller subsets. It would be ideal if all the samples in each subset have the same category label. In that case, we would say that each subset was *pure*, and could terminate that portion of the tree. Usually, however, there is a mixture of labels in each subset, and thus for each branch we will have to decide either to stop splitting and accept an imperfect decision, or instead select another feature and grow the tree further. This suggests an obvious recursive tree-growing process: Given the data represented at a node, either declare that node to be a leaf and state what category to assign to it, or find another feature to use to split the data into subsets. This is the generic tree-growing methodology, which is followed by different decision tree learning algorithms.

In this work, we used the C4.5 algorithm for decision tree learning (Quinlan 1993).The central part of the C4.5 algorithm is selecting which feature to test at each node in the tree. We would like to select the attribute that is most useful for classifying example. This is based on a quantitative measure, called *information gain*,

that measures how well a given attribute separates the training examples according to their target classification. In order to define information gain, we first define a measure commonly used in information theory, called *entropy*, which characterizes the impurity of an arbitrary collection of examples. Given a collection S and a target attribute that can take on c different values, the entropy of S relative to this c-wise classification is defined as

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \tag{5}$$

where p_i is the proportion of S belonging to class i. Given entropy as a measure of the impurity in a collection of training examples, the measure information gain is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. More precisely, the *information gain*, $Gain(S, A)$ of an attribute A relative to a collection of examples S, is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{6}$$

where $Values(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v.

Notice that the definition of entropy and information gain above is restricted to attributes that take on a discrete set of values, which is the case for categorical features. To incorporate continuous-valued numeric attributes, the continuous attribute value must be first partitioned into a discrete set of internals. In particular, for an attribute A that is continuously-valued, the algorithm can dynamically create a new Boolean attribute A_c that is true if $A < c$, and false otherwise. The threshold c is selected to produce the greatest information gain (Quinlan 1993).

- **Naïve Bayes**

The Naïve Bayes classifier is a highly practical learning method of the more general Bayesian learning method, which provides a probabilistic approach to classification. It is based on the assumption that the quantities of interest are governed

by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data. One important feature of the probabilistic learning method is that it provides a quantitative approach to weighting the evidence supporting alternative decisions.

The Bayesian approach to classifying new instance is to assign the most probable category, c_{MAP}, given the feature vector $<x_1,x_2...x_n>$ that describe the instance.

$$c_{MAP} = \arg\max_{c_i \in C} P(c_i \mid x_1, x_2 ... x_n) \tag{7}$$

Using Bayes theorem, this expression can be rewritten as

$$\begin{aligned} c_{MAP} &= \arg\max_{c_k \in C} \frac{P(x_1, x_2 ... x_n \mid c_k) P(c_k)}{p(x_1, x_2 ... x_n)} \\ &= \arg\max_{c_k \in C} P(x_1, x_2 .. x_n \mid c_i) P(c_k) \end{aligned} \tag{8}$$

It is easy to estimate each of the $P(c_k)$ by counting the frequency with which each class c_i occur in the training data. However, estimating the different $P(x_1,x_2...x_n|c_k)$ in this fashion is not feasible unless we have a very large set of training data. The problem is that the number of these terms is equal to the number of possible instances times the number of possible classes. Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.

The Naïve Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the class. In other words, the assumption is that given the class of the instance, the probability of observing the conjunction $x_1,x_2...x_n$ is just the product of the probabilities for the individual features:. $P(x_1, x_2 ... x_n \mid c_k) = \prod_j P(x_j \mid c_k)$. Therefore, the classification rule for the Naïve Bayes classifier is:

$$c_{NB} = \arg\max_{c_k \in C} P(c_k) \prod_j P(x_j \mid c_k) \tag{9}$$

where c_{NB} denote the class output by the Naïve Bayes classifier. Notice that in a Naïve Bayes classifier the number of distinct $P(x_j|c_k)$ terms that must be estimated

from the training data is just the number of distinct feature values times the number of distinct target values, a much smaller number than if we were to estimate the $P(x_1,x_2...x_n|c_k)$.

For categorical features, $P(x_j|c_k)$ is usually estimated using the m-estimate defined as follows:

$$P(x_j \mid c_k) = \frac{n_{jk} + mp}{n_k + m} \tag{10}$$

where n_{jk} and n_k are the number of class k instances with j-th equal to x_j and the number of class k instances respectively, p is the prior estimate of the probability, and m is a constant controlling how heavily to weight p relative to the observed data. A typical method for choosing p in the absence of other information is to assume uniform priors; that is if a feature has k possible values we set $p = 1/k$.

For numeric features, $P(x_j|c_k)$ is commonly assumed to be a Gaussian density

$$P(x_j \mid c_k) = N(x_j; \mu_{jk}, \sigma_{jk}) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} e^{\frac{(x-\mu_{jk})^2}{2\sigma_{jk}^2}} \tag{11}$$

where μ_{jk} and σ_{jk} are the mean and standard deviation for the j-th feature of class i instances and can be estimated by the maximum likelihood estimators

$$\begin{aligned} \mu_{jk} &= \frac{1}{\sum_i \delta(c^i = c_k)} \sum_i x_j \delta(c^i = c_k) \\ \sigma_{jk}^2 &= \frac{1}{\sum_i \delta(c^i = c_k)} \sum_i (x_j^i - \mu_{jk})^2 \delta(c^i = c_k) \end{aligned} \tag{12}$$

where the superscript i refers to the ith training example, and $\delta(c^i=c_k)$ is 1 if c^i, the class for the ith training example, is c_k and 0 otherwise. Note the role of δ is to select only those training examples in the relevant class.

- **Logistic Regression**

Logistic Regression (Hastie et al. 2001) attempt to learn $P(c|x_1,x_2...x_n)$ by assuming a parametric form for the distribution, which in the case of two classes is:

$$P(c_0 \mid x_1,x_2...x_n) = \frac{1}{1+\exp(w_0+\sum_{i=1}^{n} w_i x_i)}$$
$$P(c_1 \mid x_1,x_2...x_n) = \frac{\exp(w_0+\sum_{i=1}^{n} w_i x_i)}{1+\exp(w_0+\sum_{i=1}^{n} w_i x_i)} \quad (13)$$

Note that the two probabilities always sum to 1. One highly convenient property of this form is that it leads to a simple linear expression for classification. To classify an instance $X=<x_1,x_2...x_n>$, we generally want to output the value c_k that maximizes $P(c_k|x_1,x_2...x_n)$. In other words, we output c_0 if the following condition holds:

$$\frac{P(c_0 \mid x_1,x_2...x_n)}{P(c_1 \mid x_1,x_2...x_n)} > 1 \quad (14)$$

substituting from equation (13), this becomes

$$1 < \exp(w_0+\sum_{i=1}^{n} w_i x_i) \quad (15)$$

and after taking natural log of both sides we have a linear classification rule that assigns class c_0 if

$$0 < w_0+\sum_{i=1}^{n} w_i x_i \quad (16)$$

The parameter values for Logistics Regression can be determined by maximizing the conditional data likelihood. The conditional data likelihood is the probability of the observed class values in the training data, conditioned on their corresponding feature vectors.

$$W = \arg\max_{W} \prod_{i} P(c^i \mid X^i, W) \quad (17)$$

where $W = <w_0,w_1...w_n>$ is the vector of parameters to be estimated, c^i denotes the class for the ith training example, and X^i denotes the feature vector of the ith

training example. Utilizing the fact that c^i can take only values 0 and 1 and the log likelihood, equation (17) can be rewritten as

$$\begin{aligned} l(W) &= \sum_i c^i \ln P(c^i = 1 \mid X^i, W) + (1 - c^i) \ln P(c^i = 0 \mid X^i, W) \\ &= \sum_i c^i \ln \frac{P(c^i = 1 \mid X^i, W)}{P(c^i = 0 \mid X^i, W)} + \ln P(c^i = 0 \mid X^i, W) \\ &= \sum_i c^i (w_0 + \sum_{j=1}^{n} w_j x_j^i) - \ln(1 + \exp(w_0 + \sum_{j=1}^{n} w_j x_j^i)) \end{aligned} \tag{18}$$

where x_j^i denote the value of feature j for the ith training example.

There is no closed form solution to maximizing $l(W)$. There we need search for the optimal W using gradient ascent, in which we work with the gradient. The partial derivative of $l(W)$ with respect to w_j has the form

$$\frac{\partial l(W)}{\partial w_j} = \sum_i x_j^i (c^i - P(c^i = 1 \mid X^i, W)) \tag{19}$$

Given the formula for the partial derivative of each w_i, we can use standard gradient ascent to optimize the weights W. Beginning with initial weights of 0,we repeatedly update the weights in the direction of the gradient according to

$$w_i \leftarrow w_i + \eta \sum_i x_j^i (c^i - P(c^i = 1 \mid X^i, W)) \tag{20}$$

where η is a small constant (usually 0.01) which determines the step size. Because the conditional log likelihood $l(W)$ is concave function in W, this gradient ascent procedure will converge the a global maximum.

3.3.2.3 Edge Weighting through Classification

In the previous section, we provide details of three classification algorithms, which generate classifiers for predicting an instance's category based on training examples. When building a topic hierarchy, our aim is distinguish aggregation links from short-cut links. Using the three algorithms described earlier, we may build classifiers which can predict whether a link is aggregation link or short-cut links using the features proposed in section 3.3.2.1.

In most applications, a classifier is only used to output the predicted class label, which is symbolic. However, in our case, a numeric value is needed to assign as the weight of the link being classified, so we may obtain a weighted graph that can be used to generate shortest-path trees or minimum directed spanning trees. Also, as both the shortest-path tree model and minimum directed spanning tree model favor edges with smaller weight, a reasonable weighting method should assign larger weight to a short-cut link and smaller weight to an aggregation link.

Let c_0 and c_1 denote the class aggregation link and short-cut link respectively. Given an edge (u,v) with corresponding feature vector X_{uv}, a natural choice is to assign (u,v). So that links more likely to be short-cut links have larger weights. Notice the range of $P(c_1|X_{uv})$ is only from 0 to 1, which would make the link weights tend to be close to each other. Therefore, we use the value $-\log P(c_0|X_{uv})$ as the edge weight. For links that are most probable aggregation links, that is $P(c_0|X_{uv})$ is high, the edge weight would be low because of the negation. Also using the log transformation, the edge weight now range from 0 to ∞. Following this edge weighting scheme, we are able to interpret the shortest-path tree and directed minimum spanning tree using probabilities. In particular, the shortest path tree is indeed the tree that maximizes the probability that all the paths from the root to each node consist all of aggregation links:

$$\begin{aligned} T &= \arg\min_T \sum_{i \in V} \sum_{(u,v) \in P_{iT}} w(u,v) \\ &= \arg\min_T \sum_{i \in V} \sum_{(u,v) \in P_{iT}} -\log(P(c_0 \mid X_{uv})) \\ &= \arg\max_T \prod_{i \in V} \prod_{(u,v) \in P_{iT}} P(c_0 \mid X_{uv}) \end{aligned} \tag{21}$$

where P_{iT} is the path to node i in a candidate tree T.

On the other hand, the directed minimum spanning tree is the tree that maximizes the probability that all edges in the tree correspond to aggregation links.

$$
\begin{aligned}
T &= \arg\min_{T} \sum_{(u,v)\in E_T} w(u,v) \\
&= \arg\min_{T} \sum_{(u,v)\in E_T} P(c_0 \mid X_{uv}) \\
&= \arg\max_{T} \prod_{(u,v)\in E_T} P(c_0 \mid X_{uv})
\end{aligned}
\tag{22}
$$

Notice that the key different between the formulation of shortest-path tree and minimum directed spanning tree is that every edge is counted exactly once in minimum directed spanning tree while an edge may be counted multiple times depending on how many times it is on the tree path to other nodes.

It is easy to compute $P(c_0|X_{uv})$ based on Logistic Regression and Naïve Bayes classifier. In Logistic Regression, a specific parametric form for $P(c_0|X_{uv})$ is assumed. So after training the Logistic Regression model, we may use equation (13) to calculate $P(c_0|X_{uv})$ based on the learned optimal weights W. For Naïve Bayes, the probabilities $P(c_i)$ and $P(x_j|c_i)$ are available. To obtain $P(X_{uv} \mid c_0)P(c_0)$, we may use Bayes rule:

$$
\begin{aligned}
P(c_0 \mid X_{uv}) &= \frac{P(X_{uv} \mid c_0)P(c_0)}{P(X_{uv} \mid c_0)P(c_0) + P(X_{uv} \mid c_1)P(c_1)} \\
&= \frac{\prod_i P(x_i \mid c_0)P(c_0)}{\prod_i P(x_i \mid c_0)P(c_0) + \prod_i P(x_i \mid c_1)P(c_1)}
\end{aligned}
\tag{23}
$$

Unlike, the other two classifiers, the decision tree model doesn't take a probabilistic approach to the classification task. So the probability $P(c_0|X_{uv})$ has to be estimated based on the training data. The idea is to consider the set of training instances that reach each node, using which we may count the number of instances in each class. Suppose there are n_0 out of a total of n instances that are of class c_0 at this node. Let the true probability $P(c_0|X_{uv})$ be q, the n instances can be considered as being generated by a Bernoulli process with parameter q, of which n_0 turn out to be class c_0. Notice that it is inappropriate to use the observed frequency $f = n_0/n$ as the

estimate for q, as it doesn't take into account the size of sample used to estimate q. One standard method is to construct a confidence interval for q and use the upper confidence limit as the estimate. Given a particular confidence level c (the default value used by C4.5 is c=25%), we find confidence limit z such that

$$P\left[\frac{f-q}{\sqrt{q(1-q)/N}} > z\right] = c$$

Here z is the number of standard deviations corresponding to the confidence c for standard normal distribution, which for c = 25% is z = 0.69. Using the confident limit z, we can estimate $P(c_0|X_{uv})$ at the node:

$$P(c_0 \mid X_{uv}) = \frac{f + \frac{z^2}{2n} + z\sqrt{\frac{f}{n} - \frac{f^2}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \tag{24}$$

3.4 Experiments

In section 3.2, we have described three graph algorithms for the constructing topic hierarchies using the Website's link structure: the breadth-first tree model on unweighted graphs, the shortest-path tree and directed minimum spanning tree on weighted graphs. Then in section 3.3, we provided details on the edge weighting in weighted graph models, for which we have developed two approaches: the relevance method and the machine learning method. In this section, we describe the a set of experiments using real web data, which tests the effectiveness of the proposed algorithms.

3.4.1 Data Preparation

We have chosen 5 Websites from different domain for evaluating the algorithms, which are listed in Table 3-3. To download the Web pages from these Websites, we provide the homepages of these Websites as the seed page to a crawler. The crawler

Table 3-3 Websites used for evaluation

URL	Type	Description
www.cs.cmu.edu	Education	School of Computer Science Carnegie Mellon University
www.db.stanford.edu	Education	Database Research Group Stanford University
www.whitehouse.gov	Government	US Government
www.research.ibm.com	Commercial	IBM Research
www.palmsource.com	Commercial	Palm Software and Palm OS

then traverses the Website following breadth-first order. So Web pages linked on the homepage are downloaded first, followed by the Web pages linked these pages and so on. The crawler examines every link it encounters and only follows links to any URL that is in the same domain as the homepage. As the complete Websites may contain enormous number of Web pages, we set the maximum crawling depth to 5. Therefore, only pages that can be reached from the homepage by following no more than 5 links are downloaded. For all the downloaded pages, we change any relative paths used by the hyperlinks to absolute paths and append the default file name "index.html" to any path ending with "/" (e.g. `/~ullman/` to `/~ullman/index.html`).

For each Website, we count the number of pages at different breadth-first levels, where level 1 contains the homepage, level 2 contains the pages linked from the homepage, and so forth. These statistics are shown in Table 3-4. We also counted the average degree (i.e. number of in-links plus number of out-links) of the nodes at each level (see Table 3-5). We may observe the following properties about Website's link structure:

- The number of pages grows very fast as the depth increases. This is the property of any tree structure.
- The degree of nodes tends to decrease rapidly as a node gets further away from the homepage.

Table 3-4 Number of pages at each level

Dataset/Level	1	2	3	4
www.cs.cmu.edu	1	37	246	336
www.db.stanford.edu	1	9	102	769
www.whitehouse.gov	1	55	2424	20928
www.research.ibm.com	1	51	245	1256
www.palmsource.com	1	26	351	384

Table 3-5 Average degree of nodes at each level

Dataset/Level	1	2	3	4
www.cs.cmu.edu	428	292	20	22
www.db.stanford.edu	119	24	15	13
www.whitehouse.gov	2367	1537	130	2
www.research.ibm.com	997	89	28	11
www.palmsource.com	374	250	28	3

To obtain benchmark data on these Websites, a human judge working in the field of the information retrieval is asked to manually produce a partial topic hierarchy for each Website. The judge examines the Web pages in each site following depth first order starting at the homepage. For each page he visits, the judge first extracts the set of links which points to a subtopic of the current page from the set of hyperlinks on the page. He then randomly selects a subset of the identified subtopics which are further explored. Repeating this process for each Website, we obtain a directed tree, that is a sub graph of the complete topic hierarchy. We refer to this tree as a partial topic hierarchy and use it as the benchmark for evaluating a new tree generated by an algorithm.

To evaluate a new tree, we only need to check the locations of the pages in the benchmark. For each page v in the benchmark, let u and w denote the pages which are the parent of v in the benchmark and the new tree respectively, we count it as a *hit* if w matches u, and *miss* otherwise. This allows us to measure the quality of the new

Table 3-6 Sizes of benchmarks for each dataset

Dataset	Benchmark Size
www.cs.cmu.edu	124
www.db.stanford.edu	260
www.whitehouse.gov	179
www.research.ibm.com	441
www.palmsource.com	123

tree using its accuracy which can be easily computed by dividing the number of hits by the number of hits plus number of misses.

3.4.2 Performances of Breadth-first Search

The breadth-first search algorithm (Section 3.2.1) is the simplest among the described approaches. It involves no adjustable parameters. We use it as the baseline to compare with more complicated algorithms such as shortest-path search and directed minimum spanning tree. The accuracies of the topic hierarchy generated by breadth-first search for each Website are shown in Table 3-7.

Table 3-7 Performance of breadth-first search

Dataset	Accuracy
www.db.stanford.edu	74.3%
www.cs.cmu.edu	77.2%
www.research.ibm.com	71.9%
www.whitehouse.gov	79.4%
www.palmsource.com	71.3%

3.4.3 Performances of Shortest-path Search

Unlike the breadth-first search, the shortest-path search algorithm (Section 3.2.2) works on weighted directed graph. In section 3.3, we discussed two approached to estimate the edge weight: relevance method and machine learning method. We used both methods to construct weighted graphs and measured the performance of shortest-path search based on the different graphs.

Table 3-8 Optimal performance of shortest-path search

Dataset	Highest Accuracy	Optimal λ
www.db.stanford.edu	94.6%	0
www.cs.cmu.edu	94.9%	0.2
www.research.ibm.com	78.9%	0.7
www.whitehouse.gov	82.4%	0.2
www.palmsource.com	88.9%	0.9

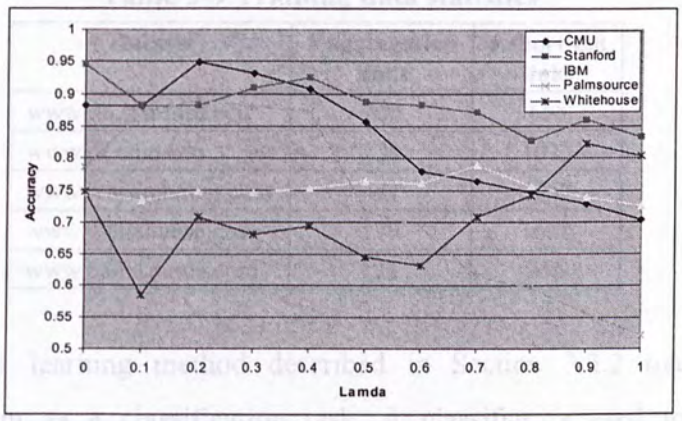

Figure 3-8: Accuracy of topic-hierarchy against change of λ

3.4.3.1 Relevance Method

The relevance method (Section 3.3.2.1) determines the edge weight by combining the path relevance and content relevance using equation (4). The parameter λ controls the relative importance of content relevance and path relevance in the edge weight computation. By varying the value of λ from 0 to 1, we may obtain weighting scheme that is based on path relevance only or content relevance only, or a combination of both.

We plotted the accuracies of topic hierarchy against varying λ values, from which we have made the following observations:

- The effect of λ on the accuracy of the topic hierarchy is quite significant, except on the IBM dataset.

- The accuracy tends to decrease as λ gets closer to 1, except for the Whitehouse dataset. This indicates that using the content relevancy alone can be unreliable.
- The optimal setting for λ seemed to be site dependent. However, a λ value smaller than 0.5 appeared to be more likely to lead to good performance. The optimal performance on each dataset and corresponding λ value are listed in Table 3-8.

3.4.3.2 Machine Learning Method

Table 3-9 Training data statistics

Dataset	# aggregation links	# short-cut links
www.db.stanford.edu	260	658
www.cs.cmu.edu	124	1023
www.research.ibm.com	441	2057
www.whitehouse.gov	179	1601
www.palmsource.com	123	756

The machine learning method described in Section 3.3.2 treats the edge weighting problem as a classification task. A classifier is used to predict the confidence of a link being aggregation link based on a set of automatically derived feature.

To apply the machine learning method for edge weighting, we must first obtain the link classifiers by running the learning algorithms on some training data. For the classification of hyperlinks, the training data should consists of instances of both aggregation links and short-cut links. The construction of the training data is based on the manually collected benchmark data for evaluating topic hierarchies. The benchmark data itself consists of aggregation links only. To get the shortcut-links, we apply the following rule: if a link (u,v) is included in the benchmark (i.e. is an aggregation link), then the other links pointing to v are short-cut links. For each Website, we use the benchmark data to get the aggregation links and find the corresponding short-cut links based on the Website's link graph. The numbers of

aggregation and short-cut links in the training data generated from each Website are shown in Table 3-9.

When evaluating the machine learning method, our objective is to find a classifier that is capable of classifying links in some unseen Website instead of the Websites it was trained on. Therefore, we use the leave-one-site-out scheme to conduct the evaluation as follows: for each of the 5 Websites, we train a classifier using the data of the other 4 sites and then use the classifier to predict edge weight for the remaining 1 site, which guarantees the data used for training and testing are independent. Therefore, the performance measures achieved could fairly indicate how well the method could generalize to other new Websites. For each dataset, we trained a decision tree, a naïve bayes and a logistic regression classifier to estimate edge weights for the directed graph and measured the performance of shortest-path search on the different graphs weighted using different classifiers. The results are shown in Table 3-10. As can be seen, the decision tree classifier produced the best results on all but the Stanford dataset, while the performances of naïve bayes and logistics regression are close to each other and significantly lower than that of decision tree.

Table 3-10 Performance of shortest-path search with different classifiers for estimate edge weight

Dataset	Decision Tree	Naïve Bayes	Logistic Regression
www.db.stanford.edu	90.9%	*92.5%	89.8%
www.cs.cmu.edu	*97.5%	91.5%	81.4%
www.research.ibm.com	*79.4%	75.3%	76.6%
www.whitehouse.gov	*87.8%	81.0%	81.6%
www.palmsource.com	*88.9%	69.2%	70.1%

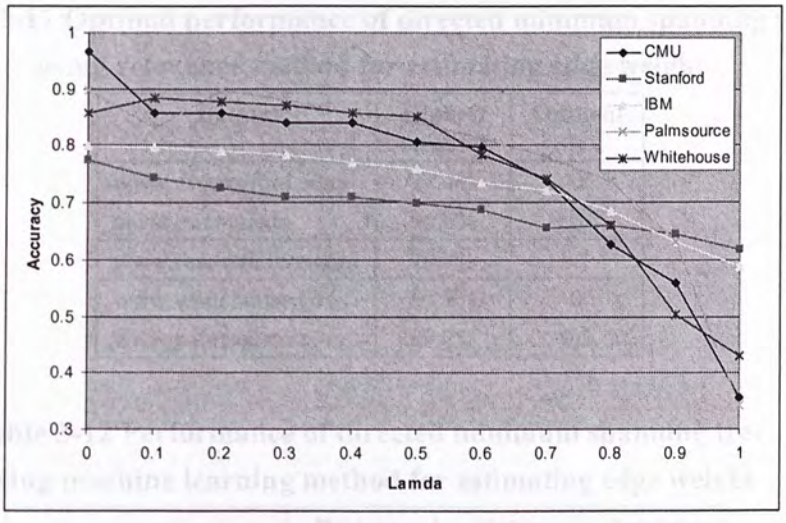

Figure 3-9 Performance of directed minimum spanning tree against λ

3.4.4 Performances of Directed Minimum Spanning Tree

The evaluation of directed minimum spanning tree is very similar to the previous section. Both methods of edge weighting, relevance method and machine learning method have been tested.

For relevance method, the accuracy of generated topic hierarchy against changing λ is plotted in Figure 3-9. In contrast to the figure for shortest-path search, it clearly shows that the accuracy consistently drops as λ increases. Also the optimal results for all the datasets are achieved at very small values of λ as can be seen in Table 3-11. These results indicate that the directed minimum spanning tree model works best when the edge weight is dominated by path relevance.

For machine learning method, the training and testing procedures are exactly the same as used for evaluating shortest-path search. The performances of directed minimum spanning tree algorithm are shown in Table 3-12. Again, decision tree led to the best results on 4 out of 5 Websites, and was only slightly outperformed by naïve bayes on the Stanford dataset.

Table 3-11 Optimal performance of directed minimum spanning tree using relevance method for estimating edge weight

Dataset	Highest Accuracy	Optimal λ
www.db.stanford.edu	77.4%	0
www.cs.cmu.edu	96.6%	0
www.research.ibm.com	79.7%	0.1
www.whitehouse.gov	85.7%	0
www.palmsource.com	89.7%	0.2

Table 3-12 Performance of directed minimum spanning tree using machine learning method for estimating edge weight

Dataset	Decision Tree	Naïve Bayes	Logistic Regression
www.db.stanford.edu	88.2%	*88.6%	85.5%
www.cs.cmu.edu	*98.3%	92.4%	89.0%
www.research.ibm.com	*85.9%	80.2%	80.2%
www.whitehouse.gov	*94.6%	89.1%	79.6%
www.palmsource.com	*92.3%	91.5%	91.5%

3.4.5 Comparison of Different Algorithms

In the previous three subsections, we have presented the experimental results for each individual algorithm for generating topic hierarchy, which are breadth-first search, shortest-path search and directed minimum spanning tree. For shortest-path search and directed minimum spanning tree, the relevance method and machine learning method for estimating the edge weights are evaluated respectively. The common measure of performance used in all the above evaluation is the accuracy of the generated topic hierarchy. In this section, we would summarize and compare the performance across all the proposed algorithms and answer the question of what would be the best approach to generating topic hierarchy. The performances of these algorithms are summarized in Table 3-13. The 9 algorithms being compared include breadth-first search (bfs); shortest-path search using relevance method for edge weighting (sps-rel); directed minimum spanning tree using relevance method for edge

weighting (dmst-rel); shortest-path search, using decision tree for edge weighting (sps-dt); directed minimum spanning tree using decision tree for edge weighting (dmst-dt); shortest-path search, using naïve bayes for edge weighting (sps-nb); directed minimum spanning tree using naïve bayes for edge weighting (dmst-nb); shortest-path search, using logistic regression for edge weighting (sps-lg); directed minimum spanning tree using logistic regression for edge weighting (dmst-lg).

It can be seen that shortest-path search and directed minimum spanning tree outperforms breadth-first search significantly. Although the effectiveness of shortest-path search and directed minimum spanning tree still depend on the method for edge weighting, their different variations constantly outperforms breadth-first search. This confirms that a weighted-graph model is more suitable for representing the link structure of a Website.

For 4 out of the 5 datasets, the most accurate topic hierarchy was generated by the directed minimum spanning tree. Also given the same edge weighting method, the performance of directed minimum spanning tree was consistently better than shortest path search. Therefore the directed minimum spanning tree is the more appropriate model for modeling topic hierarchies of Website.

Another important question to answer is what the most effective way of estimating edge weights is. For the machine learning method, the best classifier for this task is clearly the decision tree. Both shortest path search and directed minimum spanning tree generate more accurate topic hierarchies when the edges are weighted by decision tree than by the other two classifiers. The optimal performances on 4 of the 5 Websites were also produced on graphs weighted by the decision tree. However, when using the other classifiers to estimate edge weights, the resulted topic hierarchy can be worse than that generated based on relevance method. This indicates the importance of picking the suitable classifier when applying the machine learning method.

Table 3-13 Performance comparison of different algorithms for topic hierarchy generation

Dataset/ Algorithm	bfs	sps-rel	sps-dt	sps-nb	sps-lg	dmst-rel	dmst-dt	dmst-nb	dmst-lg
Stanford	74.3%	*94.6%	90.9%	92.5%	89.8%	77.4%	88.2%	88.6%	85.5%
CMU	77.2%	94.9%	97.5%	91.5%	81.4%	96.6%	*98.3%	92.4%	89.0%
IBM	71.9%	78.9%	79.4%	75.3%	76.6%	79.7%	*85.9%	80.2%	80.2%
Whitehouse	79.4%	82.4%	87.8%	81.0%	81.6%	85.7%	*94.6%	89.1%	79.6%
Palmsource	71.3%	88.9%	88.9%	69.2%	70.1%	89.7%	*92.3%	91.5%	91.5%

Chapter 4 Website Summarization Through Keyphrase Extraction

4.1 Introduction

As both the size and diversity of today's Websites grows rapidly, it is becoming more and more difficult for a user to sift through a Website and find the pages containing useful information. This is known as the "information overload" problem. Some Websites provide a sitemap which lists the hyperlinks to the major topics of a Website in a hierarchical tree where each topic is usually described by a short phrase that summarizes the nature of the topic. A sitemap allows a user to grasp the overall content organization of a Website and quickly select one or more relevant topics for further exploration.

There are two limitations for current sitemaps. First is the small size. Most sitemaps need to be manually constructed. It is therefore infeasible to build a complete sitemap for sites with hundreds or even thousands of Web pages. Most sitemaps only cover a limited number of pages, which are typically those pages that correspond to the first two or three levels of the Website's topic hierarchy. The second limitation is that the short phrase description is not informative enough. The phrases used to describe topics are usually very general terms such as "project" or "people". Describing a topic using only one or two word is often of very limited value to a user who may be looking for very specific information and is not able to get any hint from so concise descriptions.

The topic hierarchy generation algorithms described in Chapter 3 provides a solution to the first problem by automatically extracting a tree from the Website's link graph, which reflect the topic-subtopic relations between Web pages. In this chapter,

we try to address the second problem. Our objective is to automatically generate a summary of each topic that provides an overview of the content of each topic. In particular, the summaries for each topic consist of a set of keyphrases that describes the important subjects within this topic. In this work, a keyphrase can be either keyword or key-term. A keyword is a single word and key term is a multi-word expression. The use of keyphrases for summarization has been used extensively for other type of text documents such as scientific papers, which include a keyword or keyphrase field in which the author need to specify a list of phrases that characterizes their publication. The keywords enable the reader to quickly determine whether the given article is in the reader's interest. The HTML markup language also allows the creator of a webpage to specify keywords in the metadata. The goal is to make web search more precise. Documents which have one or more of given query terms in its keyword list are generally more relevant and should be ranked higher in a search engine's result list.

The use of keyphrases as summaries of the topics of Websites has several benefits. First, keyphrases are short and therefore very suitable for display on sitemaps, which displays a large number of links instead of just a few. Secondly, keyphrases can be easily adapted to produce summaries of variable lengths. Given a ranked list of keyphrases, a user may obtain summaries of different length and by simply picking the number of keyphrases in the summary, whereas for other forms of summary such as sentence based summaries, including or excluding one sentence can significantly affect the length. Finally, keywords and keyphrases can be interpreted individually and independently of each other. This is particularly useful for describing large collections of documents such as Websites, whose content is very diverse.

In this chapter, we approach the problem of summarizing Websites through keyword extraction, which automatically select important, topical phrases from the within the body of the Web pages in a Website. Automatic keyphrase extraction is a special case of the more general problem of automatic keyphrase generation, in which

the generated phrases do not necessarily appear in the body of the given documents. We have developed algorithms for performing keyword extraction based on the Website's topic hierarchy. Each Web page is summarized not only based on its own content but also its relationships with other Web pages as being indicated by the topic hierarchy. The automatically extracted keyphrases are evaluated by human users. The performance of the proposed algorithms is also compared with some other existing approaches.

A Website represents a large collection of Web pages. To summarize a Website, our goal is to automatically generate keyphrases for each Web page with the Website. Notice that unlike any ordinary corpus in which the individual documents are independent, the Web pages of a Website are inter-relations as determined by the content organization of the Website. The topic hierarchy for a Website models its content organization by capturing topic/sub-topic relationships between all the Web pages. This kind of hierarchical relationship is very important because for a page that is the root of some sub-tree in the topic hierarchy, the pages below it all represent its subtopics and extends the content on the root page itself. Therefore, the position of a Web page inside the topic hierarchy needs to be considered by a keyphrase extraction algorithm.

4.2 Background

Keyphrase extraction is one of the approaches to the more general problem of text summarization. The goal of automatic summarization is to extract from a textual document the most important content and present it to the user in a condensed form. Summarization techniques are divided into two classes: *abstraction vs. extraction*. Extraction techniques produce a summary consisting entirely of material copied from the input the text. In contrast, abstraction produces a summary at least some of whose material is not present in the input. Typically, an abstract contains some degree of paraphrase of the input content. Abstraction techniques usually require deeper

linguistic analysis such as syntax, semantics or even discourse while extraction method typically won't go beyond statistical analysis at the lexical level. Summarization techniques can also be classified as either single-document summarization or multi-document summarization depending whether the goal is to generate summary for a single document or a collection of documents.

The classic work of Edmunson (Edmunson 1969) defined the framework for much of the work on sentence extraction. In this framework, sentence extraction are preformed based on four features, which were *cue words*, *title words*, *key words*, and *sentence location*. The first three features were word-level features chosen after excluding a list of stop words. Cue words were extracted from a training corpus, whereas the other features were derived from the document to be summarized.

Title words were words from the title, subtitles, and heading found in sentences of the document. The assumption here is that authors will tend to use informative titles.

Cue words are divided into bonus words and stigma words. The idea is that words like "significant", "impossible", "hardly", etc., affect the probable extract-worthiness of a sentence.

To extract key words, words in the document were sorted in descending order of frequency. Thus, unlike cue words, these words were document specific. The most frequent words were selected as key words.

The location feature of a sentence was based on two methods. First, sentences that occurred under section or sub section headings were assigned positive weight. Second, sentences were assigned positive weights if they occurred in the first and last paragraphs, or if they were the first or last sentence in a paragraph.

The overall method of scoring sentences for extraction was based a linear combination of the weights of the four features. Let W be the overall score of a sentence s, C=Cue word, K=Key word, L=Location, T=Title:

$$W(s) = \alpha C(s) + \beta K(s) + \gamma L(s) + \delta T(s) \tag{4.1}$$

The parameters α, β, γ, δ can be tuned by feedback from comparison against manually created training extracts. In evaluation on test data, he found that key words were poorer than the other three features, and that the combination of cue-title-location was the best, with location being the best individual feature and key words alone the worst.

More recent works exploit the discourse structure of text for the purpose of summarization, especially the notion of text cohesion (Halliday and Hasan 1996). Text cohesion involves relations between words, word senses, or referring expressions, which determine how tightly connected the text is. Cohesion is expressed in terms of links in text, called ties, which express semantic relationships. Text cohesion includes 'grammtical cohesion', involving linguistic relations such as anaphora, ellipsis, and conjunctions; and 'lexical cohesion', which involves relations such as reiteration, synonymy, and hypernymy (eg., dog is-a-kind-of animal).

The most natural way of representing cohesion in text for computational purposes is to represent a text as a graph. Here the nodes are text elements, and the edges are links between text elements. The text elements may be words, sentences or paragraphs depending on the application. The relations involved are cohesions described above. The basic idea of representing texts in terms of graphs is that the topology of the graph will reveal something interesting about the salience of information in the text. In particular, a common *graph connectivity assumption* is that nodes which are connected to lots of other nodes are likely to carry salient information. (Note that this assumption is actually a more structured analog of the key word. An important idea related to cohesion and topical segments is the notion of a *lexical chain*. A lexical chain can be characterized as a sequence of related words spanning a topical unit of the text (Morris and Hirst 1991). Lexical chain was

successfully used in summarization (Barzilay and Elhadad 1999). They examine relationships of repetition, synonymy, hypernymy, antonymy, and holonymy which are derived from the WordNet thesaurus[1]. By grouping words together into lexical chains, they suggest a reader might get a better identification of the topic rather than simply picking the most frequent words in the text. In some cases, they argue, a chain of low-frequency words representing the same salient concept may be more indicative of a topic than high-frequency words, because of semantic relationships between the words. The basic problem in computing chains using WordNet is the high degree of polysemy of English words, resulting in many possible chains being formed. The authors choose the best chain for a text based on the number o weight of different relations in the chain. Chains are scored such that a 'strong' chain will include many occurrences of members of the chain and will be homogeneous. A set of heuristics are then applied to build summaries based on selected strong lexical chains.

Another class of approaches treated the sentence extraction as a classification problem and are based on machine learning methods. A standard reference work for most machine learning based summarization is (Kupiec et al. 1995). They used a Bayesian classifier which takes each test sentence and computes a probability that it should be included in a summary. The corpus they used consisted of 188 full text/summary pairs, drawn from 21 different collections of scientific articles. The features used in these experiments were sentence length, presence of cue phrases (eg.: "in conclusion", "in summary", etc.), whether a sentence's location was paragraph-initial, paragraph-medial, or paragraph-final, presence of thematic terms (i.e. high-frequency content words), and presence of proper names (identified based on case).

[1] http://wordnet.princeton.edu/

$$P(s \in E \mid F_1,.., F_n) = \frac{\prod_{i=1}^{n} P(F_i \mid s \in E)P(s \in E)}{\prod_{i=1}^{n} P(F_i)} \quad (4.2)$$

The Bayesian classifier is shown in the equation above. The left-hand-side gives the probability that sentence s from the source is included in extract E, given the sentence's n features $F_1,\ldots,F_n$. $P(s \in E)$ is the probability that a source sentence s is included in the extract E. The latter value is a constant, given by the compression rate. The $P(F_i \mid s \in E)$ is probability of feature F_i occurring in an extract sentence, and $P(F_i)$ is the probability of feature F_i occurring in the corpus of source sentences.

The overall performance of the classifier on new test documents was 42% recall of sentences in the summaries. As the summaries were lengthened, performance improved, achieving 84% sentence recall at 25% of the full text length. As with Edmundson's paper, they also found that location was the best individual feature.

Mutli-Document Summarization(MDS) is, by definition, the extension of single-document summarization to collections of related documents. Due to the explosive growth of WWW, user frequently needs to face vast amount of related information. Being able to see at a glance what a collection is about is therefore quite desirable. Further, since there is a lot of similar information recycled or repeated across different information sources, there is a need for summarization methods that can remove *redundant* information, identifies what is *common* in a variety of related documents and how documents *differ* from one another on some particular subject.

One approach to MDS which addresses redundancy is that of "Maximal Marginal Relevance' (MMR) (Carbonell et al. 1997). In this approach, multiple texts (which can be passages or sentences) are first ranked in terms of relevance to a query. Once the user has scanned some of these, the remaining texts can then be reranked so as to maximize their dissimilarity from the ones already seen. This approach therefore offers a ranking parameter that allows a user to slide between relevance to query and diversity from texts seen so far.

$$MMR(Q,R,S) = \arg\max_{D_i \in R \setminus S}(\lambda sim_1(D_i,Q) - (1-\lambda)\max_{D_j \in R} sim_2(D_i,D_j)) \quad (4.3)$$

In the equation above, $\boldsymbol{Q}$ is the query, $\boldsymbol{R}$ is the retrieved set of documents, $\boldsymbol{S}$ is the scanned subset of $\boldsymbol{R}$, and $\boldsymbol{R \setminus S}$ is what's left of $\boldsymbol{R}$ once $\boldsymbol{S}$ is removed. It allows for different weighting schemes for document/document and document/query comparisons. The important feature of MMR approach is that the redundancy of the summary is directly controlled by a single parameter λ. If set for maximal diversity (λ=0), the texts retrieved will be as dissimilar as possible, and therefore have as little overlap in vocabulary as possible. However, the issue of what value to set λ remains. In their system's interface, users can vary λ to probe for most satisfactory summaries.

Another approach utilized techniques in document clustering, especially topic detection and tracking, to identify salient information in clusters related documents (Radev et. al. 2004). A key feature of this technique is its use of cluster centroids, which consists of words which are central not only to one article in a cluster, but to *all* the articles. The use of centroid can be viewed as an extension to the key word feature based on measures such as TFIDF in single document summarization. This approach is also designed for query-independent and generic summaries instead of query-based summary as the MMR approach discussed earlier. To generate the centroid, they compute the average TF of each term across the whole cluster, which is then multiplied by the term's IDF value obtained from some training corpus to produce the term's centroid value.

Three features were used to compute a sentence's salience: Centroid value, Position value and Title overlap. The centroid value $\boldsymbol{C_i}$ for sentence $\boldsymbol{S_i}$ is computed as the sum of the centroid values $\boldsymbol{C_{w,i}}$ of all words in the sentence.

$$\boldsymbol{C_i} = \sum_{w} \boldsymbol{C_{w,i}} \quad (4.4)$$

The positional value is computed as follows:

$$\boldsymbol{P_i} = \frac{(\boldsymbol{n} - \boldsymbol{i} + 1)}{\boldsymbol{n}} * \boldsymbol{C_{\max}} \quad (4.5)$$

where C_{max} is the highest centroid value of all sentences within the document. So the first sentence gets the highest score, and the further a sentence is from the first, the lowers its positional score. The title overlap score is computed as the vocabulary overlap between a sentence and the title:

$$T_i = \frac{|T \cap S_i|}{|T \cup S_i|} \tag{4.6}$$

The final score of a sentence is a combination of the three scores.

$$SCORE(S_i) = w_c C_i + w_p P_i + w_t T_i \tag{4.7}$$

while the authors suggested the possibility of learning the weights in the combination, they did not implement such procedure and only used equal weights on each feature in their test runs. An initial ranking of sentences could be obtained using the scores computed as above. However as the redundancy problem was not taken into consideration during the above ranking process, a reranking of sentences was also performed based on how much a sentence overlaps with sentences ranked higher, which is similar to the idea of MMR.

The above methods all consider a sentence as the basic unit of information and extract complete sentences to build summaries. In keyphrase extraction, a set of phrases instead of sentences are extracted to form the summary. There are two fundamentally different approaches to the problem of automatically generating keyphrases for documents: *keyphrase assignment* and *keyphrase extraction*.

Keyphrase assignment seeks to select the phrases from a predefined vocabulary. It requires a set of training documents with keyphrases already assigned. A classifier is then built for each phrase in the vocabulary. A new document is processed by each classifier, and assigned the keyphrase of any model that classifies it positively (Dumais et. al. 1998).

Keyphrase extraction, the approach used here, doesn't rely on a predefined vocabulary, but instead chooses keyphrases from the text itself. When authors assign keyphrases without predefined vocabulary, typically 70% to 90% of their keyphrases

appear somewhere in the body of their documents (Turney 1999). Keyphrase extraction algorithms employ lexical and information retrieval techniques to extract phrases from the document text that are likely to characterize the documents. Krulwich and Burkley (1996) user heuristics to extract keyphrases. The heuristics are based on syntactic clues, such as the use of italics, the presence of phrases used in section headers, and the user of acronyms. Their motivation is to produce phrases for use as features when automatically classifying documents. Their algorithm tends to produce a relatively large list of phrases, with low precision. Munoz (1996) uses an unsupervised learning algorithm to discover two-word keyphrases, which is based on Adaptive Resonance Theory neural networks. The precision of the algorithm is low. Also, it is not applicable to one-word or more-than-two-word keyphrases. Steiner and Belew (1993) use the mutual information statistics to discover two-word keyphrases. This approach has the same limitations as Munoz (1996), when considered as a keyphrase extraction algorithm: it produces low precision list of two-word phrases.

Another class of algorithms treat keyphrase extraction as a supervised learning problem. These approaches require a corpus of documents with corresponding keyphrases for training. The GenEx keyphrase extraction system consists of a set of parameterized heuristic rules that are tuned to the training corpus by a genetic algorithm (Turney 1999). The Kea system (Whitten et al 1999) uses a Naïve Bayes learning instead of genetic algorithm to induce a probabilistic model from the training corpus. Both algorithms were tested in the scientific literature domain where there exists documents have manually assigned keyphrases. However, to generalize these approaches to other domains, one must obtain a set of documents with keyphrases assigned.

All the above works aim to summarize or extract keyphrase from normal text documents. It is a much greater challenge to handle Web pages, because Web pages differ from traditional text documents in both structure and content. Instead of

coherent text with a well defined discourse structure, Web pages often have diverse contents and are divided into multiple independent blocks (Berger and Mittal 2000).

Systems for Web page summarization have been either *context-based* or *content-based*. Context based systems (Amitay and Paris 2000) relies on the hypertext structure and the way information is described using it. Instead of analyzing the Web page itself, they collects the context of the document by sending queries of the "link:URL" to search engines. Text units containing the link to the target Web page are then extracted from the Web pages in the context. Finally, an automatic filter is used to select the best descriptions for the target Web page based on a set of features include length, punctuation, use of personal nouns, use of acronyms, use of terms expressing opinions (e.g., best, comprehensive), use of terms indicating content (e.g., about), position of verbs (beginning or end), text beginning with capital letter, and term repetition ratio. During the user evaluation, they found that on average users prefer the summaries generated by the system to the text snippets provided by search engines.

Content-based Web page summarization derives from traditional text summarization (Buyukkoten et al. 2000). To cope with the content diversity and lack of discourse structure of Web pages, the proposed a technique for partitioning a Web page into *Semantic Text Units* (STUs), which are page fragments such as paragraphs, lists, et al.. The STUs were identified based on certain structural tags such as table, paragraph, etc. For each individual STU, they used the traditional TFIDF measure to identify key words first. The sentences' salience scores were computed based on the weights of constituent words. They conducted user studies on the effectiveness of this approach for Web page browsing on PDAs and their results showed that the keyword/summary method, which shows only the keywords in the first state and the most significant sentences in the second state, was the most effective in terms of both informativeness and bandwidth requirements.

4.3 Keyphrase Extraction

4.3.1 Candidate Phrases Idenfication

For each Web page, our algorithm extracts candidate phrases in three steps. It first strips all the HTML code on the Web page to obtain only the texts, then identifies, and finally stems and case-folds the phrases.

- ***Input Cleaning***

A Web page usually contains multiple blocks of texts, whose boundaries are usually marked by tables or paragraphs. We therefore split a Web page into multiple text blocks using a list of tags such <tr>, <td> and <p>. After removing all the HTML tags, each text block is split into tokens (sequences of letters, digits and internal periods), and then modified as follows:

- Punctuation marks, brackets and number are replaced by phrase boundaries;
- Apostrophe are removed;
- Hyphenated words are split in two;
- Remaining non-token characters are deleted, as are any tokens that do not contain letters.

The result of processing is a set of lines, each a sequence of tokens containing at least one letter. Acronyms containing periods, like "C4.5", are retained as single tokens.

- ***Phrase Identification***

We use the following rules to identify candidate phrases, which were found to be both simple and effective:

1. Candidate phrases are limited to a maximum length (usually three words).
2. Candidate can't begin or end with a stopword. We used a stopword list containing 425 words in nine syntactic classes (conjunctions, articles, particles, prepositions, pronouns, anomalous verbs, adjectives and adverbs).

All contiguous sequences of words in each input line are tested using the rules above, yielding a set of candidate phrases. For example, a line that reads *the*

programming by demonstration method will generate *programming*, *demonstration*, *method*, *demonstration method*, *programming by demonstration* as candidate phrases, because *by* and *the* are on the stopword list.

- ***Case-folding and Stemming***

The final step in determining candidate phrases is to case-fold all words and stem them using the Porter stemmer (Porter 1980), which discards any suffix of a word. So, for example, the phrase *text summarization* becomes *text summariz*.

Stemming and case-folding allow us to treat different variations on a phrase as the same thing. For example, *text document* and *text documents* are essentially the same, but without stemming they would have to be treated as different phrases.

We also retain the unstemmed words for each phrase for presentation to the user in case the stemmed phrase doesn't turn out to be correct words. When several versions are available, the most frequent one is chosen.

4.3.2 Feature Calculation without Topic Hierarchy

Two features are calculated for each candidate phrase on the Web pages. They are: thematic feature, a numeric measure of a phrase's significance based on its frequency on the Web page and within the entire Web site; and the presentation feature, which measures a phrase's importance on the Web page based on its visual appearance as determined by the markups.

- ***Thematic Feature:***

The thematic feature is first identified by Luhn (1958). Edmunson (1969) proposed to assign thematic weight to keywords based on its *term frequency*, (i.e. its frequency in a document). However, the term frequency alone is not a very useful measure as it doesn't consider a term's frequency in general use. A term's general usage is reflected by document frequency – the number of documents containing the phrase in some large corpus. The document frequency indicates how common a phrase is (and rarer phrases are more likely to be keyphrases). For a Website, we

generate a document frequency file which stores each phrase and a count of the number of Web pages containing it.

The thematic feature for a phrase t in a web page d is equal to:

$$w_{thematic} = \frac{freq(t,d)}{size(d)} \log \frac{df(t)}{N} \text{ where}$$

1. $freq(t,d)$ is the frequency of t in d;
2. $size(d)$ is the number of words in d;
3. $df(t)$ is number of Web pages containing t in the Website;
4. N is the total number of Web pages in the Website.

- ***Presentation Feature:***

As we discussed in the previous section, Web pages are quite different from traditional text documents. The existence of *anchor text* and *special text* contributes much to the difference. Anchor text is the text of hyper links. It was found that anchor text often provides more accurate descriptions of Web pages than the pages themselves (Brin and Page 1998). As many of the links are on a Web page are pointing its sub topics, the anchor texts thus serve as descriptions of those subtopics. Special texts include title, headings, bold and italicized text. The special texts are usually used by the author to highlight or emphasize certain information on the Web page. Our assumption is that both anchor text and special text represent important information on a Web page .

The value of the presentation feature is equal to

$$w_{presentation} = \frac{n_a(t,d) + n_s(t,d)}{N_a(d) + N_s(d)} \text{ where}$$

1. $n_a(t,d)$ and $n_s(t,d)$ are the number of times d appear as anchor text and special text on d respectively;
2. $N_a(d)$ and $N_s(d)$ are the total number of words inside the anchor texts and special texts on d respectively.

4.3.3 Feature Calculation with Topic Hierarchy

An important difference between a Web site and an ordinary collection of documents is that the different Web pages are related. All of the traditional methods for text summarization and keyphrase extraction handle each individual document without considering its related documents as the documents are all assumed to be independent. However, such assumptions are not valid for Websites. As we have discussed in the earlier chapter, a Website's content is often organized into a hierarchy of topics. A Web page representing a broad topic such as "Prof. J. Ullman" becomes the root of a sub tree of the topic hierarchy and all the Web pages in the sub tree all represent its sub topics such as his research, teaching and books, etc. Thus, when summarizing the root page of a sub tree in the topic hierarchy, not only is the content on the root page itself relevant, but also the contents of all the pages beneath it.

Due to the way the content is hierarchically organized, sometimes the root page itself may not contain any useful description of the nature of the topic at all while the pages beneath it are actually more informative. For example, on the "members" page of the Stanford database group Website, all the information it contains are simply an exhaustive list of each member's photo and a link to his homepage. Only by looking into the individual member's pages can one notice some terms that are frequently associated with the the topic "members" such as "address", "teaching", "research", "publication", etc. Therefore, we have proposed variations of the thematic feature and presentation features which takes into account a Web page's location in the topic hierarchy and the pages beneath it.

- ***Thematic Feature:***

Following the above discussions, the score of a candidate phrase with respect to a Web page should not be determined only by its frequency on this particular page but the whole sub-tree rooted at this page in the topic hierarchy. A simple way to incorporate the content of the sub-tree is to sum the frequencies of a phrase on all the

Web pages in the sub-tree. However, such an approach ignores the distances between the root page and its descendants in the topic hierarchy. The pages that are directly beneath the root page are usually the most important sub-topics of the root while pages further down the sub-tree are less important with respect to the sub-tree as they are usually very specific information. Thus, it is necessary to weight the nodes in the sub-tree differently when computing a term's weight. To determine the proper weight for a node in the sub-tree, we apply the theory of fractal geometry.

Fractals are mathematical objects that have high degree of redundancy (Mandelbrot 1983). These objects are made of transformed copies of themselves or part of themselves (Figure 4-1). Fractals are independent of scale and appear equally detailed at any level of magnification. Such property is known as self-similarity. Any portion of a self-similar fractal curve appears identical to the whole curve. If we shrink or enlarge a fractal pattern, its appearance remains unchanged.

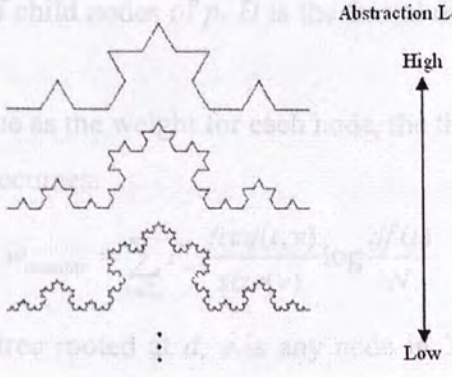

Figure 4-1 Koch curve at different abstraction level

Fractal view is a fractal-based method for controlling information displayed (Koike 1995). Fractal view provides an approximation mechanism for the observer to adjust the abstraction level and therefore control the amount of information displayed. At a lower abstraction level, more details of the fractal object can be viewed.

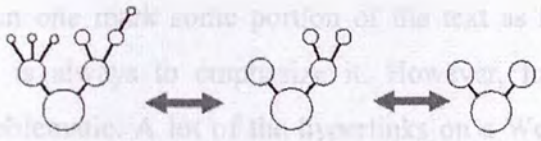

Figure 4-2 Fractal view of a tree at different abstraction level

A tree is a classical example of fractal objects. A tree is made of a lot of sub-trees, each of which is also tree. By changing the scale, the different levels of abstraction views are obtained (Figure 4-2). The idea of fractal tree can be extended to any logical tree. The degree of importance of each node is represented by its fractal value. The fractal value of focus is set to 1. The fractal value is propagated from each parent node to its descendants with the following expression:

$$F_c = F_p / N_p^{1/D}$$

where F_p is the fractal value of the parent node, F_c is the fractal value of the child node, N_p is the number of child nodes of p, D is the fractal dimension which controls the rate of decay.

Using the fractal value as the weight for each node, the thematic feature value for phrase t on Web page d becomes:

$$w_{thematic} = \sum_{v \in T_d} F_v \frac{freq(t,v)}{size(v)} \log \frac{df(t)}{N}$$

where the T_d is the sub-tree rooted at d, v is any node in T_d with fractal value F_v. Notice that this new formulation have two effects: firstly, phrases are weighted based on their frequencies on all pages in the sub-tree; secondly, the candidate keyphrases are not restricted to those appearing on the root page but are chosen from all over the sub-tree.

- ***Presentation Feature:***

The original computation of presentation feature favors any phrases that appear as anchor text or special text on a Web page. The assumption is any anchor text and special text is more important than plain text. This holds most of the time for the case

of special text, as when one mark some portion of the text as heading or bold or italicizes, the purpose is always to emphasize it. However, for anchor text, the assumption may be problematic. A lot of the hyperlinks on a Web page are created simply to facilitate navigation and the pointed pages may not be relevant at all. The hyperlink contained in the Web page template is a typical example. It is very common that the Web designer puts one or more navigation bars in the Web page template. And the texts in the hyperlinks in the template would be judged as anchor text on any Web pages created using it.

Following the above discussion, it is incorrect to treat the anchor texts indiscriminatingly. Given the topic hierarchy, we may identify which of the hyperlinks on a Web page are aggregation links and short-cut links. The anchor texts of aggregation links are more important than those of short-cut links, as the sub-topics are the most related pages of the given page. Therefore, in the calculation of the presentation feature, we treat the anchor texts of short-cut links as plain text and only rewards the anchor texts of the aggregation links.

4.3.4 Extraction of Keyphrases

To extract keyphrases from each Web page, we first identify all the candidate phrases, and then compute the value of thematic feature $w_{thematic}$ and presentation feature $w_{presentation}$. The two features need to be combined to provide a single score that is used to rank the candidate phrases. However, because $w_{thematic}$ and $w_{presentation}$ are in different scale, they first need to be normalized. This is done by dividing the $w_{thematic}$ and $w_{presentation}$ of each candidate phrases by the maximum of all the candidate phrases so that both features are transformed into the range [0, 1]. The score for a phrase t is thus:

$$score(t) = w^*_{thematic}(t) + w^*_{presentation}(t)$$

where $w^*_{thematic}$ and $w^*_{presentation}$ are the normalized thematic feature and presentation feature.

The candidate phrases for each Web page could be ranked according to score(t). To generate a keyword summary, the user only needs to specify a number k, which is the number of keywords to include in the summary. The algorithm then picks k highest ranked phrases to return to the user.

4.4 Experiments

We carried out empirical evaluation of the keyphrase extraction algorithm using Web pages in the same 5 Web sites used in the evaluation of the topic hierarchy generation algorithms. Our goals were to test whether incorporating the topic hierarchy into the keyphrase extraction algorithm could improve its effectiveness, and also to investigate the effects of the topic hierarchy's quality on the keyphrase extraction algorithm.

Table 4-1 Websites used for evaluation

URL	Type	Description
www.cs.cmu.edu	Education	School of Computer Science Carnegie Mellon University
www.db.stanford.edu	Education	Database Research Group Stanford University
www.whitehouse.gov	Government	US Government
www.research.ibm.com	Commercial	IBM Research
www.palmsource.com	Commercial	Palm Software and Palm OS

The collection of Web pages used for evaluation consists of 30 pages from each of the 5 Websites (Table 4-1). The selected pages include the homepage of each Website and pages randomly sampled from the two levels below the homepage in the topic hierarchy. We chose these pages for the following reasons: they represent the most important topics in the Website, and correspond to the roots of largest sub-trees in the topic hierarchy and thus are suitable for investigating the effect of utilizing sub-tree information in the keyword extraction algorithm. Notice that, the topic hierarchy is considered as an input to the keyword extraction algorithm. We generate the topic

hierarchies used in the keyword extraction experiments using the directed minimum spanning tree algorithm (section 3.2.3) with edges weighted by the decision tree classifier(section 3.3.2). As one of our goal is to investigate the correlation between the quality of the topic hierarchy and the performance of the keyword extraction algorithm, we also list the accuracies of the topic hierarchy used for each site(Table 4-2).

Table 4-2 Accuracies of the input topic hierarchies

Dataset	Accuracy of the topic hierarchy
www.db.stanford.edu	88.2%
www.cs.cmu.edu	98.3%
www.research.ibm.com	85.9%
www.whitehouse.gov	94.6%
www.palmsource.com	93.3%

We compared 4 different extraction schemes(Table 4-2). Using each extraction scheme, we extract a fixed number of keyphrases (10 was used in our evaluation). A human judge then assigns a numeric score to each extracted phrase that can be 0,1,2 and 3, where 0 corresponds to irrelevant and 3 corresponds to highly relevant. The different schemes can then be compared by the average score of the extracted keyphrases.

Table 4-3 Different Extraction Schemes

Extraction Scheme	Details
I-A	Compute score for each phrase using thematic feature only and restrict the candidate phrases to those occurring on the root page only.
I-B	Compute score for each phrase using thematic feature only and allow the candidate phrases to be chosen from the whole sub-tree.
II-A	Compute score for each phrase by combining thematic and presentation feature only and restrict the candidate phrases to those occurring on the root page only.
II-B	Compute score for each phrase using thematic feature only and allow the candidate phrases to be chosen from the whole sub-tree.

Another issue that needs to be addressed when using the content of the sub-tree to extract keyphrases for the root page is the computational complexity. When handling a page that is the root of a sub-tree with a very large number of nodes, for example the homepage of the Website, it would be very time consuming to process every page in the tree in order to find all the candidate phrases and compute their scores. So an important question is whether it is necessary to use the whole sub-tree in order to identify the best keyphrases for the root page. And if the answer is no, how much and what portion of the sub-tree need to be considered in order to achieve the best performance. To investigate this problem, we introduce another parameter which is the sub-tree height. The sub-tree height controls the number of levels in the sub-tree that are examined when summarizing the root page. So for example, if the sub-tree height is set to 2, only the root page and its children are examined. In the experiments, we vary the sub-tree height from 1 to 5 for each of the 4 extraction schemes and evaluate the extracted keyphrases. The average score of each extraction scheme against the sub-tree height for each of the five Websites are plotted in Figure 4-3 to 4-7.

In Figure 4-3, 4-4 and 4-5, we may clearly see that the performance for all the schemes increases with the sub-tree height, which indicates that utilizing the sub-tree content helps identifying the proper keyphrases. The effect of the sub-tree height is also evident as it increases from 1 to 3. But afterwards, increasing the sub-tree height further to 4 or 5 could result in only very limited amount of additional improvement. This suggests that the first 3 levels are the most useful portion of the sub-tree for identifying keyphrases on the root page.

We can also note from the figures that extraction scheme II-A appeared to be most effective, which suggests using the combination of thematic feature and presentation feature to score the phrases and restrict the candidate phrases to those appearing on the root page. This is in contrast to our expectation that phrases that appear on some pages in the sub-tree but not on the root page maybe good

keyphrases. This result could be best explained by the fact that by including those phrases appearing on other pages in the sub-tree in the set of candidate phrases, the size of set is greatly enlarged and so is the difficulty of selecting the correct keyphrases.

The results shown in Figure 4-6 and 4-7 (www.db.stanford.edu and www.research.ibm.com) are quite inconsistent with the other three sets of results. First, we could see that the performances keep dropping as the sub-tree height increases. Secondly, the combination of thematic and presentation feature appeared to be less effective than the thematic feature alone, especially as the sub-tree height increases. Therefore, the results for these two datasets could not prove the usefulness of the topic hierarchy for keyword extraction. However, by looking into Table 4-2, we may notice that the accuracy of topic hierarchies for the same two Websites happen to also be the lowest 2 of the 5 datasets (both are below 90%). This shows a possible correlation between the quality of the topic hierarchy and the performance of the keyword extraction algorithm: using the topic hierarchy during extraction helps only if the topic hierarchy correctly models relationships between the Web pages, otherwise, an ill-formed topic hierarchy may even hurt the performance of the keyword extraction algorithm.

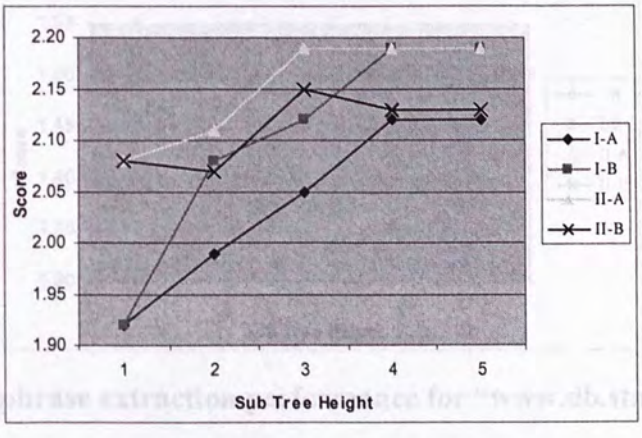

Figure 4-3 Keyphrase extraction performance for "www.cs.cmu.edu"

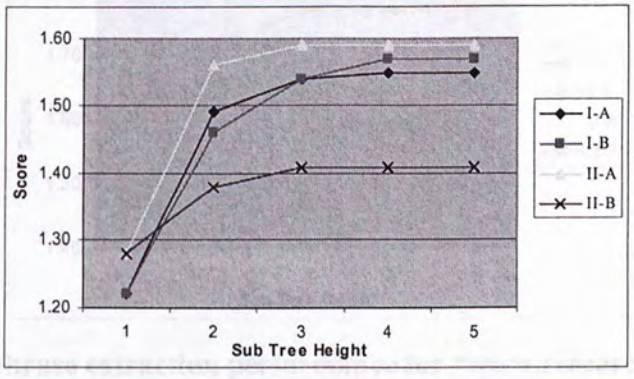

Figure 4-4 Keyphrase extraction performance for "www.palmsource.com"

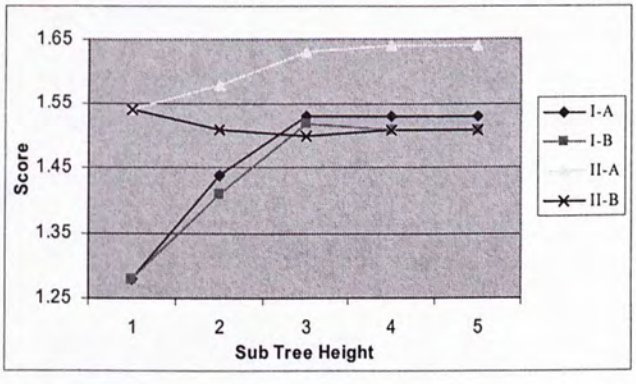

Figure 4-5 Keyphrase extraction performance for "www.whitehouse.gov"

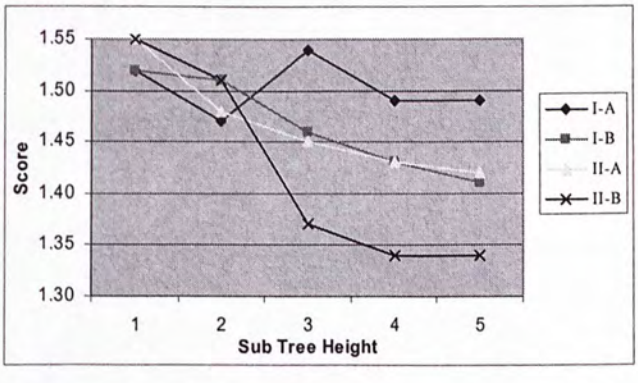

Figure 4-6 Keyphrase extraction performance for "www.db.stanford.edu"

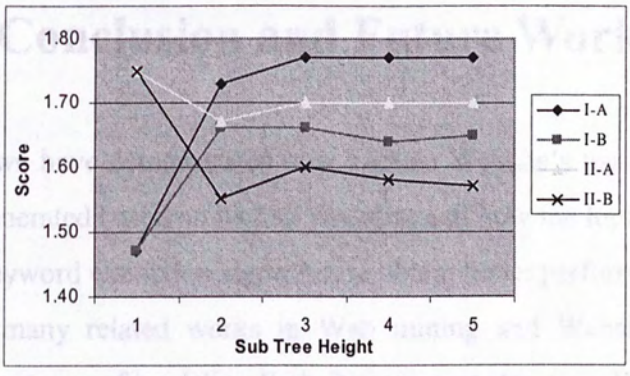

Figure 4-7 Keyphrase extraction performance for "www.research.ibm.com"

Chapter 5 Conclusion and Future Work

In this thesis, we have demonstrated how a given Website's topic hierarchy can be automatically generated based on its link structure and how the topic hierarchy can be utilized in the keyword extraction algorithm to obtain better performance.

We surveyed many related works in Web mining and Website mining and pointed out the importance of modeling Website's content structure. We have defined the topic hierarchy for Websites and demonstrated how it can be used as a general model for representing a Website's content structure. The hyperlinks among Web pages are either aggregation link or short-cut link. By identifying the aggregation links, we may build the topic hierarchy from the Website's link structure. Several graph algorithms have been adapted to extract the topic hierarchy from a Website's link graph including breadth-first search, shortest-path search and directed minimum spanning tree. We compared the unweighted and weighted graph models for representing Website's link structure. For the weighted graph models, we proposed two techniques for estimating the weight of an edge: the relevance method and the machine learning method. We have shown that the directed graph model is able to reflect the hierarchical relationships more effectively and the decision tree classifier is the best classifier to be used to estimate edge weight. When operating on the weighted graph model produced using decision tree classifier, the directed minimum spanning algorithm had the best performance in terms of the accuracy of generated topic hierarchy.

We also investigated the problem of extracting keywords to summarize Web pages in a Website. The keyword extraction algorithm utilized two different features: the thematic feature, which is derived from a phrase's frequency on the Web page and in the entire Website, and also the presentation feature, which distinguish anchor text and special text from plain text. Using the topic hierarchy, a phrase's thematic feature

is not determined by its frequency on the target page only but through combining its frequency in the sub-tree rooted at the target page in the topic hierarchy. We also proposed to use the fractal value of each node as its weight while combining the sub-tree information. The presentation feature is also adapted by discriminating anchor text for active and inactive links. The experimental results showed that features adapted using the topic hierarchy are more effective for identifying keyphrases on Web pages.

Topic hierarchy generation for Websites is a very new research topic. There are many possible improvements to our proposed techniques. One interesting direction is to investigate whether more flexible representations other than trees can be used for the topic hierarchy. Using the tree model, a restriction is that each page can have one and only one parent. However, most general ontologies are not strictly tree but a network, or directed acyclic graph more specifically.

We also believe that a variety of Website mining tasks such as Website classification could benefit from a more accurate representation of Website's content structure. In the future, we would conduct more experiments to investigate the relationships between the performance of Website mining and the accuracy of the topic hierarchy. By showing the importance of topic hierarchy, we would also have a stronger motivation to further improve techniques for topic hierarchy generation.

References:

1. Amitay, E. and C. Paris. (2000) *Automatically Summarizing Web Sites: Is There a Way Around it?* In Proceedings of the 9th ACM International Conference on Information and Knowledge Management, Mclean, VA, USA

2. Barzilay, R., and M. Elhadad. (1999). *Using Lexical Chain for Text Summarization.* In Advances in Automatic Summarization, Cambridge, Massachusetts: MIT Press

3. Berger, A., and V. Mittal. (2000) *Ocelot: a System for Summarizing Web Pages.* In proceedings of SIGIR 2000

4. Brill, E. (1995) *Transformation-Based Error-Driver Learning and Natural Language Processing: A Case Study in Part of Speech Tagging.* Computational Linguistics.

5. Brin, S. and L. Page. (1998) *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* In Proc. of the 7th World Wide Web Conference

6. Buyukkokten, O., H. Garcia-Molina and A. Paepcke. (2001), *Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices.* In Proceedings of the 10th World Wide Web Conference, Hong Kong, China

7. Carboness, J., Y. Geng and J. Goldstend. (1997) *Automated Query-Relevant Summarization and Diversity-Based Reranking.* In Proceedings of the IJCAI'97 Workshop on AI in Digital Libraries

8. Chakrabarti, S., B. Dom and P. Indyk. *Enhanced Hypertext Categorization Using Hyperlinks.* In ACM SIGMOD International Conf. on Management of Data, 1998

9. Chakrabarti, S. (2000) *Data Mining for Hypertext: A Tutorial Survey.* ACM SIGKDD Explorations, 1(2):1-11

10. Chen, Z., S. Liu, W. Liu, G. Pu and W.Y. Ma. (2002) *Building a Web Thesaurus from Web Link Structure.* In Proc. of the 25th ACM SIGIR Conf. on Research and Development in Information Retrieval, Finland, 2002

11. Cheung, K.W., and Y. Sun. (2003) *Mining Website's Clusters from Link Topology and Site Hierarchy.* In Proc. of IEEE International Conf. on Web Intelligence

12. Chu, Y.J., and T.H. Liu. (1965) *On the Shortest Arborescence of a Directed Graph.* Science Sinica,14:1396-1400.

13. Cormen, T., C. Leiserson, R. Rivest and C. Stein. (2003) *Algorithm Design and Analysis*. 2nd Edition, MIT press

14. Davison, B.D. (2000) *Topical Locality in the Web.* In Proceedings of SIGIR 2000

15. Dumais, S.T., J. Platt, D. Heckerman and M. Sahami. (1998) *Inductive Learning Algorithms and Representations for Text Categorization.* Proceedings of International Conference on Information and Knowledge Management

16. Edmons, J. (1967) *Optimum branchings*. Journal of Research of the National Bureau of Standards, 71B:233-140

17. Edmunson, H.P. (1969) *New Methods in Automatic Abstracting.* Journal of the Association of Computing Machinery 16(2)

18. Ester, M., H.P. Kriegel and M. Schubert.(2002) *Website Mining: A new way to spot Competitors, Customers and Suppliers in the World Wide Web.* In Proc. of the 8th International Conf. on Knowledge Discovery and Data Mining

19. Frank, E., G.W. Paynter, I.H. Whitten, C. Gutwin, and C.G. Nevill-Manning. (1999) *Domain-specific Keyphrase Extraction.* Proceedings of the 16th International Joint Conference on Artificial Intelligence

20. Gibson, D., J.M. Kleinberg and P. Paghavan. (1998) *Inferrring Web Communities from Link Topology.* In Proc. of 9th ACM Conf. on Hypertext and Hypermedia

21. Halliday, M.A.K and R. Hansan. (1996). *Cohension in Text.* London: Longmans

22. Hastie, T., R. Tibshirani and J. Friedman. (2001) *Elements of Statistical Learning.* Springer-Verlag

23. Kao, H., S. Lin, J. Ho, M. Chen. (2000) *Entropy-Based Link Analysis for Mining Web Informative Structures.* In Proc. of International Conf. on Information and Knowledge Management

24. Kleinberg, J.M. (1998) *Authoritative Sources in a Hyperlinked Environment.* ACM-SIAM Symposium on Discrete Algorithms

25. Koike, H. (1995) *Fractal Views: A Fractal-Based Method for Controlling Information Display.* ACM Transactions on Information Systems, ACM 13(3) 305-323

26. Krulwich, B. and C. Burkey (1996) *Learning User Information Interests Through the Extraction.* AAAI Symposium on Machine Learning in Information Extraction.

27. Kupiec, J., Pedersen, J., and F. Chen. (1995) *A Trainable Document Summarizer.* In Proceedings of the 18th International Conference on Research and Development in Information Retreival, Seattle, Washington

28. Leonidas, G. (2003) *Arborescene optimization problems solvable by Edmonds' Algorithm.* Theoretical Computer Science, 301:427-437

29. Li, W.S., O Kolak, Q. Vu and H. Takano. (2000). *Defining Logical Domains in a Website.* Proc. of 11th ACM Conf. on Hypertext and Hypermedia, San Antonio

30. Liu, N. and C. C. Yang. (2005a) *Mining Website's Topic Hierarchy.* Proceedings of the International World Wide Web Conference, Chiba, Japan

31. Liu, N. and C. C. Yang (2005b) *Automatic Extraction of Website's Content Structure from Link Structure.* Proceedings of the International Conference on Information and Knowledge Management, Bremen, Germany

32. Luhn H.P. (1958) *The Automatic Creation of Literature Abstracts.* IBM Journal of Research and Development, 159-165

33. Lynch, P. and S. Horton. (2002) *Web Style Guide: Basic Design Principles for Creating Websites*, 2nd Edition. Yale University Press

34. Mandelbrot, B. (1983) *The Fractal Geometry of Nature.* New York: W.H. Freeman

35. Munoz, A. (1996) *Compound Key word Generation From Document Databases using a Hierarchical Clustering ART Model.* Intelligence Data Analysis, 1(1), Armsterdam: Elsevier

36. Porter M.F. (1980) *An Algorithm for Suffix Stripping*, Program, 14(3) pp130-137

37. Quinlan, J.R., and R. Rivest. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann

38. Radev, D.R., H. Jing, M. Stys and D. Tam. (2004) *Centroid-based Summarization of Multiple Documents*, Information Processing and Management, 40:919-938

39. Sun, Aixin, and E.P. Lim. (2003) *Web Unit Mining – Finding and Classifying subgraphs of Web Pages.* In Proc. of International Conf. on Information and Knowledge Management

40. Turney, I.H. (1999) *Learning Algorithms for Keyphrase Extraction.* Information Retrieval, 2 (4)

41. Whitten, I.H, G.W. Paytner, E. Frank, C. Gutwin and C.G. Nevill-Manning. (1999) *KEA: Practical Automatic Keyphrase Extraction.* Proceedings of Digital Libraries 99, ACM Press

42. Whitten, I.H. and E. Frank (2005) *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufman, San Fracisco, 2005.

43. Yi, L., B. Liu, X. Li. (2003) *Eliminating Noisy Information in Web Pages for Data Mining.* In Proc. of 9th International Conf. on Knowledge Discovery and Data Mining

44. Yates, R. and B. Neto. (1999) *Modern Information Retrieval,* Addison Wesley

45. Zhang, Y., N.Z. Heywood and E. Milios. (2003) *Summarizing Websites Automatically.* In Proc. of 16th Conf. of the Canadian Society for Computational Studies of Intelligence