2012

Web Page Classification and Hierarchy Adaptation

Xiaoguang Qi
Lehigh University

WEB PAGE CLASSIFICATION AND HIERARCHY ADAPTATION

by

XiaoGuang Qi

A Dissertation

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Computer Science

Lehigh University

January 2012

This dissertation is accepted in partial

fulfillment of the requirements for the degree of

Doctor of Philosophy.

(Date)

(Accepted Date)

Brian D. Davison
(Committee Chair)

Henry S. Baird

Hector Munoz-Avila

Lin Lin
(Department of Management)

Acknowledgments

I heartily thank my advisor, Prof. Brian D. Davison, for his guidance, encouragement, and support throughout my Ph.D. study. Without his guidance, I could not have finished this dissertation.

I am also grateful to my committee members, professors Henry S. Baird, Lin Lin, and Hector Munoz-Avila for their excellent guidance and suggestions.

I owe my gratitude to Dr. Cliff Brunk and Dr. Dmitry Pavlov from Yandex Labs, and Dr. Dennis DeCoste from Microsoft Live Labs for offering me the internship opportunities at these companies. The internship experiences provided me valuable opportunities to work with, and learn from talented experts in my research area, to acquire first-hand knowledge in search industry, and to gain access to large-scale, real-world data and unparalleled computing resources.

Many thanks to my labmates Baoning Wu, Lan Nie, Vinay Goel, Jian Wang, Shruti Bhandari, YaoShuang Wang, Na Dai, Ovidiu Dan, Liangjie Hong, Xiong Xiong, Zhenzhen Xue, Zaihan Yang, and Dawei Yin for their collaboration and helpful discussions.

My gratitude to my family, especially my wife, Xiaoning Yang, for their support and

encouragement in pursuing the degree. This dissertation would not have been possible without their love and support.

My work has been financially supported by different sources which include the National Science Foundation under award IIS-0328825, IIS-0545875, and Microsoft Live Labs under Accelerating Search funding program.

Contents

List of Tables

List of Figures

Abstract

Classification is a supervised learning problem in which a classifier is trained on a set of data labeled with predefined categories and then applied to label future examples. It plays a fundamental role in a number of essential tasks in information retrieval and management. Advanced classification approaches will benefit systems that search or manage web information, as well as other types of information in general.

In this dissertation, we investigate methods to improve classification from two aspects: feature enhancement and hierarchy adaptation. For feature enhancement, information from the neighboring pages on the web graph is studied. Novel methods to effectively utilize such neighboring information to improve classification are proposed and analyzed. For hierarchy adaptation, evolutionary computation methods are used to search for better hierarchies in order to improve hierarchical classification. We also investigate problems that impede user navigation in hierarchies, and propose novel methods to facilitate efficient navigation. Experiments on multiple real-world datasets show that the proposed approaches can significantly outperform previous state-of-the-art methods.

Chapter 1

Introduction

1.1 Motivation

Classification is a supervised learning problem in which a classifier is trained on a set of data labeled with predefined categories and then applied to label future examples. It plays a fundamental role in a number of essential tasks on information retrieval and management. On the Web, classification of page content is essential to focused crawling [31], to assist development of web directories such as those provided by Yahoo [208] and the Open Directory Project (ODP) [140], to topic-specific web link analysis [84, 136, 162], to analysis of the topical structure of the Web [30], and to contextual advertising [21, 22].

A classifier is usually evaluated with regard to how accurately it can label unseen instances. The accuracy of the classifier often directly affects the performance of the system built on top of it. Inaccurate classification results will lead to an overall performance

degradation, which in turn adversely affects user experience, and sometimes causes direct monetary loss. For example, in a ranking system, if an important page is incorrectly classified into a category that has no connection with the query, it will be considered less relevant by the ranking algorithm, and thus not be ranked as high as it should be. In contextual advertising, if a page about a car racing game is incorrectly categorized as an auto dealer, the advertisement matching system will display irrelevant advertisements on the page, losing the clicks it could have attracted using correct classification.

Since many information retrieval tasks depend on accurate classification, research in advanced classification approaches will benefit systems that search or manage information on the Web, as well as other types of information in general. In this dissertation, two important topics related to classification are proposed and studied: web page classification and hierarchy adaptation.

1.1.1 Web page classification

Since its emergence, the World Wide Web has changed people's lives in almost every possible way. In less than two decades, the Web has evolved from an information repository into a powerful platform that supports a wide variety of essential tasks and applications. Information and applications on the Web range from education to entertainment, from home decoration to space technology, from photo sharing among friends and family to world-wide events that attract global attention. Classification on the Web means much more than merely assigning category labels and organizing information. It supports at

least two core applications on the Web: web search and advertising, making it an important topic of broad interest.

Traditional web page classification treats web pages as text documents without considering the additional features that the Web can provide (e.g., hyperlinks, markups). Such an approach usually yields suboptimal performance as web pages themselves often contain inadequate information for classification. In our preliminary experiment in which we applied support vector machines to a 3,600-page ODP dataset, merely 77% of the pages were correctly classified into one of the twelve broad categories when only on-page textual features are used. The low performance can be caused by various reasons. Unlike text documents, web pages no longer rely on text to present its information. Some pages are mainly composed with pictures and flash media where text can be missing, misleading, or not enough for a text classifier to make a reasonable prediction.

Previous research shows that incorporating link information along with the content of web pages can enhance classification. However, existing approaches either rely heavily on labeled pages, or only utilize certain types of neighbors. In this dissertation, we propose the Neighboring Algorithm which uses the class or topic vector of neighboring pages to help in the categorization of a web page. Our approach does not require labeled pages in the neighborhood, and can take into consideration multiple types of neighboring pages.

When using content of neighbors, existing work typically uses information from neighboring pages as a whole. We argue that different fields of information on the neighboring pages bear different importance. Inspired by the success of the fielded extension to BM25

in information retrieval [163], we conjecture that permitting text from different fields of neighboring pages to contribute differently may improve classification performance. In this dissertation, we propose the F-Neighbor Algorithm, as a fielded extension to our Neighboring Algorithm which uses class information and full content of neighboring pages. In particular, we break up neighboring web pages, as well as the page to be classified into several text fields (e.g., title, anchor text, body text), and combine the text fields according to the individual importance they have.

1.1.2 Hierarchy adaptation

The second topic in this dissertation, hierarchy adaptation, evolves from our interest in classification, with the subject focused on hierarchical classification.

A hierarchy, or a taxonomy, is a hierarchical structure that organizes concepts or topics according to their relations. Classification can be performed based on a flat set of categories, or on categories organized as a hierarchy. Classification based on a flat set of categories is called flat classification, whereas hierarchy-based classification is called hierarchical classification. Hierarchical classification has been shown to offer superior performance than flat classification (e.g., [58, 117, 16]). By classifying objects first into high level categories, and then iteratively into finer-grained subcategories, the classifier at each branching point should have an easier task than classifying into all categories at once.

Hierarchical classification is typically performed based on human-defined hierarchies.

Since such hierarchies reflect a human view of the domain, they are easy for people to understand and utilize. However, these hierarchies are usually created without consideration for automated classification. As a result, hierarchical classification based on such hierarchies is unlikely to yield optimal performance. Previous research on improving hierarchical classification mostly focused on making classification methods more effective or utilizing additional features that are made available by the hierarchical structure, without any change to the predefined hierarchy. In most applications, we are only interested in the target category into which an instance is classified, regardless of the internal process of how an instance is classified at each level and iteratively passed to a descendant category. This gives us the opportunity to improve classification accuracy by using a different internal hierarchical structure that better suits automatic classification. In this dissertation, we propose a new method based on evolutionary computation to create hierarchies with better classification accuracies.

Besides their utility in automatic classification, hierarchies are usually used in the organization of information to assist human browsing. We next investigate adaptation of hierarchies for better navigation. A major drawback of hierarchies is that they require users to have the same view of the topics as the hierarchy creator. That is, when a user follows a top-down path to find the specific topic of her interest, she has to make choices along the constrained sequence that is present in the hierarchy. As a result, users who do not share that mental taxonomy are likely to have additional difficulties in finding the desired topic. For example, in one hierarchy, information about Emacs, an open

source text editor, can be organized under /Software/OpenSource/Editors/Emacs. Such a hierarchy will not be helpful for a user who looks for information about Emacs but does not know that Emacs is an open source package. This problem can be somewhat reduced by remedies like cross-topic links (as used in ODP). However, after adding such links, the target nodes of the links will have more than one parent, making the hierarchy no longer a tree. Under this approach, nodes are not replicated—links are just added to the graph. Logically, such links are only appropriate when the alternative path also applies to all descendants of the topic. Furthermore, such links bring dependencies among topics and increase editing cost: editing one topic may result in changes in its linked topics and then the linked topics of those topics. In this dissertation, we propose a new approach to hierarchy expansion which is able to provide more flexible views to better assist human browsing for desired information. Based on an existing hierarchy, our algorithm finds possible alternative paths and generates a new, expanded hierarchy with flexibility in user browsing choices.

In summary, web page classification and hierarchy adaptation are important and interesting topics that need further study. In this dissertation, we propose various algorithms to improve web classification by incorporating neighboring information, and to adapt hierarchies for better classification as well as human browsing. These approaches can either enhance automatic classification accuracy or improve user experience when browsing for information through large hierarchies.

1.2 Contributions

The principal contributions of this dissertation include:

- A new approach to classify web pages using information from neighboring pages, and an examination of the relative contribution from different neighbor types.
- Demonstrated usefulness of field information in HTML document classification, and a finding that emphasizing page titles, especially parent titles, over other text can bring improvement.
- A new approach to improved classification by hierarchy adaptation using genetic operators customized for hierarchies.
- An analysis of the problems of existing hypernym/hyponym taxonomies, and a new approach to generate flexible taxonomies that is able to alleviate the found problems.

I am the lead author of six peer-reviewed papers on the topic of web classification and hierarchy adaptation. Two of these papers [151, 152] are published as full papers in CIKM'06 and SIGIR'08, which are among the top prestigious conferences on research in web search, information retrieval, and knowledge management. A survey on web page classification [153] is published in *ACM Computing Surveys*. A hierarchy adaptation paper [156] is published as a short paper in CIKM'10. A paper on web link classification [155] was published in the Proceedings of The Third International Workshop On Adversarial Information Retrieval on the Web (AIRWeb). A paper on adaptation of hierarchies for

improved classification [154] is accepted by CIKM'11 as a poster paper.

I also worked as a co-author of a SIGIR'06 paper [136], an AIRWeb'09 paper [48], a Hypertext'11 paper [49], a WWW'11 poster [50], and a non-peer-reviewed paper in TREC'09 [216].

1.3 Dissertation Outline

The dissertation is organized as follows. In Chapter 2, we review the background and related work on web classification and hierarchy adaptation. In Chapter 3, we propose the Neighboring Algorithm to use information of neighboring pages to help classification of web pages. The F-Neighbor Algorithm is proposed in Chapter 4, utilizing field information in web classification. In Chapter 5, we introduce a hierarchy evolution algorithm to improve hierarchies for more accurate hierarchical classification. An algorithm to improve hierarchies for better human browsing experiences is proposed in Chapter 6. We end this dissertation by a discussion and conclusion in Chapter 7.

Part of the work presented in this dissertation is in collaboration with Dr. Brian D. Davison, Dawei Yin, and Zhenzhen Xue.

Chapter 2

Background

2.1 Web Page Classification

Classification plays a vital role in many information management and retrieval tasks. On the Web, classification of page content is essential to focused crawling, to the assisted development of web directories, to topic-specific web link analysis, to contextual advertising, and to analysis of the topical structure of the Web. Web page classification can also help improve the quality of web search. In this section, we will review existing approaches to web classification.

2.1.1 Problem definition

Web page classification, also known as web page categorization, is the process of assigning a web page to one or more predefined category labels. Classification is traditionally posed

as a supervised learning problem [131] in which a set of labeled data is used to train a classifier which can be applied to label future examples.

The general problem of web page classification can be divided into more specific problems: subject classification, functional classification, sentiment classification, and other types of classification. Subject classification is concerned about the subject or topic of a web page. For example, judging whether a page is about "arts", "business" or "sports" is an instance of subject classification. Functional classification cares about the role that the web page plays. For example, determining a page to be a "personal homepage", "course page" or "admission page" is an instance of functional classification. Sentiment classification focuses on the opinion that is presented in a web page, i.e., the author's attitude about some particular topic. Other types of classification include genre classification (e.g., [225]), search engine spam classification (e.g., [81, 27]) and so on. Here, we focus on subject and functional classification.

Based on the number of classes in the problem, classification can be divided into binary classification and multi-class classification, where binary classification categorizes instances into exactly one of two classes (as in Figure 2.1(a)); multi-class classification deals with more than two classes. Based on the number of classes that can be assigned to an instance, classification can be divided into single-label classification and multi-label classification. In single-label classification, one and only one class label is to be assigned to each instance, while in multi-label classification, more than one class can be assigned to an instance [193]. If a problem is multi-class, e.g., four-class classification, it means four

classes are involved, e.g., Arts, Business, Computers, and Sports. It can be either single-label, where exactly one class label can be assigned to an instance (as in Figure 2.1(b)), or multi-label, where an instance can belong to any one, two, or all of the classes (as in Figure 2.1(c)). Based on the type of class assignment, classification can be divided into hard classification and soft classification. In hard classification, an instance can either be or not be in a particular class, without an intermediate state; while in soft classification, an instance can be predicted to be in some class with some likelihood (often a probability distribution across all classes, as in Figure 2.1(d)).

Based on the organization of categories, web page classification can also be divided into flat classification and hierarchical classification. In flat classification, categories are considered parallel, i.e., one category does not supersede another. While in hierarchical classification, the categories are organized in a hierarchical tree-like structure, in which each category may have a number of subcategories. An illustration is shown in Figure 2.2. Section 2.2 will address the issue of hierarchical classification further.

2.1.2 Applications of web classification

Classification of web content is essential to many information retrieval tasks. Here, we present a number of such tasks.

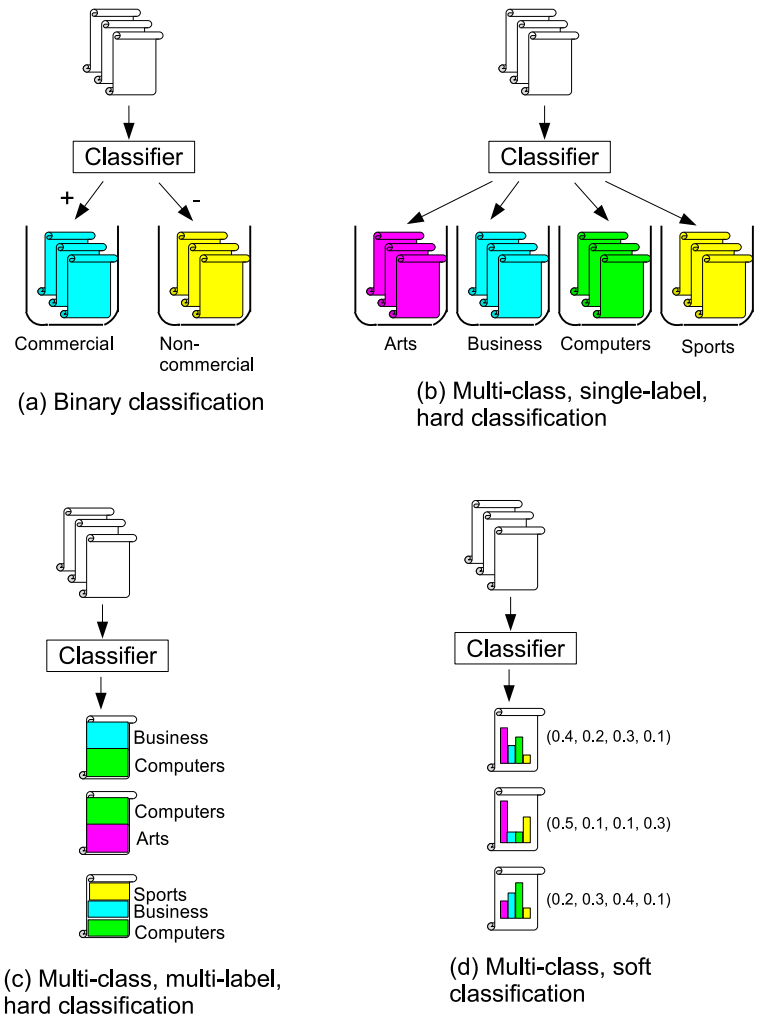


Figure 2.1: Types of classification

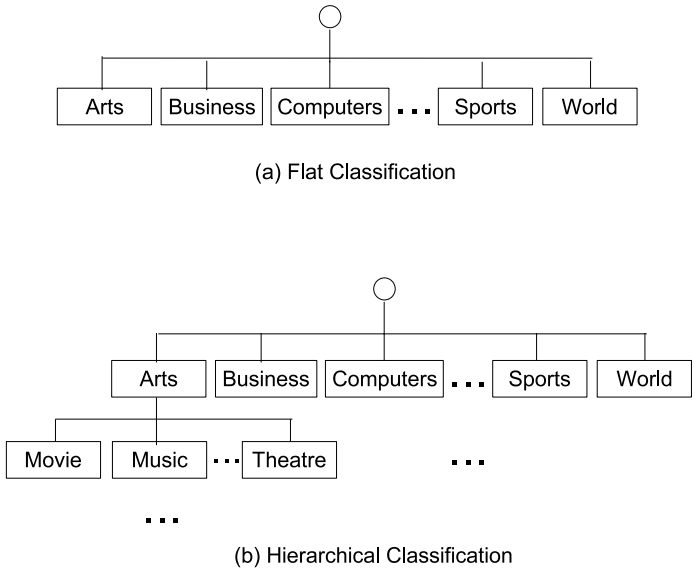


Figure 2.2: Flat classification and hierarchical classification.

Constructing, maintaining or expanding web directories (web hierarchies)

Web directories, such as those provided by Yahoo![1] and the dmoz Open Directory Project (ODP)[2], provide an efficient way to browse for information within a predefined set of categories. Currently, these directories are mainly constructed and maintained by editors, requiring extensive human effort. As of April 2011, it was reported [140] that there are 90,561 editors involved in the dmoz ODP. As the Web changes and continues to grow, this manual approach will become less effective. One could easily imagine building classifiers to help update and expand such directories. For example, Huang et al. [93, 94] proposed an approach to automatic creation of classifiers from web corpora based on user-defined hierarchies. Furthermore, with advanced classification techniques, customized (or even

[1]Yahoo!: `http://www.yahoo.com/`

[2]The dmoz Open Directory Project (ODP): `http://www.dmoz.org/`

dynamic) views of web directories can be generated automatically.

Improving quality of search results

Query ambiguity is among the problems that undermine the quality of search results. For example, the query term "bank" could mean the border of a body of water or a financial establishment. Various approaches have been proposed to improve retrieval quality by disambiguating query terms. Chekuri et al. [33] proposed to use automatic web page classification to increase the precision of web search. A statistical classifier, trained on existing web directories, is applied to new web pages and produces an ordered list of categories in which the web page could be placed. At query time the user is asked to specify one or more desired categories so that only the results in those categories are returned, or the search engine returns a list of categories under which the result pages would fall. This approach works when the user is looking for a known item. In such a case, it is not difficult to specify the preferred categories. However, there are situations in which the user is less certain about what documents will match, for which the above approach does not help much.

Search results are usually presented in a ranked list. However, presenting categorized, or clustered, results can be more useful to users. An approach proposed by Chen and Dumais [35] classifies search results into a predefined hierarchical structure and presents the categorized view of the results to the user. Their user study demonstrated that the category interface is liked by users better than the result list interface, and is more efficient

for users to find the desired information. Compared to the approach suggested by Chekuri et al., this approach is less efficient at query time because it categorizes web pages on-the-fly. However, it does not require the user to specify desired categories; therefore, it is more helpful when the user does not know the query terms well. Similarly, Käki [100] also proposed to present a categorized view of search results to users. Experiments showed that the categorized view is beneficial for the users, especially when the traditional ranking of results is not satisfying.

In 1998, Page et al. developed the link-based ranking algorithm called PageRank [142]. PageRank calculates the authoritativeness of web pages based on a graph constructed by web pages and their hyperlinks, without considering the topic of each page. Since then, research has been conducted to differentiate authorities of different topics. Haveliwala [85] proposed Topic-sensitive PageRank, which performs multiple PageRank calculations, one for each topic. When computing the PageRank score for each category, the random surfer jumps to a page in that category at random rather than just any web page. This has the effect of biasing the PageRank to that topic. This approach needs a set of pages that are accurately classified. Nie et al. [136] proposed another web ranking algorithm that considers the topics of web pages. In that work, the contribution that each category has to the authority of a web page is distinguished by means of soft classification, in which a probability distribution is given for a web page being in each category. In order to answer the question "to what granularity of topic the computation of biased page ranks make sense," Kohlschutter et al. [104] conducted analysis on ODP categories, and showed that

ranking performance increases with the ODP level up to a certain point.

Helping question answering systems

A question answering system may use classification techniques to improve its quality of answers. Yang and Chua [211, 212] suggested finding answers to list questions (where a set of distinct entities are expected, e.g., "name all the countries in Europe") through web page functional classification. Given a list question, a number of queries are formulated and sent to search engines. The web pages in the results are retrieved and then classified by decision tree classifiers into one of the four categories: collection pages (containing a list of items), topic pages (representing an answer instance), relevant pages (supporting an answer instance), and irrelevant pages. In order to increase coverage, more topic pages are included by following the outgoing links of the collection pages. After that, topic pages are clustered, from which answers are extracted.

Additionally, there have been a number of approaches to improving quality of answers by means of question classification [83, 87, 111, 220] which are beyond the scope of this dissertation.

Building efficient focused crawlers or vertical (domain-specific) search engines

When only domain-specific queries are expected, performing a full crawl is likely inefficient. Chakrabarti et al. [31] proposed an approach called focused crawling, in which only documents relevant to a predefined set of topics are of interest. In this approach, a classifier is used to evaluate the relevance of a web page to the given topics so as to provide

evidence for the crawl boundary.

Other applications

Besides the applications discussed above, web page classification is also useful in web content filtering [82, 36], assisted web browsing [6, 144, 99], contextual advertising [20, 22], ontology annotation [172], and in knowledge base construction [46].

2.1.3 The difference between web classification and text classification

The more general problem of text classification [170, 1, 185, 191, 171, 25, 15] is beyond the scope of this dissertation. Compared with standard text classification, classification of web content is different in the following aspects. First, traditional text classification is typically performed on "structured corpora with well-controlled authoring styles" [33], while web collections do not have such a property. Second, web pages are semi-structured documents in HTML, so that they may be rendered visually for users. Although other document collections may have embedded information for rendering and/or a semi-structured format, such markup is typically stripped for classification purposes. Finally, web documents exist within a hypertext, with connections to and from other documents. While not unique to the Web (consider for example the network of scholarly citations), this feature is central to the definition of the Web, and is not present in typical text classification problems. Therefore, web classification is not only important, but distinguished from traditional text classification, and thus deserving of the focused review and study found in this dissertation.

2.1.4 Features

In this subsection, we review the types of features found to be useful in web page classification research.

Written in HTML, web pages contain additional information, such as HTML tags, hyperlinks and anchor text (the text to be clicked on to activate and follow a hyperlink to another web page, placed between HTML <A> and </A> tags), other than the textual content visible in a web browser. These features can be divided into two broad classes: on-page features, which are directly located on the page to be classified, and features of neighbors, which are found on the pages related in some way with the page to be classified.

Using on-page features

Textual content and tags. Directly located on the page, the textual content is the most straightforward feature that one may consider to use. However, due to the variety of uncontrolled noise in web pages, directly using a bag-of-words representation for all terms may not achieve top performance. Researchers have tried various methods to make better use of the textual features. One popular method is feature selection, which we cover in Section 2.1.5. Term n-gram representation is another method that is found to be useful. Mladenic [132] suggested an approach to automatic web page classification based on the Yahoo! hierarchy. In this approach, each document is represented by a vector of features, which includes not only single terms, but also up to 5 consecutive words. The advantage of using n-gram representation is that it is able to capture the concepts

expressed by a sequence of terms (phrases), which are unlikely to be characterized using single terms. Imagine a scenario of two different documents. One document contains the phrase "New York". The other contains the terms "new" and "york", but the two terms appear far apart. A standard bag-of-words representation cannot distinguish them, while a 2-gram representation can. However, an n-gram approach has a significant drawback; it generates a space with much higher dimensionality than the bag-of-words representation does. Therefore, it is usually performed in combination with feature selection.

One obvious feature that appears in HTML documents but not in plain text documents is HTML tags. It has been demonstrated that using information derived from tags can boost the classifier's performance. Golub and Ardo [78] derived significance indicators for textual content in different tags. In their work, four elements from the web page are used: title, headings, metadata, and main text. They showed that the best result is achieved from a well-tuned linear combination of the four elements. This approach only distinguished the four types of elements while mixing the significance of other tags. Kwon and Lee [112, 113] proposed classifying web pages using a modified k-Nearest Neighbor algorithm, in which terms within different tags are given different weights. They divided all the HTML tags into three groups and assigned each group an arbitrary weight.

Thus, utilizing tags can take advantage of the structural information embedded in the HTML files, which is usually ignored by plain text approaches. However, since most HTML tags are oriented toward representation rather than semantics, web page authors may generate different but conceptually equivalent tag structures. Therefore, using HTML

tagging information in web classification may suffer from the inconsistent formation of HTML documents.

Good quality document summarization can accurately represent the major topic of a web page. Shen et al. [175] proposed an approach to classifying web pages through summarization. They showed that classifying web pages based on their summaries can improve accuracy by around 10% as compared with content based classifiers.

Rather than deriving information from the page content, Kan and Thi [101, 102] demonstrated that a web page can be classified based on its URL. Baykan et al. [10, 11] further improved the quality of purely-URL based classification by using all letter n-gram combinations as features. While not providing ideal accuracy, this type of approach eliminates the necessity of downloading the page. Therefore, it is especially useful when the page content is not available or time/space efficiency is strictly emphasized.

Visual analysis. Each web page has two representations, if not more. One is the text representation written in HTML. The other one is the visual representation rendered by a web browser. They provide different views of a page. Most approaches focus on the text representation while ignoring the visual information. Yet the visual representation is useful as well.

A web page classification approach based on visual analysis was proposed by Kovacevic et al. [105], in which each web page is represented as a hierarchical "visual adjacency multigraph." In the graph, each node represents an HTML object and each edge represents the spatial relation in the visual representation. Based on the result of visual analysis,

heuristic rules are applied to recognize multiple logical areas, which correspond to different meaningful parts of the page. They compared the approach to a standard bag-of-words approach and demonstrated great improvement. In a complementary fashion, a number of visual features, as well as textual features, were used in the web page classification work by Asirvatham and Ravi [7]. Based on their observation that research pages contain more synthetic images, the histogram of the images on the page is used to differentiate between natural images and synthetic images to help classification of research pages.

Although the visual layout of a page relies on the tags, using visual information of the rendered page is arguably more generic than analyzing document structure focusing on HTML tags [105]. The reason is that different tagging may have the same rendering effect. In other words, sometimes one can change the tags without affecting the visual representation. Based on the assumption that most web pages are built for human eyes, it makes more sense to use visual information rather than intrinsic tags.

On-page features are useful but they provide information only from the viewpoint of the page creator. Sometimes it is necessary to use features that do not reside on the page. We will discuss this issue next. Besides the features discussed above, self-tagging systems like that used by Technorati[3] allow authors to associate their blogs with arbitrary category names. This feature, as will be discussed in Section 2.1.7, is also useful in classification.

[3]Technorati: `http://www.technorati.com/`

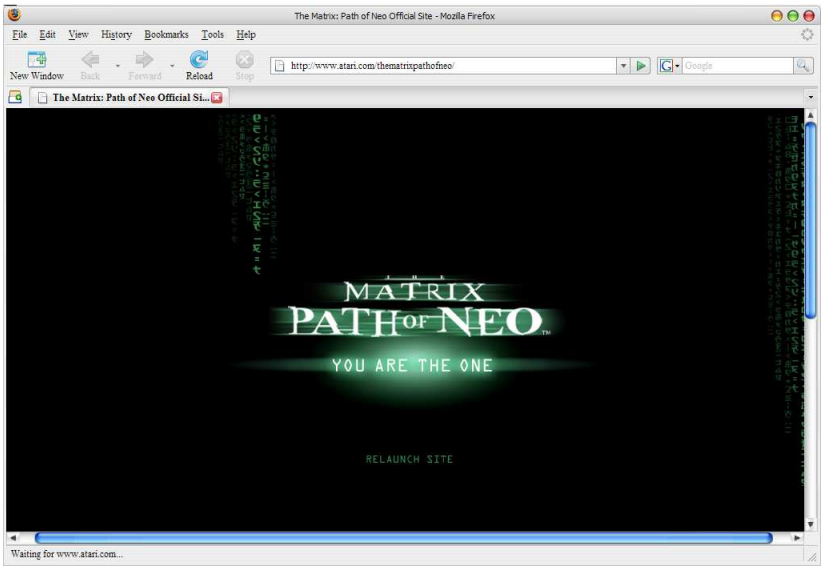


Figure 2.3: An example web page which has few useful on-page features.

Using features of neighbors

Motivation. Although web pages contain useful features as discussed above, in a particular web page these features are sometimes missing, misleading, or unrecognizable for various reasons. For example, some web pages contain large images or flash objects but little textual content, such as in the example shown in Figure 2.3. In such cases, it is difficult for classifiers to make reasonable judgments based on features on the page.

In order to address this problem, features can be extracted from neighboring pages that are related in some way to the page to be classified to supply supplementary information for categorization. There are a variety of ways to derive such connections among pages. One obvious connection is the hyperlink. Since most existing work that utilizes features of neighbors is based on hyperlink connections, in the following, we focus on hyperlink

connections. However, other types of connections can also be derived; and some of them have been shown to be useful for web page classification. These types of connections are discussed later.

Underlying assumptions. When exploring the features of neighbors, some assumptions are implicitly made in existing work. Usually, it is assumed that if pages p_a and p_b belong to the same category, pages neighboring them in the web graph share some common characteristics. This assumption does not require that the neighboring pages belong to the same category as p_a and p_b do. In the following, we refer to this thesis as the *weak assumption*. The weak assumption works in both subject classification and functional classification. Under the weak assumption, a classifier can be derived from the features of the neighboring pages of training examples, and used to predict the categories of testing examples based on the features of their neighbors.

In subject classification, a stronger assumption is often made—that a page is much more likely to be surrounded by pages of the same category. In other words, the presence of many "sports" pages in the neighborhood of p_a increases the probability of p_a being in "sports". We term this the *strong assumption*. The strong assumption requires a strong correlation between links and topics of web pages. Davison [52] showed that linked pages were more likely to have terms in common. Chakrabarti et al. [30] studied the topical structure of the Web and showed that pages tend to link to pages on the same topic. Similarly, Menczer [126] also showed a strong correlation between links and content of web pages. The strong assumption has been shown to work well in subject classification

on broad (i.e., high-level) categories. However, evidence for its validity on fine-grained categories is lacking. Furthermore, it seems unlikely that the strong assumption works in functional classification. Under the strong assumption, one might build statistical classifiers to predict the category of the page in question simply by taking the majority class of its neighboring pages.

Neighbor selection. Another question when using features from neighbors is that of which neighbors to examine. Existing research mainly focuses on pages within two steps of the page to be classified. At a distance no greater than two, there are six types of neighboring pages according to their hyperlink relationship with the page in question: parent, child, sibling, spouse, grandparent and grandchild, as illustrated in Figure 2.4. The effect and contribution of the first four types of neighbors have been studied in existing research. Although grandparent pages and grandchild pages have also been used, their individual contributions have not yet been specifically studied. In the following, we group the research in this direction according to the neighbors that are used.

In general, directly incorporating text from parent and child pages into the target page does more harm than good because parent and child pages are likely to have different topics than the target page [29, 73, 214]. This, however, does not mean that parent and child pages are useless. The noise from neighbors can be greatly reduced by at least two means: using an appropriate *subset* of neighbors, and using an appropriate *portion* of the content on neighboring pages. Both methods have been shown to be helpful.

Using a subset of parent and child pages can reduce the influence from pages on

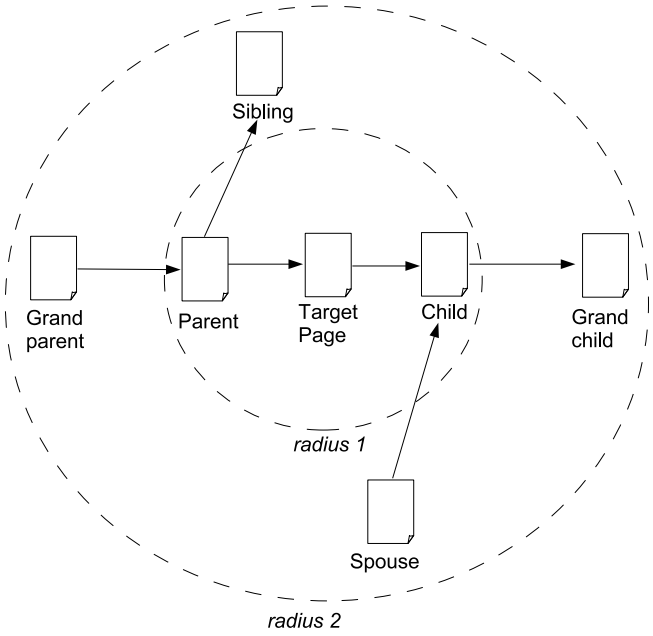


Figure 2.4: Neighbors within radius of two.

different topics than the target page. For example, while utilizing parent and child pages, Oh et al. [141] require the content of neighbors to be sufficiently similar to the target page. Using a portion of content on parent and child pages, especially the content near enough to the hyperlink that points to the target page, can reduce the influence from the irrelevant part of neighboring pages. Usually, title, anchor text, and the surrounding text of anchor text on the parent pages are found to be useful. This family of approaches takes advantage of both hyperlinks and HTML structure information. Below, we review some existing approaches of this type.

Attardi et al. [8] proposed to use the title, anchor text, and a portion of text surrounding the anchor text on parent pages to help determine the target page's topic, and showed

promising results. Fürnkranz [64] used features on parent pages like anchor text, the neighborhood of anchor text, and the headings that precede the link, and showed improvement over the classifier that uses text on the target page alone. In later work [65], an interesting approach was proposed by Fürnkranz in which text on the parent pages surrounding the link is used to train a classifier instead of text on the target page. As a result, a target page will be assigned multiple labels by such a classifier, one for each incoming link. These labels are then combined by some voting scheme to form the final prediction of the target page's class. Yang et al. [214] reviewed various approaches to hypertext classification. Their results are mixed, finding that identification of hypertext regularities and appropriate representations are crucial to categorization performance. They note, however, that "algorithms focusing on automated discovery of the relevant parts of the hypertext neighborhood should have an edge over more naive approaches." Sun et al. [183] showed that SVM classifiers using the text on the target page, page title (as separate features), and anchor text from parent pages can improve classification compared with a pure text classifier. Similarly, Glover et al. [76] demonstrated that utilizing extended anchortext (the surrounding text of anchortext, including the anchor text itself) from parent pages can improve the accuracy compared with the classifier which uses on-page content only. Utard and Fürnkranz [194] also proposed to use a portion of text as opposed to full text on parent pages. They studied individual features from parent pages such as anchor text, the neighborhood of anchor text, the paragraph containing the link, the headings before the anchor text, as well as the text on the target page, and showed significant improvement of

pair-wise combination of such features over the individual features. Besides directly using a portion of the text on parent pages, implicitly utilizing information from parent pages could also be successful, such as applying wrapper learners to anchor text on parent pages proposed by Cohen [44].

Sibling pages are even more useful than parents and children. This was empirically demonstrated by Chakrabarti et al. [29] and again by Qi and Davison [151]. Such sibling relationships can also help in relational learning of functional categories [177].

Using multiple types of neighbors could provide additional benefit. Calado et al. [24] studied the use of several link similarity measures in web page topical classification, in which the link similarities are derived from hyperlinked neighbors within two steps. Appropriate combinations of such link similarities and textual classifiers can make great improvement over textual classifiers. Qi and Davison [151] proposed a method to enhance web page classification by utilizing the class and content information from neighboring pages in the link graph. The categories represented by four kinds of neighbors (parents, children, siblings and spouses) are combined to help with the page in question. That study of the contribution of the four types of neighbors revealed that while sibling pages are the most important type of neighbor to use, the other types are also of value.

Instead of individually considering each target page together with its neighbors, some algorithms may collectively consider the class labels of all the nodes within a graph. This type of approach is discussed in Section 2.1.5.

Proper use of information from neighbors can generate a more accurate, more comprehensive representation of the web page. Therefore, it does not only benefit classification as discussed above, but also improves quality of retrieval and unsupervised learning. Incorporating information from hyperlinked neighboring pages, Sugiyama et al. [181] proposed a method to refine the TFIDF representation of a web page to improve retrieval quality. By applying graph partitioning methods on a crafted graph combining direct hyperlinks, co-citation relationships, and textual similarities, He et al. [86] was able to generate better clusters. Drost et al. [56] proposed to find communities in linked data by using similarity metrics based on co-citation and bibliographic coupling relationships, as well as content similarity. Angelova and Siersdorfer [4] proposed an approach to linked document clustering by means of iterative relaxation of cluster assignments on a linked graph.

In summary, on one hand, although parent, child, sibling, and spouse pages are all useful in classification, siblings are found to be the best source; on the other hand, since using information from neighboring pages may introduce extra noise, they should be used carefully. The effect of grandparent and grandchild pages has not been well studied.

As mentioned earlier, little work has examined the effect of pages beyond two steps away. There are at least two reasons for this: first, due to the explosive number of neighbors, utilizing features of neighbors at a long distance is expensive; second, the farther away the neighbors are, the less likely they have topics in common with the page being classified [30], and thus they are less useful in classification.

Features of neighbors. The features that have been used from neighbors include

labels, partial content (anchor text, the surrounding text of anchor text, titles, headers), and full content.

Some researchers take advantage of neighboring pages which have already been labeled by humans. For example, Chakrabarti et al. [29], Slattery and Mitchell [177], and Calado et al. [24] used the labels of neighbors in their work. The advantage of directly using labels is that human labeling is more accurate than classifiers. The disadvantage is that these labels are not always available. (Human-labeled pages, of course, are available on only a very small portion of the Web.) When the labels are not available, these approaches would either suffer significantly in terms of coverage (leaving a number of pages undecidable) or reduce to the result of traditional content-based classifiers.

As discussed earlier, using partial content of neighbors can reduce the effect of irrelevant topics on neighboring pages. As one kind of partial content of parents, anchor text is usually considered a tag or a concise description of the target page [2, 52]. It is useful in web classification as well as in web search. However, given that anchor text is usually short, it may not contain enough information for classification. As shown by Glover et al. [76], "anchortext alone is not significantly better (arguably worse) than using the full-text alone." Using surrounding text of anchor text (including the anchor text itself), as in [76], or using anchor text indirectly, as in [44], can address this problem. For carefully created pages, information in titles and headers is generally more important than the generic text. Therefore, it is reasonable to use titles and anchors instead of the full content of neighboring pages, or to emphasize them when using full content. Compared

with using labels of neighbors, using partial content of neighboring pages does not rely on the presence of human labeled pages in the neighborhood. The benefit of using such partial content, however, partially relies on the quality of the linked pages.

Among the three types of features, using the full content of neighboring pages is the most expensive; however it may generate better accuracy. Oh et al. [141] showed that using the class of neighbors provides a 12% gain in F-measure over the approach which only considers on-page content. They also showed that including content of neighbors can increase F-measure by another 1.5%. Qi and Davison [151] demonstrated that additionally using the topical labeling of neighboring page content outperforms two other approaches which only use the human-generated class labels of neighbors (see Chapter 3 for details). In addition, utilizing field information in web pages can bring further improvement [152] (see Chapter 4 for details).

The approaches that utilize features of neighbors are compared in Table 2.1. From the table, we can see that class label is a frequently-used feature. Interestingly, anchortext is less frequently used than one would expect given its apparent descriptive power.

Utilizing artificial links. Although hyperlinks are the most straightforward type of connection between web pages, it is not the only choice. One might also ask which pages should be connected/linked (even if not linked presently). While simple textual similarity might be a reasonable start, a stronger measure is to consider pages that co-occur in top query results [63, 12, 74, 197, 218, 53]. In this model, two pages are judged to be similar by a search engine in a particular context, and would generally include

Approach	Assumption based upon	Types of neighbors used	On-page features utilized?	Types of features used	Combination method
Chakrabarti et al. [29]	weak	sibling	no	label	N/A
Attardi et al. [8]	weak	parent	no	text	N/A
Fürnkranz [64]	weak	parent	no	anchor text, extended anchor text, & headings	multiple voting schemes
Slattery & Mitchell [177]	weak	sibling	no	label	N/A
Fürnkranz [65]	weak	parent	no	anchor text, extended anchor text, & headings	multiple voting schemes
Glover et al. [76]	weak	parent	yes	extended anchor text	N/A
Sun et al. [183]	weak	parent	yes	anchor text, plus text & title of target page	using them as separate features
Cohen [44]	weak	parent	no	text & anchor	N/A
Calado et al. [24]	strong	all six types types within two steps	yes	label	Bayesian Network
Angelova & Weikum [5]	weak	"reliable" neighbors in a local graph	yes	text & label	N/A
Qi & Davison [151]	strong	parent, child, sibling, spouse	yes	text & label	weighted average

Table 2.1: Comparison of approaches using features of neighbors.

pages that contain similar text and similar importance (so that they both rank high in a query). Based on the idea of utilizing information in queries and results, Shen et al. [176] suggested an approach to creating connections between pages that appear in the results of the same query and are both clicked by users, which they term "implicit links". Thus, they utilize similarity as formed by the ranking algorithm, but also by human insight. Their comparison between implicit links and explicit links (hyperlinks) showed that implicit links can help web page classification. A similar approach which classifies web pages by utilizing the interrelationships between web pages and queries was proposed by Xue et al. [206]. The main idea is to iteratively propagate the category information of one type of object (pages or queries) to related objects. This approach showed an improvement of 26% in F-measure over content-based web page classification. In addition to web page classification, artificial connections built upon query results and query logs can be used for query classification.

Links derived from textual similarities can also be useful. Based on a set of feature vectors generated from web directories, Gabrilovich and Markovitch [67] proposed to use feature vectors that are similar enough to the content of the target document to help classification, although such links are not explicitly generated. Given a web directory such as dmoz or Yahoo!, a feature vector is generated for each node (category) by selecting highly representative terms in the category description, URL, and the web sites within that category. After that, the feature generator compares the input text with the feature vectors of all the directory nodes, and vectors that are similar enough are chosen to enrich the

bag-of-words representation of the input page's text. Their other approaches that utilize external knowledge (such as Wikipedia[4] and ODP) for automatic feature generation are described in [68, 69].

There are other methods to generate such artificial links but have not been tested with respect to web classification, such as the "generation links" proposed by Kurland and Lee [109, 110], in which links are created between documents if the language model induced from one document assigns high probability to another. Another example is the links proposed by Luxenburger and Weikum [120], which are generated through high order links within query logs and content similarity.

Discussion: features

On-page features directly reside on the page to be classified. Methods to extract on-page features are fairly well-developed and relatively inexpensive to extract. While obtaining features of neighbors is computationally more expensive (particularly for researchers not within search engine companies), these features provide additional information that cannot be obtained otherwise. When designing a classifier, a decision needs to be made regarding the the trade-off between accuracy and efficiency. Yang et al. [215] compared the computational complexity of several popular text categorization algorithms by means of formal mathematical analysis and experiments. However, most work on web specific classification lacks an analysis of computational complexity, which makes an implementer's

[4]Wikipedia: `http://wikipedia.org/`

decision more difficult.

A comparison of many of the approaches reviewed in this section across several characteristics is shown in Table 2.2. In order to provide a rough idea of performance for each approach, the baseline classifier with which the approach is compared and its reported improvement over the baseline is listed in the table. However, since the results of different approaches are based on different implementations and different datasets, the performance comparison provided here should only serve as a start of a comprehensive evaluation. From Table 2.2, we can see that

- web classification techniques achieve promising performance, although there appears to be room for improvement;
- bag-of-words and set-of-words are popular document representations; and
- existing approaches are evaluated on a wide variety of metrics and datasets, making it difficult to compare their performance.

Although the benefit of utilizing features of neighbors has been shown in many papers, little work has been performed to analyze the underlying reason why such features are useful. In general, features of neighbors provide an alternative view of a web page, which supplements the view from on-page features. Therefore, collectively considering both can help reduce classification error. Jensen et al. [96] investigated the underlying mechanism of collective inference, and argued that the benefit does not only come from a larger feature space, but from modeling dependencies among neighbors and utilizing known class labels.

Approach	Task	Baseline classifier	Reported improv.	Document represent.	Feature selection criteria	Evaluation dataset
Chakrabarti et al. [29]	Topical	A term-based classifier built on TAPER	From 32% to 75% (accuracy)	Bag-of-words	A score-based function derived from Duda & Hart [57]	Yahoo directory
Mladenic [133]	Topical	N/A	N/A	N-gram	Gram frequency	Yahoo directory
Fürnkranz [64]	Functional	A text classifier	From 70.7% to 86.6% (accuracy)	Set-of-words	Entropy (for baseline classifier)	WebKB
Slattery & Mitchell [177]	Functional	N/A	N/A	Relations	No feature selection	WebKB
Kwon & Lee [112]	Topical	A kNN classifier with traditional cosine similarity measure	From 18.2% to 19.2% (micro-averaging breakeven point)	Bag-of-words	Expected mutual information and mutual information	Hanmir
Fürnkranz [65]	Functional	A text classifier	From 70.7% to 86.9% (accuracy)	Set-of-words	Entropy (for baseline classifier)	WebKB
Sun et al. [183]	Functional	A text classifier	From 0.488 to 0.757 (F-measure)	Set-of-words	No feature selection	WebKB
Cohen [44]	Topical	A simple bag-of-words classifier	From 91.6% to 96.4% (accuracy)	Bag-of-words	No feature selection	A custom crawl on nine company web sites
Calado et al. [24]	Topical	A kNN textual classifier	From 39.5% to 81.6% (accuracy)	Bag-of-words	Information gain	Cade directory
Golub & Ardi [78]	Topical	N/A	N/A	Word/phrase	No feature selection	Engineering Electric Library
Qi & Davison [151]	Topical	An SVM textual classifier	From 73.1% to 91.4% (accuracy)	Bag-of-words	No feature selection	ODP directory

Table 2.2: Comparison of web page classification approaches.

Such explanations may also apply to why web page classification benefits from utilizing features of neighbors.

Sibling pages are even more useful than parents and children. We speculate that the reason may lie in the process of hyperlink creation. When linking to other pages, authors of web pages often tend to link to pages with related (but not the same) topics of the current page. As a result, this page, as a whole, may not be an accurate description of its outgoing links. The outgoing links, however, are usually on the same topic, especially those links adjacent to each other. In other words, a page often acts as a bridge to connect its outgoing links, which are likely to have common topics. Therefore, sibling pages are more useful in classification than parent and child pages.

2.1.5 Algorithms

The types of features used in web page classification have been reviewed in the previous section. Here, we focus on the algorithmic approaches.

Dimension reduction

Besides deciding which types of features to use, the weighting of features also plays an important role in classification. Emphasizing features that have better discriminative power will usually boost classification. Feature selection can be seen as a special case of feature weighting, in which features that are eliminated are assigned zero weight. Feature selection reduces the dimensionality of the feature space, which leads to a reduction in

computational complexity. Furthermore, in some cases, classification can be more accurate in the reduced space. A review of traditional feature selection techniques used in text classification can be found in [213].

There are a variety of measures to select features. Some simple approaches have been proven effective. For example, Shanks and Williams [174] showed that only using the first fragment of each document offers fast and accurate classification of news articles. This approach is based on an assumption that a summary is present at the beginning of each document, which is usually true for news articles, but does not always hold for other kinds of documents. However, this approach was later applied to hierarchical classification of web pages by Wibowo and Williams [198], and was shown to be useful for web documents.

Besides these simple measures, there have been a number of feature selection approaches developed in text categorization, such as information gain, mutual information, document frequency and the χ^2-test. These approaches can also be useful for web classification. Kwon and Lee [112] proposed an approach based on a variation of the k-Nearest Neighbor algorithm, in which features are selected using two well-known metrics: expected mutual information and mutual information. They also weighted terms according to the HTML tags in which the term appears, i.e., terms within different tags bear different importance. Calado et al. [24] used information gain, another well-known metric, to select the features to be used. However, based on existing research, it is not clear to what extent feature selection and feature weighting contributed to the improvement. Yan et al. [209] proposed a novel feature selection approach which is more efficient and effective

than information gain and χ^2-test on large-scale datasets.

In text categorization, there is a class of problems in which categories can be distinguished by a small number of features while a large number of other features only add little additional differentiation power. Gabrilovich and Markovitch [66] studied such types of classification problems and showed that the performance of SVM classifiers can be improved in such problems by aggressive feature selection. They also developed a measure that is able to predict the effectiveness of feature selection without training and testing classifiers.

Unlike the above approaches which explicitly select a subset of features, another type of approach decomposes the original document-feature matrix into smaller matrices, which effectively transforms the original feature space into a smaller but less intuitive space. Latent Semantic Indexing (LSI) [54] is a popular approach in this category. However, its high computational complexity makes it inefficient to scale. Therefore, research experiments utilizing LSI in web classification (e.g., [219, 161]) are based on small datasets. Some work has improved upon LSI (e.g., probabilistic [92, 91]) and made it more applicable to large datasets. Research has demonstrated the effectiveness of such improvements [45, 61]. Their efficiency on large datasets, however, needs further study. A matrix factorization approach similar to LSI was proposed by Zhu et al. [223], which jointly factorizes link matrix and content matrix to compute a transformed document representation. Their experiments showed improvement over traditional classification algorithms.

Relational learning

Since web pages can be considered as instances which are connected by hyperlink relations, web page classification can be solved as a relational learning problem, which is a popular research topic in machine learning. Therefore, it makes sense to apply relational learning algorithms to web page classification. Relaxation labeling is one of the algorithms that works well in web classification.

Relaxation labeling was originally proposed as a procedure in image analysis [165]. Later, it became widely used in image and vision analysis, artificial intelligence, pattern recognition, and web mining. "In the context of hypertext classification, the relaxation labeling algorithm first uses a text classifier to assign class probabilities to each node (page). Then it considers each page in turn and reevaluates its class probabilities in light of the latest estimates of the class probabilities of its neighbors" [28].

Relaxation labeling is effective in web page classification [29, 119, 5]. Based on a new framework for modeling link distribution through link statistics, Lu and Getoor [119] proposed a variation of relaxation labeling, in which a combined logistic classifier is used based on content and link information. This approach not only showed improvement over a textual classifier, but also outperformed a single flat classifier based on both content and link features. In another variation proposed by Angelova and Weikum [5], not all neighbors are considered. Instead, only neighbors that are similar enough in content are used.

Besides relaxation labeling, other relational learning algorithms can also be applied

to web classification. Sen and Getoor [173] compared and analyzed relaxation labeling along with two other popular link-based classification algorithms: loopy belief propagation and iterative classification. Their performance on a web collection is better than textual classifiers. Macskassy and Provost [121] implemented a toolkit for classifying networked data, which utilized a collective inference procedure [96], and demonstrated its powerful performance on several datasets including web collections. Unlike others, Zhang et al. [221] proposed a novel approach to relational learning based on both local text and link graph, and showed improved accuracy.

2.1.6 Modifications to traditional algorithms

Besides feature selection and feature weighting, efforts have also been made to tweak traditional algorithms, such as k-Nearest Neighbor and Support Vector Machine (SVM), in the context of web classification.

K-Nearest Neighbor classifiers require a document dissimilarity measure to quantify the distance between a test document and each training document. Most existing kNN classifiers use cosine similarity or inner product. Based on the observation that such measures cannot take advantage of the association between terms, Kwon and Lee [112, 113] developed an improved similarity measure that takes into account the term co-occurrence in documents. The intuition is that frequently co-occurring terms constrain the semantic concept of each other. The more co-occurred terms two documents have in common, the stronger the relationship between the two documents. Their experiments showed

performance improvements of the new similarity measure over cosine similarity and inner product measures. Gövert et al. [79] reinterpreted the k-Nearest Neighbor algorithm with probability computation. In this probabilistic kNN, the probability of a document d being in class c is determined by its distance between its neighbors and itself and its neighbors' probability of being in class c.

Centroid-based classifiers are widely used in web applications because of their high efficiency. The class centroids can be easily computed and incrementally updated. However, they are usually inferior in terms of quality performance measures. Guan et al. [80] proposed a new centroid representation called "Class Feature Centroid", using a combination of inter-class term distribution and inner-class term distribution. Their experiments showed that the proposed method can outperform traditional centroid-based classifiers and even SVM classifiers.

Most supervised learning approaches only learn from training examples. Co-training, introduced by Blum and Mitchell [18], is an approach that makes use of both labeled and unlabeled data to achieve better accuracy. In a binary classification scenario, two classifiers that are trained on different sets of features are used to classify the unlabeled instances. The prediction of each classifier is used to train the other. Compared with the approach which only uses the labeled data, this co-training approach is able to cut the error rate by half. Ghani [71, 72] generalized this approach to multi-class problems. The results showed that co-training does not improve accuracy when there are a large number of categories. On the other hand, their proposed method which combines error-correcting output coding

(a technique to improve multi-class classification performance by using more than enough classifiers, see [55] for details) with co-training is able to boost performance. (See [186] for another example of using error-correcting output coding in text classification.) Park and Zhang [143] also applied co-training in web page classification which considers both content and syntactic information.

Classification usually requires manually labeled positive and negative examples. Yu et al. [217] devised an SVM-based approach to eliminate the need for manual collection of negative examples while still retaining similar classification accuracy. Given positive data and unlabeled data, their algorithm is able to identify the most important positive features. Using these positive features, it filters out possible positive examples from the unlabeled data, which leaves only negative examples. An SVM classifier could then be trained on the labeled positive examples and the filtered negative examples.

Cost-sensitive learning is not a new topic; however, there are interesting facets when performing cost-sensitive learning on the Web. Liu et al. [118] argued that when training models on web pages, popular or important web pages should be given more weight. They modified the SVM algorithm to assign higher weight to relatively high PageRank pages, and showed improvement over the default SVM.

Combining information from multiple sources

Methods utilizing different sources of information can be combined to achieve further improvement, especially when the information considered is orthogonal. In web classification,

combining link and content information is quite popular [24, 151].

A common way to combine multiple sources of information is to treat information from different sources as different (usually disjoint) feature sets, on which multiple classifiers are trained. After that, these classifiers are combined together to generate the final decision. There are various methods to combine such classifiers [108], including well-developed methods in machine learning such as voting and stacking [200]. Combining SVM kernels derived from different sources is another viable method. Joachims et al. [98] showed that the combined kernels of text and co-citation among web pages could perform better than each kernel individually. A similar combining approach is used in [203]. Besides these combining methods, co-training, as discussed previously in Section 2.1.6, is also effective in combining multiple sources since different classifiers are usually trained on disjoint feature sets.

Based on the assumption that each source of information provides a different viewpoint, a combination has the potential to have better knowledge than any single method. However, it often has the disadvantage of additional resource requirements. Moreover, the combination of two does not *always* perform better than each separately. For example, Calado et al. [24] show that the combination of a bibliographic coupling similarity measure with a kNN content-based classifier is worse than kNN alone.

2.1.7 Other issues

Previously we discussed the two important factors in classification: features and algorithms. Other related issues, such as web page preprocessing and dataset selection, also have an effect on classification. We cover such issues here.

Web page content preprocessing

In most experimental work reviewed in this section, preprocessing is performed before the content of web pages is fed into a classifier. HTML tags are usually eliminated. However, the content of meta keyword, meta description and "ALT" fields of image tags is usually preserved. Although stemming may be used in web search, it is rarely utilized in classification. The intuition is that stemming is used in indexing mainly in order to improve recall, while in the scenario of web classification, given enough training instances, different forms of a particular term will appear if the term is important.

Dataset selection and generation

Since web page classification is usually posed as a supervised learning problem, it requires the presence of labeled training instances. In addition, test instances also need to be labeled for the purpose of evaluation.

Since manual labeling can require an excessive amount of human effort, many researchers use subsets of the existing web directories that are publicly available. The two most frequently used web directories are the Yahoo! directory and the dmoz ODP. Others

include Cade directory[5] (a Brazilian web directory, now merged with Yahoo!), HanMir[6] (a Korean web directory), and Engineering Electric Library[7] (EEL, a directory providing engineering information).

The use of different datasets makes it difficult to compare performance across multiple algorithms. Therefore, the web classification research community would benefit from a standard dataset for web page classification, such as the TREC datasets [137] for information retrieval. Although the "WebKB" dataset [46] is one such dataset, it is small and provides only limited functional classes.

In addition to using web directories as evaluation dataset, one might consider artificially generated datasets. Davidov et al. [51] proposed a method that can automatically generate labeled datasets for text categorization with desired properties, e.g., classification difficulty. Based on existing hierarchical web directories such as ODP, this algorithm first finds categories with appropriate degree of classification difficulties based on certain distance measures (e.g., tree distance, textual feature dissimilarity), then crawls web sites under those categories and filters out noise. Although the generated dataset may potentially inherit the bias coming from the web directory on which it is based (such as toward older or high-quality pages), this approach practically eliminates the need for human effort in the generation of some kinds of datasets, while providing flexibility for users to control the characteristics of the generated dataset. Unfortunately, this approach does not provide

[5]Cade: `http://br.cade.yahoo.com/`

[6]HanMir: `http://hanmir.com/`

[7]EEL: `http://eels.lub.lu.se/`

datasets containing an appropriate web structure that more recent neighborhood-based classification approaches would require.

Web site classification

Web *sites* (rather than pages) can also be classified. One branch of research only uses a web site's content. Pierre [147] proposed an approach to the classification of web sites into industry categories utilizing HTML tags.

Another branch focuses on utilizing the structural properties of web sites. It has been shown that there is close correlation between a web site's link structure and its functionality. Amitay et al. [3] used structural information of a web site to determine its functionality (such as search engines, web directories, corporate sites). Motivated by the same intuition, Lindemann and Littig [116] further analyzed the relation between structure and functionality of web sites.

There is also research that utilizes both structural and content information. Ester et al. [60] investigated three different approaches to determining the topical category of a web site based on different web site representations. In their algorithms, a web site can be represented by a single virtual page consisting of all pages in the site, by a vector of topic frequencies, or by a tree of its pages with topics. Experiments showed that the tree classification approach offers the best accuracy. Tian et al. [190] proposed to represent a web site by a two-layer tree model in which each page is modeled by a DOM (Document Object Model) tree and a site is represented by a hierarchical tree constructed

according to the links among pages. Then a Hidden-Markov-Tree-based classifier is used for classification.

Classification of web pages is helpful for classifying a web site. For example, in [60], knowing the topic of the pages in a site can help determine the web site's topic. Presumably, site categorization could also benefit web page classification but the results of such an approach have not been reported.

Blog classification

The word "blog" was originally a short form of "web log", which, as defined by the Merriam-Webster Dictionary[8], is a web site that contains an online personal journal with reflections, comments, and often hyperlinks provided by the writer. As blogging has gained in popularity in recent years, an increasing amount of research about blogs has also been conducted. Research in blog classification can be broken into three types: blog identification (to determine whether a web document is a blog), mood classification, and genre classification.

Research in the first category aims at identifying blog pages from a collection of web pages, which is essentially a binary classification of blog and non-blog. Nanno et al. [135] presented a system that automatically collects and monitors blog collections, identifying blog pages based on a number of simple heuristics. Elgersma and Rijke [59] examined

[8]Merriam-Webster Dictionary: `http://mw1.merriam-webster.com/dictionary/blog`

the effectiveness of common classification algorithms on blog identification tasks. Using a number of human-selected features (some of which are blog specific, e.g., whether characteristic terms are present, such as "Comments" and "Archives"), they found that many off-the-shelf machine learning algorithms can yield satisfactory classification accuracy (around 90%).

The second category of research includes identification of the topic, mood or sentiment of blogs. Most existing approaches in this category recognize mood at the level of individual post; some target recognition of mood reflected by a collection of posts. Mihalcea and Liu [127] showed that blog entries expressing the two polarities of moods, happiness and sadness, are separable by their linguistic content. A naive Bayes classifier trained on uni-gram features achieved 79% accuracy over 10,000 mood-annotated blogposts. Similarly, Chesley et al. [38] demonstrated encouraging performance in categorizing blog posts into three sentiment classes (Objective, Positive, and Negative). However, real-world blog posts indicate moods much more complicated than merely happiness and sadness (or positive and negative). Mishne [129] showed that classifying blog posts into a more comprehensive set of moods is a challenging task (for both machine and human). When doing binary classification of blog posts on more than a hundred predefined moods, SVM classifiers trained on a variety of features (content and non-content) made small (while consistent) improvement (8%) over random guess. Using a similar approach, Leshed and Kaye [114] achieved 76% overall accuracy when classifying into 50 most frequent moods. While recognizing mood for individual blog posts can be difficult, later work done by

Mishne and Rijke [130] showed that determining aggregate mood across a large collection of blog posts can achieve a high accuracy. Many blog systems allow authors and users of blogs to assign arbitrary tags to blog posts. These tags can be considered as additional information for classification. Based on empirical study, Berendt and Hanser [17] drew the conclusion that "tags are not meta data, but just more content - to some people." Tags are shown to be more effective features in classification than blog titles and descriptions [184]. Besides being used as features for classification, tags can also be seen as the task for classification. Research has shown that tags can be predicted with reasonable accuracies [95, 196].

The third category focuses on the genre of blogs. Research in this category is usually done at blog level. Nowson [139] discussed the distinction of three types of blogs: news, commentary, and journal. Qu et al. [157] proposed an approach to automatic classification of blogs into four genres: personal diary, news, political, and sports. Using unigram tfidf [167] document representation and naive Bayes classification, Qu et al.'s approach can achieve an accuracy of 84%.

2.1.8 Summary

Web page classification is a type of supervised learning problem that aims to categorize web pages into a set of predefined categories based on labeled training data. Classification tasks include assigning documents to categories on the basis of subject, function, sentiment, genre, and more. Unlike more general text classification, web page classification methods

can take advantage of the semi-structured content and connections to other pages within the Web.

We have surveyed the space of published approaches to web page classification from various viewpoints, and summarized their findings and contributions, with a special emphasis on the utilization and benefits of web-specific features and methods.

We found that while the appropriate use of textual and visual features that reside directly on the page can improve classification performance, features from neighboring pages provide significant supplementary information to the page being classified. Feature selection and the combination of multiple techniques can bring further improvement.

We expect that future web classification efforts will certainly combine content and link information in some form. In the context of the research surveyed here, future work would be well-advised to:

- Emphasize text and labels from siblings (co-cited pages) over other types of neighbors;
- Incorporate anchortext from parents; and,
- Utilize other sources of (implicit or explicit) human knowledge, such as query logs and click-through behavior, in addition to existing labels to guide classifier creation.

2.2 Taxonomies and Hierarchical Classification

We first review existing approaches related to the general problem of assisting data browsing, then discuss approaches for taxonomy generation and hierarchical classification.

2.2.1 Approaches to assist data browsing

How to organize data and present it to users in a meaningful way has been a difficult problem. We briefly review three mainstream methods for dealing with the issue: taxonomies, faceted search/browsing, and tagging systems.

Taxonomies have been shown to be useful in organizing information across various domains, such as organism classification, library classification, disease classification, and web hierarchies. The traditional taxonomies require concepts to be organized in a strict tree structure. Krishnapuram and Kummamuru [106] reviewed existing approaches to automatic taxonomy generation, analyzed the issues with traditional taxonomies, and pointed out their major problem: "most real-world concepts (and subconcepts) are not crisp" in that they usually belong to more than one category. Therefore, it is unnatural to constrain each concept to have exactly one parent. Based on such an analysis, they proposed a new notion – "fuzzy hierarchies", in which "each concept at level k of the hierarchy is a child of all the concepts at level $k-1$, albeit to different degrees". Compared with traditional taxonomies, fuzzy hierarchies may better match the fuzzy representation in the human mind. However, they still have strict levels; i.e., users still need to make a fixed sequence of choices to reach a particular leaf. Furthermore, no method has been

proposed to generate such a hierarchy.

Ranganathan developed a faceted library classification system called Colon Classification [158, 159], which uses colons to separate identifiers of facets. In this classification scheme, an object is described using a pre-defined set of facets. The introduction of facets into classification systems eliminated the intrinsic restrictions of taxonomies. The same idea powers the modern faceted search/browsing systems, which allows users to filter information from a number of facets in any arbitrary order, as opposed to following pre-defined paths in taxonomies.

Tagging systems became popular along with the rise of social media/social networks. Typical examples of such systems include delicious[9], flickr[10], and bibsonomy[11]. They allow users to assign arbitrary tags to the objects (web pages, photos, scientific publications, etc.), and retrieve objects that are associated with any particular tag. Some also allow sharing of such tags among users. Among the above three methods, the tagging system is the least organized in that it doesn't impose any structural relationship among tags. Such a property makes it easier to assign tags. However, better organized tags (e.g., a tag hierarchy) may enable more effective exploration of information.

[9] http://delicious.com/

[10] http://www.flickr.com/

[11] http://www.bibsonomy.org/

2.2.2 Taxonomy generation

We have briefly reviewed the strength and weakness of taxonomies, faceted search/browsing and tagging systems. Now, we focus on the automatic generation of taxonomies.

A taxonomy is an efficient way to assist users browsing through data collections. It can also be used in classifier training and testing [29, 64, 177, 152], in automatic generation of evaluation datasets for classification approaches [51], in automatic evaluation of search results [13, 14], and in word sense disambiguation [160]. Here, we review existing automatic taxonomy generation approaches in three categories: approaches based on hierarchical clustering using textual content, on associated objects, and on term relationships.

Among the three types of approaches, hierarchical clustering is the most studied. The objects to be clustered range from short queries to whole documents, or sometimes even a collection of documents.

Hierarchical clustering is an effective approach used in systems that assist users to interactively browse through collections of documents. As a well-known example of such systems, Scatter/Gather [47] clusters the documents into a number of clusters, and presents the clusters with short summaries to the user. The user selects a subset of clusters of his interest. The system then re-clusters the selected subset of documents, and presents to the user the refined clusters. This iterative process can be repeated until the user finds what he needs.

The above interactive browsing-and-recluster approach is effective on small and

medium datasets. On large datasets, however, the high online cost makes it less effective. Therefore, many other taxonomy generation approaches are designed mainly as offline systems. The TaxGen System developed at IBM Germany [134] utilizes bottom-up hierarchical clustering methods to build taxonomies out of text collections. The system is able to expand the generated hierarchies by including new documents using automatic categorization approaches. Based on probabilistic document clustering, the "Cluster-Abstraction Model" proposed by Hofmann [90] is able to generate document hierarchies and abstract representation at the same time. The abstractions at each node of the hierarchy are represented by a set of representative terms.

Clustering query results can help users find information faster. In a query result clustering approach based on page titles and snippets proposed by Kummamuru et al. [107], taxonomies are generated by considering document coverage, compactness of the generated hierarchy, and distinctiveness among sibling nodes. Their experiments showed that the proposed algorithm can effectively facilitate search for information.

Generation of query taxonomies is essentially similar to generation of document taxonomies. Most existing approaches are also based on hierarchical clustering techniques. However, since queries are usually short, the query terms themselves are usually not informative enough for automatic clustering. Therefore, many used query result snippets as additional features for query clustering, and achieved good performance [39, 40, 41, 37].

The second category focuses on generating taxonomies from tagging systems, taking advantage of human assigned tags for objects. As a typical approach in this category,

Heymann and Garcia-Molina [88] proposed to generate taxonomies out of flat tagging systems like `delicious.com`. First, cosine similarities between tags are computed based on the objects they are used to annotate. Then, a graph of tags is generated based on such similarities. After that, starting from the center of the similarity graph, a greedy algorithm selects new edges (and the connected nodes) one by one to add to the taxonomy.

Term co-occurrence is a straight-forward measure for term relatedness, which assumes the asymmetric occurrence relationship between terms indicates their semantic subsumptions. Sanderson and Croft [168] proposed to generate taxonomies from text collections using simple statistics of term co-occurrence. Given a query, a set of keywords are extracted from each result. Then keyword x subsumes y if

$$p(x|y) \geq 0.8 \; and \; p(y|x) < 1. \tag{2.1}$$

This approach was adopted by Clough et al. [43] in their work on automatic image taxonomy generation. A similar approach is also used to extract subsumption term pairs from Flickr tags [169].

An advanced method proposed by Glover et al. [75] identifies hypernym, synonym, and hyponym relationships using the relative distribution of the term frequencies within a class of documents and their frequencies in the whole collection. A term hierarchy can then be built based on the identified relationships. "Formal Concept Analysis" [70] is a data analysis method that can automatically generate an ontology based on the given objects and their properties. It can be adapted to effectively build concept hierarchies based on document collections [42], and are shown to achieve better performance than hierarchical

agglomerative clustering and Bi-Section-KMeans [179] (an iterative clustering algorithm based on KMeans).

Usually, the key to taxonomy generation is how to identify term/concept relationships. WordNet [128] has been shown to be a reliable resource from which such relationships can be mined. Ponzetto and Strube [149] applied filters derived from WordNet to network of Wikipedia categories for extracting a large scale taxonomy containing a large amount of subsumption. Suchanek et al. [180] generated hybrid resource by mapping Wikipedia categories to WordNet. Ponzetto and Navigli [148], presented a method using WordNet subsumption hierarchy to perform category disambiguation, which leads to the integration of Wikipedia and WordNet, in that Wikipedia categories are enriched with accurate sense information from WordNet. Kozareva et al. [224] proposed a weakly supervised algorithm for reading web texts, learning taxonomy terms, and identifying hypernym/hyponym relations, which offers the possibility of automatically generating term taxonomies.

Hyperlink structure within a website was shown to be useful in ranking [205]; it is also useful in taxonomy generation. Yang and Liu [210] proposed a unique approach to construct a topic hierarchy for a web site based on its hyperlink structure. Starting from the hyperlink graph within a website, edge weightings produced by classification methods are combined with graph algorithms to generate the topic hierarchy.

2.2.3 Hierarchical classification

In this subsection, we focus on work on hierarchical classification without changing the hierarchical structure. Kiritchenko [103] provided a detailed review of hierarchical text categorization.

Using classification tasks on web pages, Dumais and Chen [58] demonstrated that hierarchical classification is more efficient and accurate than flat classification. One major challenge in general for classification tasks is data sparsity, in which a category has too few labeled instances for a classifier to learn a reasonable model. This problem is prominent in hierarchical classification at lower levels. For such nodes, McCallum et al. [125] proposed to use information from parent nodes to smooth the estimated parameters. A similar idea is used in "Hierarchical Mixture Model" [192] proposed by Toutanova et al., where a generative model incorporates the term probabilities from all parent classes into the current class. Wibowo and Williams [199] suggested that an instance should be assigned to a higher level category when a lower level classifier is uncertain.

The hierarchical classification approaches mentioned above share a common characteristic: they were posed as meta-classifiers built on top of base classifiers. Since information about the hierarchy is handled by the meta-classifier, the base classifier is not aware of the hierarchical structure. Cai and Hofmann [23] proposed a new approach called hierarchical support vector machines in which the hierarchical structure information is incorporated into the loss function of base classifier (in this case, SVM). This approach can also be applied to a general, multi-label classification.

Scalability and effectiveness of hierarchical classifiers on large-scale hierarchies is always a critical issue on real-world applications. Liu et al. [117] studied this problem both analytically and empirically. They found that although hierarchical classification is better for SVM classifiers compared with flat classification, it decreases classification performance when using k-Nearest Neighbor and naive Bayes classifiers.

Utilizing additional features that are specific to a particular domain can potentially improve classification performance. We've shown many such examples on web classification in Section 2.1 where web-specific features are shown to be useful. Complementary features that are only available in hierarchical classification scenarios are also useful as shown by Bennett and Nguyen [16] using two methods called "refinement" and "refined experts", respectively. "Refinement" enhances the training process by performing cross-validation on the training set and using the predicted labels to filter training data so that it better matches the distribution of test data. In "refined experts", an augmented document representation is generated by including the predicted labels from lower level categories. In their experiments, both methods outperform the typical hierarchical SVM, while "refined experts" yields a better performance.

With regard to the evaluation of hierarchical classification, Sun and Lim [182] proposed measuring the performance of hierarchical classification by the *degree* of misclassification, as opposed to measuring the correctness, considering distance measures between the classifier-assigned class and the true class.

2.2.4 Hierarchy adaptation

A variety of approaches have been proposed for hierarchy generation or adaptation. Some of them aim to better assist human browsing (e.g., [134, 107, 156]). Some are proposed and evaluated for general purposes, rather than accurate classification (e.g., [168, 75, 149]). Here, we only focus on the methods that are solely or partially designed for a better automatic classification. These methods can be categorized into two subcategories: generative approaches and adaptation approaches.

Based on a set of predefined leaf categories and associated documents, a generative approach generates a hierarchy using clustering algorithms according to certain similarity measures. A method using linear discriminant projection to generate hierarchies was proposed by Li et al [115]. In this approach, all documents within the hierarchy are first projected onto a lower dimensional space. Then, the leaf categories are clustered using hierarchical agglomerative clustering to generate the hierarchy. Instead of building the hierarchy bottom-up, Punera et al. [150] proposed a hierarchy generation method using top-down clustering.

Unlike generative approaches, adaptation approaches need an existing hierarchy to start. Such initial hierarchies are usually built by humans, but could also be those built by automated methods. The high level idea shared among this category is to make changes to the existing hierarchy such that classification performed on the adapted hierarchy is more accurate than the original. Peng and Choi [145] proposed an efficient method to classify a web page into a topical hierarchy and automatically expand the hierarchy as new

documents are added. In order to classify search results into a large hierarchy accurately and efficiently, a "deep classification" approach is proposed by pruning the hierarchy into a smaller one before classification is performed [202, 204]. Another adaptation approach called "hierarchy adaptation algorithm" [188] is proposed by Tang et al., in which each node in the hierarchy is checked iteratively, and slight modifications are then made locally to particular nodes. This approach can also be used to model dynamic change of taxonomies [187]. Nitta [138] extended this approach to make it more efficient on large-scale hierarchies. Based on an observation that unnecessarily deep hierarchies usually do not perform well, Malik [124] proposed a method to "flatten" a hierarchy by promoting low level categories up to the k-th level and removing the internal nodes.

In summary, approaches in both categories aim to produce hierarchies that are better for classification. The adaptation approaches usually require a reasonable initial hierarchy. In Chapter 5 of this dissertation, we will propose a hierarchy adaptation method that does not rely on a human built hierarchy, and performs better than existing approaches.

Chapter 3

Web Page Classification Using Neighbor Information

3.1 Introduction

The general problem of text classification is well-studied; a number of classifiers have shown good performance in traditional text classification tasks. However, directly applying textual classifiers to web documents often produces unsatisfying results. Compared to standard text classification, classification of web content is different. First, experiments of traditional text classification are usually performed on "structured corpora with well-controlled authoring styles" [34], while web collections do not have such a property. Second, unlike plain text corpora, web pages no longer rely solely on text to express their meanings. With the popularity of non-textual media such as flash and images, web pages

sometimes do not contain enough textual information for text-based classifiers.

Fortunately, hyperlinks on web pages provide useful clues to classification. As we reviewed in Chapter 2, previous research demonstrated that incorporating link information along with the content of web pages can enhance classification [29, 28, 77]. In Chapters 3 and 4, we propose two methods to enhance topical classification of web pages by utilizing the information from neighboring pages in the hyperlink graph.

The first approach, the Neighboring Algorithm, uses class or topic vector of four types of neighboring pages (parents, children, siblings and spouses) to help the categorization of a web page. Unlike existing work, our method does not rely on the appearance of labeled pages in the neighborhood of the page under scrutiny, and thus has wider applicability. In addition, not only sibling pages but also three other kinds of neighboring pages are taken into consideration.

The second approach, the F-Neighbor Algorithm, is proposed as a fielded extension to the Neighboring Algorithm by borrowing the idea from hypertext retrieval where fielded information is shown to be useful. By utilizing important text fields from text on web pages, F-Neighbor can balance the contribution of different fields, and thus more accurately capture topics of web pages. In this chapter, we focus on the Neighboring Algorithm, leaving F-Neighbor to the following chapter.

The rest of this chapter is organized as follows. In Section 3.2, we introduce the Neighboring Algorithm. In Section 3.3, we show the experimental setup and results. And finally, we summarize the Neighboring Algorithm work in Section 3.4.

3.2 The Neighboring Algorithm

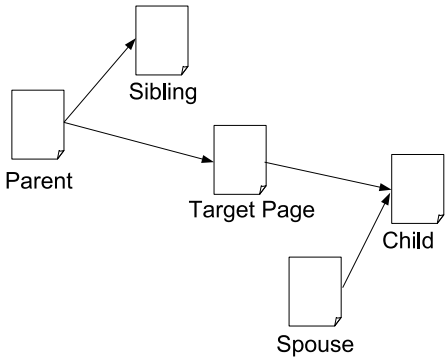
Previous research has shown that utilizing information about neighboring web pages can enhance web page classification. However, it is unlikely that all neighbors provide similar value; a more selective approach may improve performance further. In addition, we would like to know the significance of human-labeled pages in the neighborhood since machine-generated classifications are also possible. We hope that a better understanding of these factors can lead to additional improvements in web content classification.

In the following, the page to be classified is called the "target page", and nearby pages in the link graph are called the "neighboring pages". We will focus on classifying web pages into ODP categories. However, the approach is generic enough to be applied to other taxonomies.

3.2.1 Analyzing the neighborhood

In order to help classify a target page, we use nearby pages with four kinds of relationships to the target: parent pages, child pages, sibling pages and spouse pages (shown in Figure 3.1). These four sets of pages may overlap. In other words, a page may have multiple roles. For example, a page can be both the sibling and spouse of the target page at the same time. In that case, both roles may be of value.

Each of the four sets can be further divided into two subsets: labeled pages and unlabeled pages. Labeled pages are those pages whose categories are already known, such as the pages in the ODP. Unlabeled pages are those whose labels are not known.



Parent pages	$\{ q \mid q \rightarrow p \text{ and } q \neq p \}$
Child pages	$\{ q \mid p \rightarrow q \text{ and } q \neq p \}$
Sibling pages	$\{ q \mid \exists\ r \text{ s.t. } r \rightarrow p,\ r \rightarrow\ q \text{ and } q \neq p \}$
Spouse pages	$\{ q \mid \exists\ r \text{ s.t. } p \rightarrow r,\ q \rightarrow\ r \text{ and } q \neq p \}$

Figure 3.1: Four kinds of neighboring pages of p

Since any classifier produces only an approximation to the desired human labeling, we will generally use the human judgment whenever it is available. Otherwise, a standard text classifier will be used to generate a soft classification. That is, the probabilities of the page being in each category are given as the classification result. Therefore, after classification, each page p is represented by a probability distribution vector $\overrightarrow{v_p} = (v_{p,1}, v_{p,2}, ..., v_{p,i}, ..., v_{p,n})$, in which each component $v_{p,i}$ is the normalized probability of the page being in the corresponding category c_i. This vector is referred to as "topic vector". For unlabeled pages, this vector is generated by the classifier. For labeled pages, this vector is set according to Equation 3.1:

$$v_{p,k} = \begin{cases} 1 & \text{if } C[k] = L[p] \\ 0 & \text{if } C[k] \neq L[p] \end{cases} \tag{3.1}$$

where C is a sorted list of the names of each category, and $L[p]$ is the human labeling of page p. Equation 3.1 assumes every labeled page only belongs to one category. For

pages that are labeled with multiple categories, either an arbitrary decision needs to be made to choose one out of the multiple labels, or the probability is evenly distributed across all the labels. In our experiments, we use the first label in alphabetical order. The soft classification is mainly used for internal representation. Although the output of our algorithm is also in the form of probability distribution, it is converted into hard labeling for the purpose of evaluation, i.e., labeling the page by the class to which it most likely belongs.

The reason why we use soft classification rather than hard labeling is based on observations of real-world pages. First, web pages have complex structures and each part of the page may talk about related but different topics. Second, even for the pages that concentrate on one topic, it is natural that the topic may belong to multiple categories. For example, the homepage of a health insurance company may belong to both "Business" and "Health". Part of the reason for this lies in the ambiguously-defined taxonomy. Some pages in the ODP directory are placed into multiple categories by human editors, which fortifies our confidence in using soft classification.

So far, we have introduced four types of neighboring pages according to their link relationships to the target. They can be further categorized into two subtypes according to the availability of its pre-determined label. Each page of these types can be represented by a topic vector. Next, we discuss how to utilize this information to help classify the target page.

3.2.2 An overview of neighboring algorithm

The neighboring algorithm, as shown in Figure 3.2, consists of four major steps: page level weighting, grouping, group level weighting, and the weighting between the target page and neighbors. First, each neighboring page, represented by its topic vector, is weighted according to its individual properties. Such properties include whether its human-generated label is available, whether it is from the same host as the target page, etc. Then, these pages are grouped into four groups according to their link relationship with the target page, that is, parent, child, sibling, or spouse. At this step, the topic vectors of the pages within the same group are aggregated; and the topic vector of a group is computed as the centroid of the topic vectors of all its members. After grouping, there are four topic vectors, each representing one group. Then, group level weighting combines the four vectors and generates the topic vector for all the neighbors. Finally, the topic vector of neighbors and the topic vector of the target page is combined to generate the final topic vector, based on which a decision of the page's topic is made.

The intuition of the Neighboring Algorithm is to utilize the information from neighbors to adjust the classifier's view of the target page. For example, consider the scenario illustrated in Figure 3.2. The target page was originally classified as "yellow" by a classifier that only considers the page itself. However, the neighbors strongly suggest otherwise. Therefore, by considering the information of neighbors, the algorithm is able to adjust the decision and classify the target page as "red".

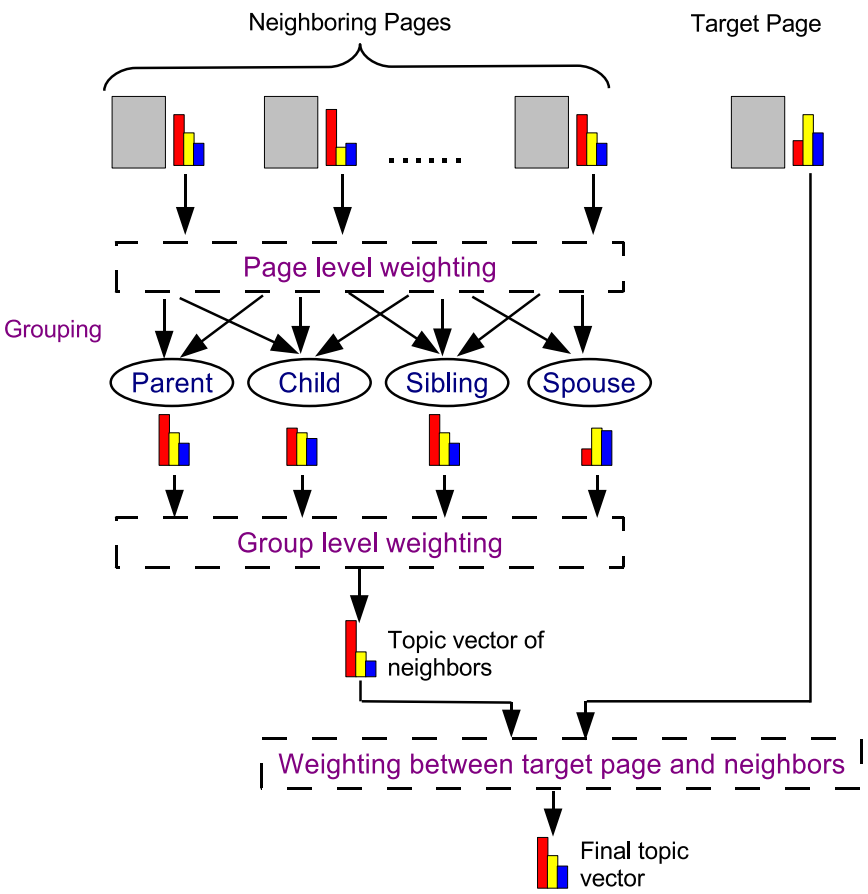


Figure 3.2: Neighboring algorithm overview

3.2.3 Utilizing neighboring information

After analyzing the neighborhood structure, the neighboring pages are placed into one or more of the four sets based on their relationship in the web graph. Each page is represented by a topic vector $\overrightarrow{v_p}$. In the next step, these topic vectors combined with the topic vector of the target page will be used to improve classification performance.

In general, the topic vectors of all neighboring pages may help determine the target page's category. For example, one simple approach would be to set the target page's class

to the majority class of all neighbors. However, in this work we find that different types of pages are of different importance to classification (see Section 3.3.4 for details). Therefore, we introduce weighting factors to bias the influence from different types of neighboring pages.

Page level weighting

Bias on labeled pages. As described in Section 3.2.1, the pages in the neighborhood of the target page may or may not be labeled. The topic vectors of labeled pages are determined by the human labeling, while the topic vectors of unlabeled pages are produced by a classifier. In order to control the noise introduced by the classifier, we use a factor η (where $0 \leq \eta \leq 1$) to down-weight the vectors of unlabeled pages. That is, we modify the topic vector v_p by multiplying it by its weight $w(p)$. The modified topic vector v'_p is computed by Equation 3.2.

$$v'_{p,k} = v_{p,k} \times w(p) \tag{3.2}$$

where $w(p) = \begin{cases} 1 & \text{if } p \text{ is labeled} \\ \eta & \text{if } p \text{ is unlabeled} \end{cases}$

When η=0, the influence coming from those unlabeled pages will be totally ignored, which implies that we don't trust the classifier at all. When η=1, the unlabeled pages are treated equally to the labeled ones, which means we assume the classifier performs as well as human labeling. The value of η will be tuned through experiments.

Intra-host link bias. Hyperlinks connecting pages within the same web site often

serve the purpose of navigation and do not confer authority. Therefore, they are often ignored or considered less useful in the scenario of link-based ranking. However, the situation can be different in web page classification tasks. For example, on a shopping web site, a product list of digital cameras may contain links to all the digital cameras, which are also on the same topic. As a result, we wish to explicitly determine the utility of internal links for web page classification.

In order to find out the answer, we introduce the parameter θ to weight the influence of the neighboring pages that are within the same web host of the target page. We modify the topic vector again to include this parameter.

$$v''_{p,k} = \begin{cases} \theta \cdot v'_{p,k} & host(p) == host(s) \\ (1-\theta) \cdot v'_{p,k} & host(p) \neq host(s) \end{cases} \tag{3.3}$$

where s is the target page and *host()* is a function that returns a page's host name. When θ=0, intra-host links are ignored. When θ=1, inter-host links are ignored.

Counting the multiple paths. Now that we generated and modified the topic vector for each page in the neighborhood, it is time to consider the relationship between the target page and the neighboring pages. Here, an interesting issue may arise: if a sibling page has more than one parent in common with the target page, that is, from a link graph view, there are multiple paths between the target page and its sibling page, should the weight be counted as one or as the number of parents in common? The same question applies to the spouse pages, too. We leave this question to be answered by the experiments.

In the weighted path variation, the influence of a sibling page (or a spouse page) to the target page's topic is weighted by the number of common parents (or children). In the unweighted version, such weighting scheme is ignored. That is, no matter how many parents (or children) they have in common, it is counted only once.

Grouping and group level weighting

In Section 3.2.1, we introduced four types of neighboring pages: parent pages (A), child pages (B), sibling pages (C) and spouse pages (D). We expect that the pages in these four sets may have different influence on the target page's topic. Therefore, a weighting vector $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)$ is used to allow for bias among them, where $\beta_1, \beta_2, \beta_3, \beta_4 \geq 0$ and $\beta_1 + \beta_2 + \beta_3 + \beta_4 = 1$.

The combined neighboring pages' topic vector v_n can be computed by Equation 3.4.

$$\begin{aligned} v_{n,k} = \beta_1 \times \frac{\sum_{p \in A} v''_{p,k}}{\sum_{p \in A} w(p)} + \beta_2 \times \frac{\sum_{p \in B} v''_{p,k}}{\sum_{p \in B} w(p)} \\ + \beta_3 \times \frac{\sum_{p \in C} v''_{p,k}}{\sum_{p \in C} w(p)} + \beta_4 \times \frac{\sum_{p \in D} v''_{p,k}}{\sum_{p \in D} w(p)} \end{aligned} \tag{3.4}$$

The weighted combination formula allows straight-forward tuning of the parameters, and easy identification of relative contribution from different components.

Combining neighboring pages with target page

Like neighboring pages, the target page s will also get its topic vector v_s through a textual classifier. Then the final topic vector v for the target page s will be a weighted combination

of v_s and v_n.

$$v_k = \alpha \times v_{s,k} + (1 - \alpha)v_{n,k} \tag{3.5}$$

or in vector representation:

$$\vec{v} = \alpha \times \vec{v}_s + (1 - \alpha)\vec{v}_n \tag{3.6}$$

where $0 \leq \alpha \leq 1$.

When α=0, the labeling of the target page is solely decided by its neighbors without considering its own content at all. When α=1, the labeling is based solely on the pure textual classifier while the information from the neighboring pages is ignored.

Now that the combined topic vector v is obtained by taking into consideration all the neighboring pages' information as well as the target page, a conversion from probabilistic distribution to hard labeling is needed before evaluation. The conversion simply picks the category corresponding to the largest component in v as the label of the target page.

3.3 Testing the Neighboring Algorithm

So far, we have described a highly parameterized model for generating a topic distribution (and thus implicitly, a hard classification) for a target page, given the labels and textual contents of neighboring pages and their relationships to the target page. In this section, we describe experiments using that model on real-world data to evaluate the impact of various parameters and to assess the potential of a web-page classifier using an appropriately tuned version of the model.

3.3.1 Experimental setup

Taxonomy

We choose to use the classification taxonomy from the Open Directory Project [140]. Constructed and maintained by a large community of volunteer editors, the ODP, also known as the dmoz Directory, is claimed to be the largest human-edited directory of the Web.

The metadata being used in our work was downloaded from dmoz.org in September 2004, and contains 0.6 million categories and 4.4 million web pages. A crawler was used to fetch the web pages pointed to by the ODP, out of which 95% were successfully retrieved.

HTML file preprocessing

All of the web pages we use have gone through a text preprocessor. This includes the pages to train the classifier, as well as the target pages and their neighboring pages which we will use for evaluation.

The functionality of the preprocessor is as follows:

- eliminate HTML tags except the content from the "keywords" and "description" metatags (because they may be of help in deciding a page's topic);
- unescape escaped characters;
- eliminate characters other than alphanumeric characters;
- eliminate terms whose length exceeds a certain limit (4096 characters in this case).

Arts	Business	Computers	Games
Health	Home	Recreation	Reference
Science	Shopping	Society	Sports

Table 3.1: Twelve top-level categories used in the dmoz Directory.

Therefore, after preprocessing, each HTML file is transformed into a stream of terms.

The preprocessing is essential for at least two reasons. First, it filters out noisy terms such as "html", "body", which may compromise the classification accuracy. In our experience, this preprocessing can increase the classification accuracy by 3% (in absolute value). Second, preprocessing eliminates some unnecessary features and thus makes web pages shorter, reducing time and space required by the classifier.

Text-based classifier training

We trained a textual classifier using linear kernel support vector machines with default settings based on SVM^{light} [97]. Since SVM is a binary classifier, we generalize it using one-against-others approach, i.e., to train twelve classifiers each using one class as positive class and the rest as negative classes.

First, as in the work by Chakrabarti et al. [29], we selected 12 out of the 17 top-level categories from the dmoz Directory (listed in Table 3.1). A random selection of 19,000 pages from each of the 12 categories (i.e., 228,000 in total) are used to train the textual classifier. No feature selection was employed (i.e., all features were used). The trained textual classifier is used as a baseline and the base classifier to generate the initial topic vector for the Neighboring Algorithm.

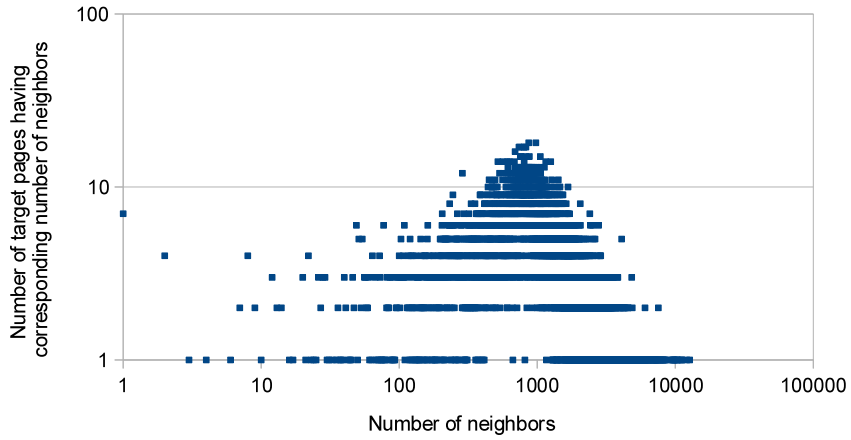


Figure 3.3: Distribution of the number of neighbors per target page.

Tuning set and test set

Two datasets are used in our experiments to measure performance: a sample of 12,000 web pages from ODP and a sample of 2000 web pages from Stanford's WebBase collection [89].

The ODP dataset consists of 1000 pages per category from the 12 top level categories listed in Table 3.1. We randomly split them into five folds and performed cross validation. Each run uses data in four folds to tune the parameters and the remaining fold to evaluate the classification performance. In the following, the reported classification accuracy is averaged over the five folds unless specified otherwise. We obtained the URLs of the neighboring pages and then crawled the union of those pages from the Web. The outgoing links are directly extracted from the web pages, while the incoming links are obtained by querying Yahoo search with "inlink:" queries through the Yahoo API [207]. Due to API usage limits, we obtained at most the first 50 incoming links for each page.

On average, 778 neighbors are retrieved for each target page. The number of neighbors of a target page falls into a wide range from zero to more than 10,000. The distribution is shown in Figure 3.3. The numbers of each individual type of neighbors used in our test are listed in Table 3.2. Although we distinguish the four kinds of neighbors literally, they actually overlap with one another. Therefore, the actual total number of neighboring pages is less than the sum.

For the WebBase dataset, 2000 target pages are selected from a 2005 crawl. The link graph provided with the data collection is used to find the neighboring pages. The use of WebBase dataset has two purposes. First, the ODP pages are mostly high quality pages, while WebBase is a generic crawl from the Web. Therefore, experiments on the WebBase dataset are potentially able to demonstrate performance on more typical web pages rather than just high-quality pages. Second, in the ODP dataset, the number of neighboring pages is limited by the method used to collect incoming links. By using WebBase data, we hope to determine the importance of the role played by the number of incoming links in our algorithm.

Parent pages	518,309
Child pages	260,154
Sibling pages	4,917,296
Spouse pages	3,642,242
Unique neighbors	6,483,871

Table 3.2: Numbers of the four types of neighbors

Removing "dmoz copies"

It is noteworthy to point out that when going through our data set manually, we found that there are plenty of "dmoz copies". A "dmoz copy" is a mirror of a portion of the dmoz ODP. Given that dmoz metadata is publicly available, setting up such a mirror site is straightforward, and not necessarily bad. However, our algorithm may unduly benefit from those copies.

Imagine a page pointed to by dmoz Directory is under scrutiny. By querying for the parents of this page, we may get several or even tens of dmoz copies which link to other pages with the same topic. Since the labels of those sibling pages are known (because they are in dmoz Directory), they are utilized by our algorithm in determining the target page's topic. Furthermore, the existence of "dmoz copies" provides multiple paths between the target page and the sibling pages. Therefore, in the weighted path version, the labels of those labeled sibling pages will probably dominate the contribution from the neighbors and thus boost the accuracy.

In order to minimize the benefit from "dmoz copies", we used a simple pruning method to remove the copies from our data set. The method is based on the observation that most URLs and titles of "dmoz copies" contain the names of directories in dmoz, such as "Computer/Hardware" and "Business/Employment". This program checks the URLs and titles of every neighboring page and removes those whose URL or title contains such directory names. In the ODP dataset, 160,868 pages were found by this pruning step. They are removed for all the experiments. This removal is necessary; a preliminary experiment

shows a 3% drop in accuracy after removal.

The limitation of this approach is obvious. This pruning method may unnecessarily remove pages that are not "dmoz copies". It may also pass by some real "dmoz copies" if they do not use those directory names in their URLs or titles. However, a manual check on a random sample of more than 100 parent pages did not discover any additional "dmoz copies".

3.3.2 Parameter tuning

The parameters introduced in Section 3.2 need to be tuned. We used the standard hill climbing method to find the optimal parameter setting. Starting from a random point in the parameter space, the method searches the surrounding points, finds the best solution within the neighborhood, and makes that point as the starting point of the next iteration. This process iterates until no further improvement can be made. To reduce the chance of being stuck at a local optimum, we let the search processes start from three different points and run independently. Table 3.3 shows the average number of iterations and the best accuracy on the tuning set for each fold. For fold 1, 2, 3, and 4, all hill climbing processes converged at the same setting where α=0.1, β=(0.1, 0.1, 0.7, 0.1), η=0, θ=0.7, using weighted path. For fold 5, two out of the three processes converged at the above setting, the other one converged at a local optimum where α=0.1, β=(0, 0, 0.8, 0.2), η=0, θ=0.7, but the difference in accuracy is merely marginal.

The impact of the individual parameters will be studied in Section 3.3.4. We summarize

some highlights here:

- We should trust the human labeling while ignoring the result of text classifiers;
- The algorithm performs better when using weighted paths than unweighted;
- Intra-host neighbors should not be ignored; emphasizing intra-host neighbors can introduce a slight improvement over the default setting (treating intra-host and inter-host neighbors indifferently);
- Siblings are the most useful among the four neighbor types, while other neighbor types also contribute.

3.3.3 Experimental results

Experiments on the labeled ODP dataset

After tuning the parameter settings, we ran our algorithm on the test set with the setting α=0.1, β=(0.1, 0.1, 0.7, 0.1), η=0, θ=0.7, and using the weighted path version.

For the purpose of comparison, we also implemented one of the algorithms (*IO-bridge*) suggested by Chakrabarti et al. [29] and another algorithm (*K+C*) proposed by Calado et

Fold	Avg. Num. Iter.	Best Accuracy
1	15	0.8899
2	22.67	0.8886
3	16.33	0.8857
4	16.33	0.8872
5	15.67	0.8878
Avg	17.2	0.8878

Table 3.3: Average Number of Iterations

al [24].

The main idea of *IO-bridge* is to build an engineered document corresponding to each document in the dataset, in which the constructed document consists only of prefixes of the category names of the sibling pages of the target page. In *IO-bridge*, only the sibling pages within a human labeled dataset are considered. After that, the training and testing is performed on the engineered dataset rather than the original one.

The best performance reported by Calado et al. was achieved when combining the result of the kNN text classifier with co-citation similarity derived from link graph ($K+C$). In the following, we compare our algorithm with both *IO-bridge* and $K+C$.

The comparison of the algorithms is shown in Figure 3.4. The baseline, textual SVM, has an accuracy of 72.9% averaged across the 5 folds. IO-bridge increases the accuracy to 80.2%. (IO-bridge is reported [29] to have increased the accuracy from 32% to 75% on a 800-document dataset extracted from Yahoo Directory.) $K+C$ has an accuracy of 76.3%. However, if only the link-based measure (co-citation) is considered, its accuracy (86%) is much higher than the combination of link-based and content-based measures. Our algorithm (referred to as "Neighboring" in Figure 3.4), can achieve 88.8% accuracy. T-test shows the Neighboring Algorithm performs statistically significantly better than all other compared algorithms ($p < 10^{-4}$).

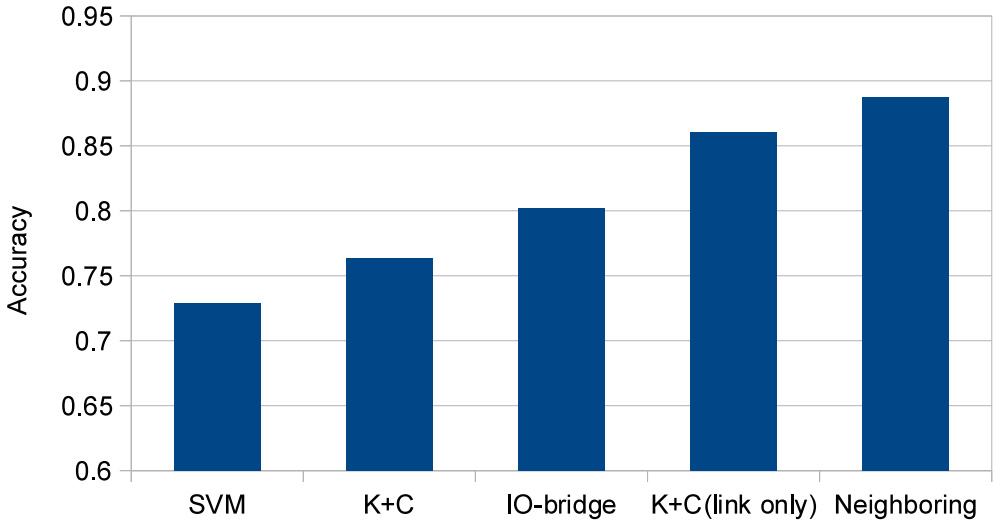


Figure 3.4: Comparison of accuracy of different algorithms

Experiments on the WebBase dataset

To eliminate the potential bias of highly-regarded web pages from the ODP, we tested our approach on randomly selected pages from the WebBase dataset. We continued to use the parameter settings of α=0.1, β=(0.1, 0.1, 0.7, 0.1), η=1, θ=0.7 and the weighted path version. We manually labeled more than 100 randomly selected pages for evaluation purposes. The accuracy of the Neighboring Algorithm is 36.8%, an improvement of more than 20% over the accuracy of the SVM textual classifier (30.4%).

3.3.4 Parameter study

In this subsection, we show how the performance of the algorithm over the tuning set is affected by the value of the parameters. The parameter study in the following is performed

on the first fold of the ODP dataset. On a Xeon 3.0 GHz CPU, the parameter tuning process for one fold takes approximately 27 CPU hours to converge.

η: bias on labeled pages

In order to determine the effect of η (the weight of unlabeled pages), we performed a test by changing the value of η while fixing the values of other parameters. The other parameters are set as follows: θ=0.7, β={0.1, 0.1, 0.7, 0.1}, and α=0.1. As shown in Figure 3.5, the best performance in these tests is achieved when η=0. As we increase the value of η, the accuracy shows a steadily decreasing trend, highlighting the importance of human-labeled neighbors. The result suggests that we should trust the human labeling whenever it is available while ignoring the result of textual classifier. This is understandable given the poor performance of textual classifiers on web data. However, as shown by experiments on WebBase dataset in Section 3.3.3, when human labels are not available, textual classifier is still useful.

Weighted paths vs. unweighted paths

The comparison of the weighted and unweighted version is also shown in Figure 3.5, from which we can see that the weighted version outperforms the unweighted. The explanation of this result is quite straightforward: having more parents (or children) in common implies a stronger connection between the two pages. Therefore, it is natural for the influence between them to be weighted.

We also tested a variation of weighted path in which a neighbor is weighted by a

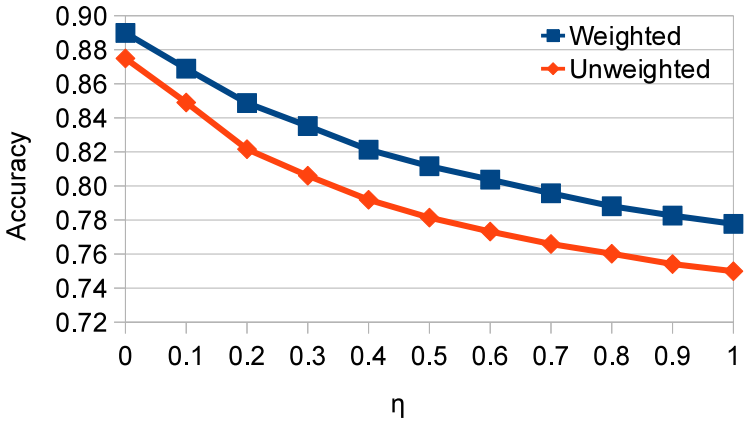


Figure 3.5: Accuracy vs. weight of unlabeled pages (η)

logarithm of the number of paths. The results are worse than when using weighted path.

θ: are internal links useful?

Earlier in this paper we raised the question: are intra-host links useful in web page classification tasks? According to our findings, the answer is a qualified "yes".

We performed additional experiments to see how the accuracy changes when θ varies from 0 to 1. Figure 3.6 shows the test result. Although not remarkably, the weight of intra-host links does influence the accuracy observably. When increasing θ starting from 0, the accuracy climbs up slightly until getting to its peak when θ=0.7. After that, the accuracy decreases. The result suggests that the neighbors within the same host typically have some connection in topic with the target, improving performance slightly when combined with inter-host links. In addition, rather than being weighted less as in link-based ranking algorithms, the results suggest that intra-host links should be given more weight than

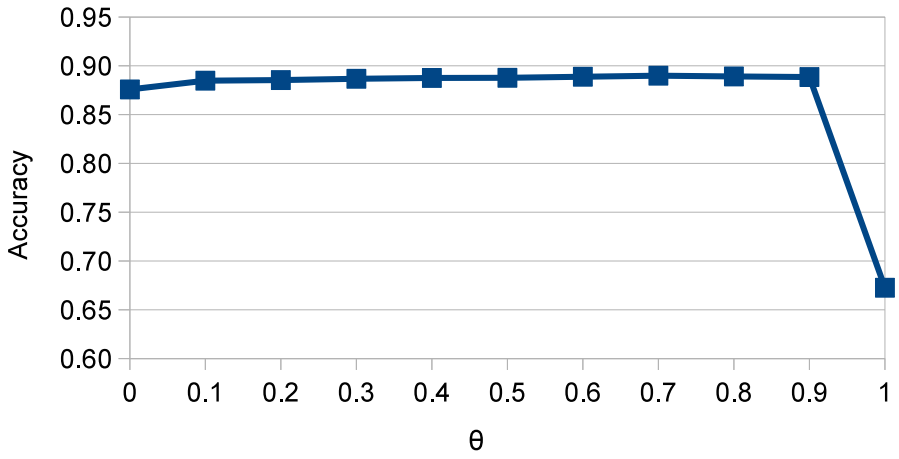


Figure 3.6: Accuracy vs. weight of intra-host links (θ)

inter-host links.

β: weights among neighbors

We expect that different types of neighboring pages have different contributions in predicting the target page's topic. First, we study the individual impact of each type of neighbor. Figure 3.7 shows the individual contribution of each of them, among which sibling pages contribute the most. Spouse pages are the least reliable source of information.

Next, in Figure 3.8, the influence of each kind of neighboring pages is augmented in contrast to the others. For example, in Group A, four tests are performed, each picking one kind of neighbors and setting the corresponding component in β to 0.4 while setting the other three to 0.2. In particular, the "parent" column in Group A shows the accuracy under the setting β= (0.4, 0.2, 0.2, 0.2). Similarly, in Group B, the major component is

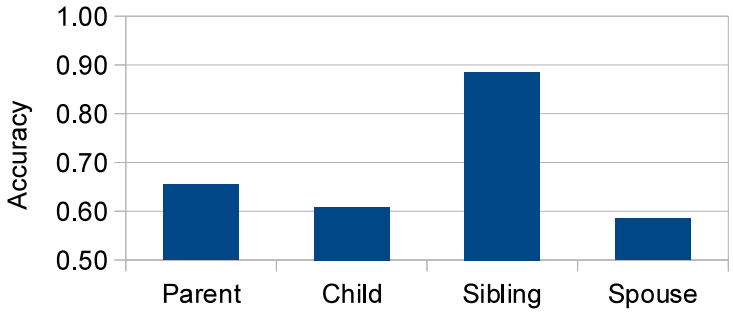


Figure 3.7: Individual contribution of four types of neighbors

0.7 and the rest are set to 0.1.

Figure 3.8 shows that having the sibling pages to make the major contribution is clearly better than any of the other three. However, does that mean we should give full weight to sibling pages?

In order to answer that question, we gradually change the weight of sibling pages from 0 to 1 and let the other three evenly share the remaining weight. The result is plotted in Figure 3.9. As we can see, although siblings are the best source of information, putting excessive weight on siblings will decrease the accuracy.

α: combining the neighbors with the target page

We start by applying a textual classifier to the target page and try to correct the classifier's decision when the neighboring pages strongly indicate otherwise. As is shown in Figure 3.10, the accuracy peaks at α=0.1 which means it is important to emphasize the information from neighboring pages.

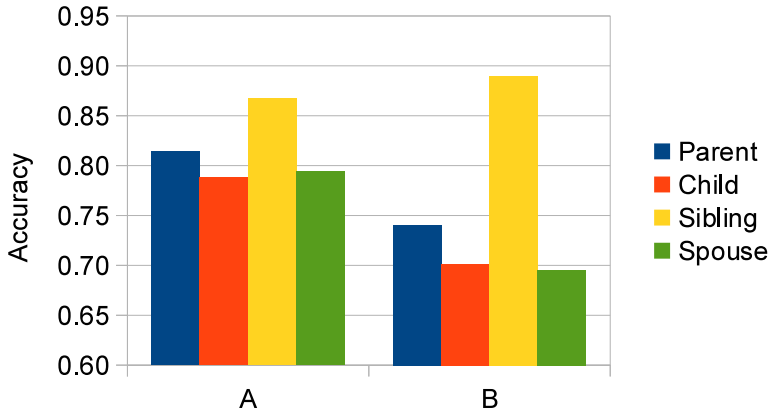


Figure 3.8: Accuracy as principal neighbor type is changed

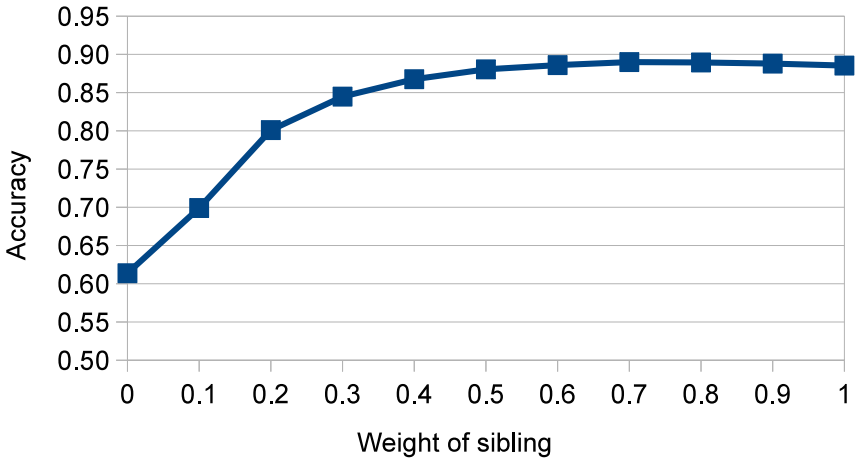


Figure 3.9: Accuracy vs. weight of siblings

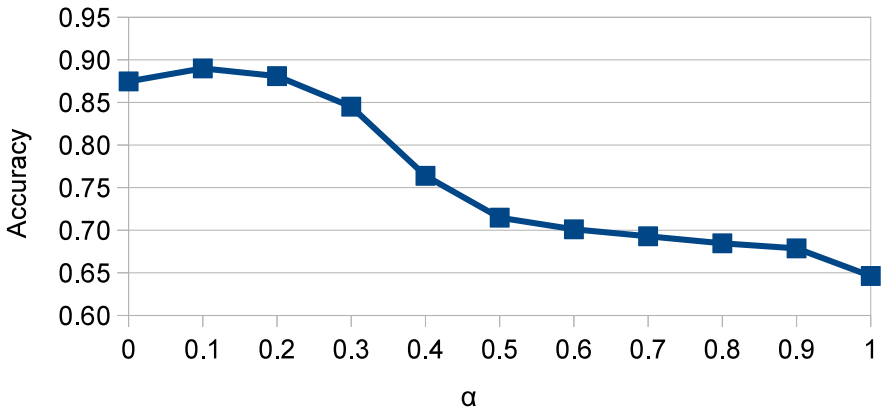


Figure 3.10: Accuracy vs. weight of target page content (α)

Although the result seems to strongly suggest neighboring pages are a better source of information for the target page's topic than the target page itself, we argue that there are at least two more possible reasons which may lead to such a result. First, neighboring pages, greatly outnumbering the target page, provide more features, based on which the classifier is able to collectively make a better decision. Second, some target pages do not have enough textual content for the classifier to use. A definitive explanation will require further study.

3.4 Conclusion

This chapter has explored a method to utilize class information from neighboring pages to help judge the topic of a target web page. The experimental results show that, under appropriate parameter settings, our algorithm statistically significantly outperforms the

SVM textual classifier as well as existing algorithms.

Our contributions in this chapter include the following:

- We tested multiple algorithms on a large, real-world data set.
- We showed greatly improved accuracy on web page classification, reducing error rates by almost two thirds over common text classification approaches.
- We explored the effects that a number of factors have on the classification, and proposed explanations of our findings. We found that sibling pages give a good indication of a page's topic and that intra-host links provide some benefit.
- We are the first to point out the "dmoz copy effect" in web page classification and proposed a way to address it (although this has been raised as a general issue in web link analysis).

The Neighboring Algorithm takes the advantage of labeled web pages and the hyper-linked neighborhood information. On a web page, contents are marked up by different HTML tags. Most web classification approaches, including the Neighboring Algorithm, treat the content indifferently. In the following chapter, we will argue that text from different fields should bear different importance in classification, and then demonstrate that utilizing such field information can further improve the classification performance.

Chapter 4

Web Page Classification Using Fielded Neighbor Information

4.1 Introduction

In 1994, Robertson and Walker [164] proposed a function to rank documents based on the appearance of query terms in those documents, which was later referred to as Okapi BM25. In time, BM25 became a common component of information retrieval systems. A decade later, Robertson et al. [163] extended this model to combine multiple text fields including anchor text, and showed improvement over original BM25. Since then, the fielded BM25 model (BM25F) has become more popular, taking the place of its non-fielded predecessor. Besides retrieval, the fielded model has also shown to be useful in expert finding from email corpora [9, 146]. In this chapter, we borrow the idea that

extended BM25 to BM25F, hoping it can boost classification performance as it did for retrieval.

In the previous chapter, we proposed the Neighboring Algorithm which uses class or topic vectors of four types of neighboring pages (parents, children, siblings and spouses) to help in the categorization of a web page. However, when using these pages, all text on a page are considered equally important. Inspired by the success of utilizing field information in previous examples, we propose the F-Neighbor Algorithm, a fielded extension to Neighboring. By extracting important text fields from generic text on the web pages, F-Neighbor can balance the contribution of different text, and thus more accurately captures topics of web pages. In our experiments on two datasets, F-Neighbor is shown to further improve classification accuracy over the Neighboring Algorithm.

4.2 Approach

The default Neighboring algorithm considers text on each neighboring page as a whole, disregarding where the text appears. Here, we argue that text appearing in different fields may carry different values. For example, anchor text (the text to be clicked on to activate and follow a hyperlink to another web page, placed between HTML <A> and </A> tags) is usually considered a good description of the page to which it points; therefore, anchor text could be more useful in classification than other text on the parent page. As an extension to the Neighboring algorithm, we examine the importance of text in different fields on neighboring pages.

4.2.1 Utilizing text fields

The idea of differentiating text in different fields of hypertext is not new. It has been successfully applied to web information retrieval [163], to web classification [78], as well as other research. However, little research [64, 65, 77] has examined the importance of text fields on neighboring pages in classification problems. Here, we propose to utilize the textual information appearing in the following fields to help classify a web page:

- title of the target page;
- full text of the target page;
- titles of parent, child, sibling, and spouse pages;
- full text of parent, child, sibling, and spouse pages;
- label of parent, child, sibling, and spouse pages;
- anchor text (referring to target) on parent pages;
- surrounding text of anchor text (including anchor text itself) on parent pages (referred to as "extended anchor text", which in our experiments consists of the 50 characters before and after the anchor text).

For each page that needs be classified, these fields are extracted and used for classification. The fields on the pages themselves are extracted directly by parsing the web pages. The labels of neighboring pages can be acquired from human maintained hierarchies such

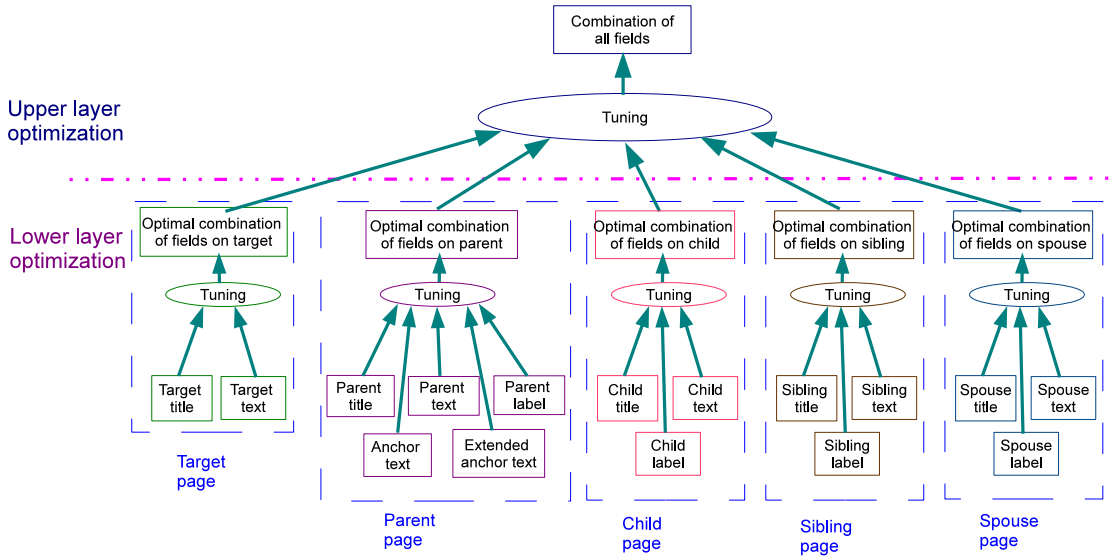


Figure 4.1: The process of two layer optimization.

as dmoz ODP or Yahoo Directory. The label field can be empty if no labels of a particular neighbor type is available. In the experiments, the labels are collected from dmoz. Unlike fields residing on the target page, each type of the text fields from neighboring pages usually has multiple instances. For example, a target page with ten sibling pages has ten instances of "sibling:title" field. These instances are aggregated by computing the centroid of each type of field, as described below.

4.2.2 Text representation

We formalize the computations described above using the following equations. For the fields on the target page, a tfidf representation is computed for each virtual document using the equations used by the Cornell SMART system [166]. The term frequency and inverse document frequency are defined by Equation 4.1 and 4.2, where $n(d,t)$ is the

number of times term t appears in document d, $|D|$ is the the total number of documents in the collection D, and $|D_t|$ is the number of documents that contain term t.

$$TF(d,t) = \begin{cases} 0 & \text{if } n(d,t) = 0 \\ 1 + log(1 + log(n(d,t))) & \text{otherwise} \end{cases} \tag{4.1}$$

$$IDF(t) = log\frac{1+|D|}{|D_t|} \tag{4.2}$$

Each document d is represented by a vector $\vec{d}$ in which each component d_t is its projection on axis t, given by

$$d_t = TF(d,t) \times IDF(t) \tag{4.3}$$

Finally, vector $\vec{d}$ is normalized using Equation 4.4 so that the length of the vector is 1.

$$d'_t = \frac{d_t}{\sqrt{\sum_{s \in T} d_s^2}} \tag{4.4}$$

The vector $\vec{d'}$ computed by Equation 4.4 is used to represent a field from the target page.

Equations 4.1 through 4.4 are also applied to fields of neighboring pages. However, in contrast to the fields on the target page, each type of neighboring field usually has multiple instances coming from different neighbors (as in the previous example, a target page with ten siblings has ten instances of sibling title and ten instances of sibling text). In order to generate a single representation for each type of neighboring field, an additional step is needed. This is performed by simply computing the centroid of the instances of the type. Empty fields (such as an empty title) are not considered in this computation. Note that the tfidf vectors are normalized before combination to prevent a long textual field of one page from dominating the fields from other pages.

$$\vec{d}_{f_i} = \begin{cases} \vec{d'} & \text{if } f_i \text{ is a field on target} \\ \frac{1}{N_{f_i}} \sum_{j=1}^{N_{f_i}} \vec{d'}_j & \text{if } f_i \text{ is a field of neighbors} \end{cases} \tag{4.5}$$

Now for each target document we have computed twelve vectors ($\vec{d_{f_1}}$ through $\vec{d_{f_{12}}}$) representing the various text fields. We will combine them by weighted sum as in Equation 4.6 to form a single vector on which the classifier is performed.

$$\vec{d}_{comb} = \sum_{i=1}^{12} w_{f_i} * \vec{d_{f_i}} \quad \text{where } 1 = \sum_{i=1}^{12} w_{f_i} \tag{4.6}$$

The vector $\vec{d}_{comb}$ is used as the document representation in the F-Neighbor algorithm. The weights w_{f_i} in Equation 4.6 will be determined experimentally.

4.2.3 Parameter tuning

The determination of the weights w_{f_i} in Equation 4.6 can be seen as an optimization problem in which the classification accuracy based on the aggregated representation is the target to be optimized. Therefore, any generic optimization technique can be applied. In this work, we used a two-layer optimization approach, in which the lower layer optimization optimizes the combination among fields of the same neighbor type (e.g., child title and child text), while the upper layer optimizes among all the neighbors based on the result of the lower layer optimization. Figure 4.1 illustrates this optimization process.

4.3 Experiments

4.3.1 Dataset and classifier

The experiment is performed on same the 12,000 page ODP dataset as introduced in Chapter 3. The partitions in the five folds remain the same as before. Still, linear kernel SVM classifiers with default parameter setting based on SVM^{light} [97] is used in our experiment, using one-against-others generalization. The reported classification accuracy is the average across the five folds unless stated otherwise.

4.3.2 Lower layer optimization

At lower layer optimization, fields of each neighbor type are combined independently to achieve the best accuracy within that neighbor type. The questions are: what is the best combination that can be gained within each neighbor type; and, how much is the benefit of emphasizing titles/anchor text over other text.

We start by showing the benefit of emphasizing titles by investigating the combinations of title and text within each neighbor type (and target page itself), where the weight of title and the weight of text add up to one. Figure 4.2 shows the tuning result on the first fold of data. X-axis is the weight of title. A weight of 0 means classifying on the whole content, without emphasizing the title. A weight of 1 means classifying only using the title. The results shows that although there is marginal benefit emphasizing titles of the target (2%) and siblings (7%), other neighbor types benefit a lot (18% relative improvement for parent pages, 10% for child, and 20% for spouse).

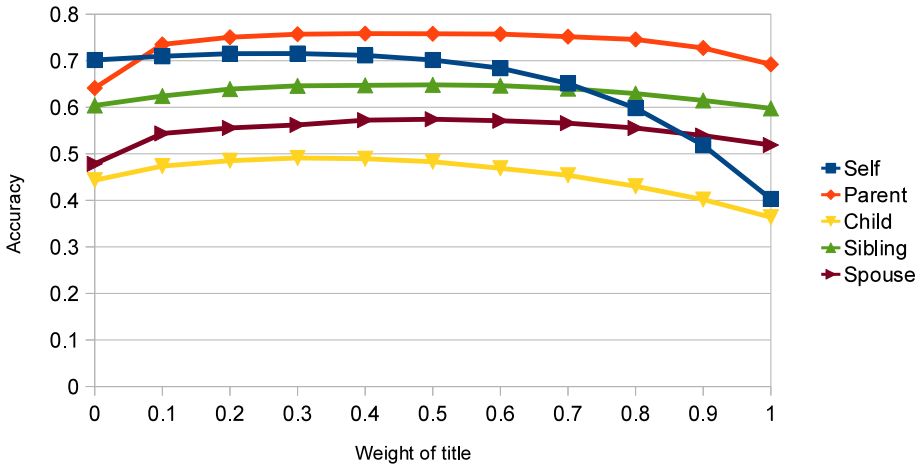


Figure 4.2: The tuning of title and text.

Anchor	Extended anchor	Title	Text	Accuracy
0.0	0.1	0.4	0.5	0.7589
0.0	0.0	0.4	0.6	0.7582
0.1	0.0	0.4	0.5	0.7580
0.0	0.0	0.5	0.5	0.7579
0.0	0.1	0.5	0.4	0.7575

Table 4.1: Combinations of parent fields with highest accuracy.

We continue to examine the usefulness of anchor text and extended anchor text. Unlike previous work [77], the result is not encouraging. Although there is some benefit by individually emphasizing anchor text (an increase from 64% to 69% in accuracy) and extended anchor text (also 69%), neither of them is as powerful as emphasizing title (76%). We also combined four fields on parent page together, i.e., parent title, parent text, anchor text, extended anchor text, with their weight sum to one. As shown in Table 4.1, the top combinations with highest accuracy on Fold 1 assigned little value to anchor text and extended anchor text.

In addition, we combined the four fields on parent and the text on the target page (without emphasizing title of target). The result is similar: anchor text and extended anchor text is somewhat useful when combined with target text, but not as useful as the others.

In 2002, Glover et al. [77] showed significant improvement using off-page anchor text compared with classifiers only using local text. Although we also find anchor text useful, it is not as significant. The reasons could be:

- different datasets: although pages in Yahoo directory used by Glover et al. are similar in many ways to those in ODP, Glover et al. used finer categories (second-level or deeper) while we only consider top-level categories;
- different number of incoming links: Glover et al. used at most 20 incoming links for each target page, while we used at most 50, which may consequently affect the describing power of the collected anchor text;
- different ways of using anchor text: Glover et al. directly pulled all anchor text into a virtual document without normalization, and only consulted a local text classifier when the classifier based on anchor text was uncertain, while we normalize each anchor text before combining, and combine it with local text without considering uncertainty.

Through our experiment, we found that parent title is more important than anchor text. Possible reasons are as following.

- We focused on broad categories. Anchor text are usually specific (e.g., "Nokia"), therefore do not help much in classifying broad categories. On the other hand, titles are usually more generic than any anchor text on the same page (e.g., "cell phones"), which makes them more useful.

- The pages that we used in our experiment are considered to be good quality pages. The content of these pages tend to be well-written, self-describing. Therefore, resorting supplemental information (such as anchor text) is not easy to achieve significant improvement.

Next, we examine the usefulness of labels. For each type of neighbor, we tuned weights of fields for best classification accuracy for two separate situations. First, we tune on fields without considering any labels. Then, we include labels as a field and tune again. Figure 4.3 shows the comparison between the two situations on Fold 1. We can see that considering labels can bring significant improvement for child, sibling, and spouse pages.

In summary, the best combination achieved for each fold in lower layer optimization is listed in Table 4.2. For each type of pages (target, parent, etc.), the first row shows the best accuracy in each fold, the second row shows the best combination parameter setting. For example, the parameter setting of "3:7" for target page means 0.3 weight for target title and 0.7 for normal text. For parent, the five fields are anchor text, extended anchor text, title, text, and label, respectively. For child, sibling, and spouse, the three fields are title, text, and label.

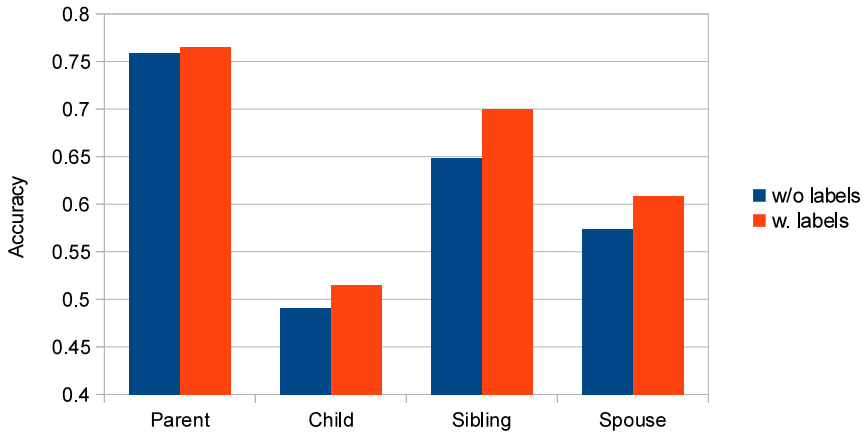


Figure 4.3: Comparison of best performance with labels and without labels.

		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	**Avg**	**StDev**
Target	Accu.	0.715	0.711	0.715	0.711	0.719	0.715	0.003
	Param.	*3:7*	*3:7*	*2:8*	*3:7*	*3:7*		
Parent	Accu.	0.766	0.767	0.767	0.768	0.762	0.766	0.002
	Param.	*0:1:3:4:2*	*3:4:0:2:1*	*0:0:4:4:2*	*0:1:3:4:2*	*0:0:4:4:2*		
Child	Accu.	0.515	0.523	0.516	0.521	0.524	0.520	0.004
	Param.	*3:5:2*	*2:5:3*	*2:4:4*	*2:5:3*	*3:5:2*		
Sibling	Accu.	0.700	0.706	0.694	0.705	0.704	0.702	0.005
	Param.	*3:5:2*	*4:4:2*	*4:4:2*	*4:4:2*	*4:4:2*		
Spouse	Accu.	0.609	0.605	0.607	0.605	0.614	0.608	0.004
	Param.	*3:4:3*	*3:3:4*	*4:4:2*	*3:4:3*	*4:3:3*		

Table 4.2: Summary of lower layer optimization result.

4.3.3 Upper layer optimization

Based on the optimal combination achieved in lower layer optimization of each neighbor type, upper layer optimization tunes the weighting between neighbor types with the weighting within each neighbor type fixed. For example, if the best weight of sibling title and sibling text found in lower layer optimization are $wl_{sibling,title}$, and $wl_{sibling,text}$(as listed in Table 4.2, 0.1 and 0.9), respectively, then their final weight in the full combination

Target	Parent	Child	Sibling	Spouse	Accuracy
0.2	0.4	0.0	0.2	0.2	0.8838
0.3	0.4	0.0	0.2	0.1	0.8826
0.3	0.3	0.0	0.2	0.2	0.8817
0.2	0.5	0.0	0.2	0.1	0.8815
0.2	0.3	0.0	0.3	0.2	0.8814

Table 4.3: Combinations of upper layer optimization with highest accuracy in Fold 1.

Fold	Target	Parent	Child	Sibling	Spouse	Accuracy
1	0.2	0.4	0.0	0.2	0.2	0.8838
2	0.3	0.4	0.0	0.2	0.1	0.8915
3	0.2	0.4	0.0	0.2	0.2	0.8897
4	0.2	0.4	0.1	0.2	0.1	0.8861
5	0.3	0.4	0.0	0.2	0.1	0.8902

Table 4.4: The best combination of parameters at upper level for each fold.

is $wh_{sibling} * wl_{sibling,title}$, and $wh_{sibling} * wl_{sibling,text}$, respectively, where $wh_{sibling}$ (as well as weighting of other neighbor types) is to be determined experimentally.

The top five combinations with highest accuracy in Fold 1 are listed in Table 4.3. The best combination for each fold is listed in Table 4.4. The parameters in these tables show little value of child pages, low but consistent value of sibling and spouse pages, and high value of parent pages. Compared with the usefulness showed by the default Neighboring Algorithm in Chapter 3, the usefulness of child and spouse is similar. Parent pages gain more importance because of emphasizing title; while siblings become less useful. Assuming the feature values for every field are pre-calculated, the whole tuning process (on both upper and lower layers) for one fold on ODP data takes approximately 183 CPU hours on a Xeon 3.0 GHz CPU.

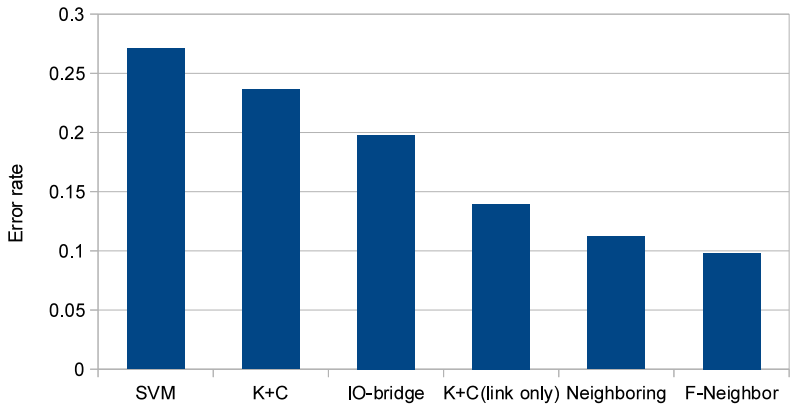


Figure 4.4: Comparison of error rate of different algorithms.

4.3.4 Experimental result on ODP dataset

After tuning the weighting of each field, we apply the best parameter setting to the set-aside test documents. We compare F-Neighbor with the Neighboring Algorithm from Chapter 3, as well as other algorithms being compared previously. The result is shown in Figure 4.4. Previously, the Neighboring Algorithm has an accuracy of 88.8%. The F-Neighbor algorithm further raised the accuracy to 90.2%, reducing the error rate by one eighth. T-tests show that the F-Neighbor Algorithm outperforms other algorithms statistically significantly ($p \leq 5 \times 10^{-3}$).

In order to show the advantage of F-neighbor over the Neighboring Algorithm, we selected a subset consisting of all target pages that have less than 200 sibling pages. The subset contains 1,333 pages, approximately 11.1% of all target pages in the ODP dataset. Figure 4.5 shows the error rates of the SVM textual classifier, the Neighboring Algorithm, and the F-Neighbor Algorithm, respectively (averaged across 5 folds). On this subset,

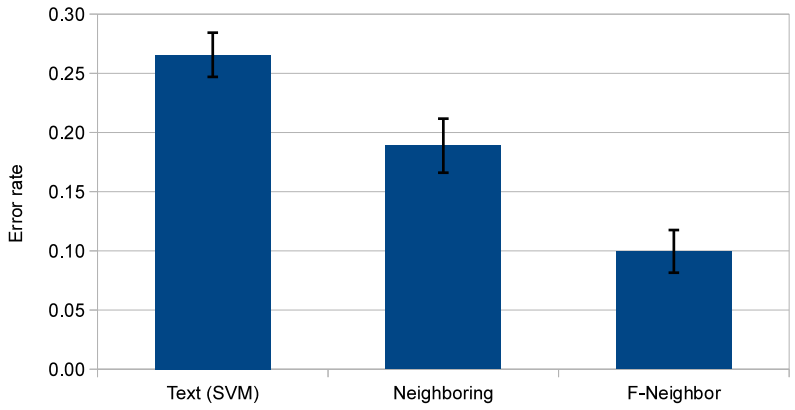


Figure 4.5: Comparison of error rate of on a select subset.

the Neighboring Algorithm reduces the error rate by almost one third compared with the baseline, SVM classifiers based on the target page itself. The F-Neighbor Algorithm further cuts the error rate of Neighboring by a half. The error bar on each column represents the standard deviation. T-tests show that the F-Neighbor Algorithm outperforms Neighboring and the textual classifier statistically significantly ($p \leq 2 \times 10^{-3}$).

4.3.5 Experimental result on WebBase dataset

In order to show our algorithm's performance on generic web pages, we also apply the learned classifier to the WebBase dataset as mentioned in Chapter 3. In the previous experiment, the default Neighboring Algorithm increased the accuracy from 30.4% to 36.8%. The F-Neighbor Algorithm further improves performance by more than 10% (to 40.8%).

4.4 Discussion and Conclusion

In Chapters 3 and 4, we have shown the improvements that our proposed algorithms can provide for web page classification. However, they also have some limitations, which we discuss here.

- While performance is a function of a number of tunable parameters, we have not fully explored the parameter space. In the tuning process, we assumed that the weighting of the fields of a neighbor type is independent of other neighbor types. Such a untested assumption may lead to sub-optimal solutions.

- The ODP dataset used for most of our experiments generally consists of highly-regarded pages. Our experiment with WebBase data suggests that raw performance on the ODP dataset may be higher than generic web pages. This effect might be mitigated by using training data that better matches the test data (e.g., training on random web pages).

- We only utilized neighbor information to help determine the target page's topic. The classification of the target page itself, however, may similarly affect the neighboring pages' topic. A relaxation technique (e.g., as used in another algorithm from [29]) might be a useful addition to our approach.

- For simplicity, classification is only performed on the first-level categories of dmoz Directory. Conducting similar classification at a deeper level, or on more fine-grained topics, may expose more interesting facts.

- In the F-Neighbor Algorithm, while we break up a web page into several fields according to its HTML markup, other fields may also be worth consideration, e.g., headers or text in large fonts. To be more general, one may break up pages based on metrics other than HTML tags, such as spatial information or even complex models as the one proposed by Xiao et al. [201].

- The results of our tuning process in both Neighboring and F-Neighbor algorithms indicate that neighboring pages are a better source to tell a page's topic than the page itself. We provided a preliminary explanation in Section 3. A deeper study is needed to verify the actual reason.

- In our algorithms, when a neighboring page has multiple roles (e.g., being a parent and a sibling at the same time), it is accounted for all roles it plays. A further exploration may reveal better solutions. For example, instead of duplicating a page according to its multiple roles, we can treat it as a single page while splitting its influence to the target page according to its roles. We may also choose one role in which the page has the highest potential to contribute.

Our contributions in this chapter include the following:

- We demonstrated that it is very useful to incorporate field information of neighboring pages into web page classification.

- We are the first to find out the unexpectedly high utility of parent page titles in web classification on broad topics.

In summary, this chapter has demonstrated the usefulness of information from neighboring pages, especially the fielded information, in judging the topic of a target web page. The experimental results show that, under appropriate parameter settings, our algorithm statistically significantly outperforms textual classifiers as well as existing algorithms, including the Neighboring Algorithm in the previous chapter.

Chapter 5

Hierarchy Evolution for Improved Classification

5.1 Introduction

In general, classifiers categorize web pages into a predefined set of categories. In Chapters 3 and 4, we proposed methods to directly enhance web classification by utilizing neighboring pages on the Web, in which classification is performed on a flat set of categories. Such categories can also be organized into a hierarchy. As we discussed in Chapter 2, based on how the categories are organized, classification can be divided into flat classification and hierarchical classification. In flat classification, a single classifier learns to classify instances into one of the target categories. In hierarchical classification, a separate classifier is trained for each non-leaf node in the hierarchy. During training, each classifier is trained

to categorize the instances that belong to any of the descendants of the current node into its direct subcategories. When deployed, an instance is first classified by the root classifier, and then passed to one of the first level categories with the highest probability. This process is repeated iteratively from top to bottom, invoking only one classifier at each level, until reaching a leaf node.

Previous work has shown that hierarchical classification is more accurate than flat classification (e.g., [58, 117, 16]). By organizing categories into a hierarchical structure and training a classifier for each non-leaf node, each classifier can focus on a smaller set of subcategories, and thus reduce the confusion from sibling branches. It is also reported that hierarchical classification can significantly reduce training and testing time compared with flat classification [117, 124].

Hierarchical classification is typically performed utilizing human-defined hierarchies. Since such hierarchies reflect a human view of the domain, they are easy for people to understand and utilize. However, these hierarchies are usually created without consideration for automated classification. As a result, hierarchical classification based on such hierarchies is unlikely to yield optimal performance.

In this chapter, we propose a new classification method based on genetic algorithms to create hierarchies better suited for automatic classification. In our approach, each hierarchy is considered to be an individual. Starting from a group of randomly generated seed hierarchies, genetic operators are randomly applied to each hierarchy to slightly reorganize the categories. The newly generated hierarchies are evaluated and a subset that

are better fitted for classification are kept, eliminating hierarchies with poor classification performance from the population. This process is repeated until no significant progress can be made. In our experiments on several text classification tasks, our algorithm significantly improved classification accuracy compared to the original hierarchy and also outperformed state-of-the-art adaptation approaches. Compared with previous work, our approach is different in at least two aspects:

- After each iteration, we keep a comparatively large number of best performing hierarchies rather than only keep the best one and discard the rest; we will show later that the best hierarchy does not always come from an adaptation of the previous best hierarchy.

- Unlike previous approaches that gradually adapt a hierarchy by making a slight change at each step, the crossover operators we customize for hierarchy evolution allow a new hierarchy to inherit characteristics from both parents, so that it is significantly different from either parent. This will take previous approaches many more iterations to achieve, or perhaps unachievable at all because of the use of a greedy search strategy.

Our contributions include:

- a new, better-performing approach to improved classification by hierarchy adaption; and,

- adaptation of genetic operators on hierarchies and an analysis of their utility.

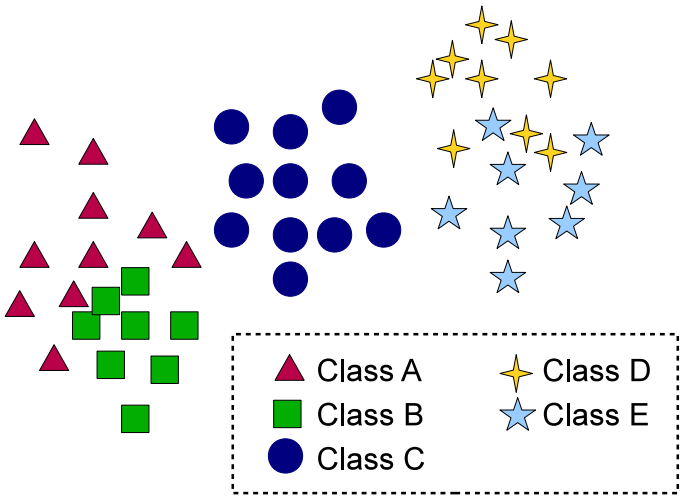


Figure 5.1: An imaginary five-class classification problem.

The rest of this chapter is organized as follows. We motivate and introduce our approach in Section 2, report the experimental setup and results in Section 3, and conclude this chapter in Section 4.

5.2 Approach

5.2.1 Motivation

As we introduced previously, hierarchical classification can often perform better than flat classification. The main reason is that, by classifying objects first into high level categories, and then iteratively into finer-grained subcategories, the classifier at each branching point should have an easier task than classifying into all categories at once. However, this is not always true. Consider the example in Figure 5.1, where class $A \cup B$, class C, and class $D \cup E$ can be easily separated, while separating class A from class B, class D from class

E is comparatively difficult. If a classifier first separates $A \cup B$, C, $D \cup E$ from each other, then separates A from B, D from E, it should be easier than classifying all the five classes at once. However, if based on a suboptimal hierarchy, it first tries to separate $A \cup D$ from $B \cup C \cup E$, the increased difficulty may significantly reduce the quality of classification. From this example, we can see that although hierarchical classification often performs better than flat classification, it depends on the choice of hierarchy.

In order to further motivate this work using real-world data, we randomly selected 7 leaf categories containing 227 documents from the LSHTC dataset (see Section 5.3.1 for details about LSHTC dataset), exhaustively generated all the possible hierarchies based on the selected categories, and tested the classification performance for every hierarchy. In total, 39,208 hierarchies were generated, out of which 48.2% perform worse than flat classification in terms of accuracy. The distribution of accuracy is shown in Figure 5.2. The top 0.03% of all the hierarchies can achieve 100% accuracy, with the next 0.03% at 31.6% accuracy. Around 51.7% of the hierarchies perform as well as flat classification with an accuracy of 27.1%. The rest perform worse than flat classification. Some even classify all instances incorrectly. This further verifies our intuition that improving hierarchical classification needs a well-chosen hierarchy. In the following, we will describe how to adapt genetic algorithms to search for a better hierarchy.

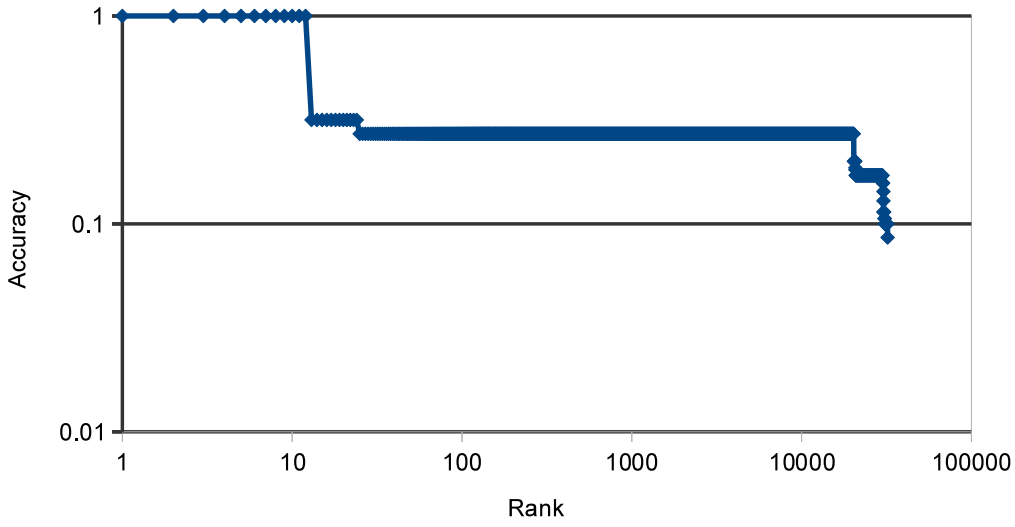


Figure 5.2: Distribution of accuracies of all possible hierarchies based on the seven randomly selected leaf categories.

5.2.2 Overview

A genetic algorithm (GA) is a search/optimization method that resembles the evolution process of organisms in nature. Like any other search method, it searches through the solution space of a problem, looking for an optimal solution. In our problem, we consider each possible hierarchy to be a solution (or an individual in the GA, specifically). As illustrated previously, our solution space is often too large to perform an exhaustive search except for very small datasets.

A typical GA usually starts with an initial population of individuals, then iteratively repeats the following search procedure until the stopping criterion is satisfied. In each iteration, a subset of individuals is selected for reproduction by applying mutation and

crossover operations on them. Then the fitness of each new individual is evaluated, and low-fitness individuals are dropped, leaving a better fitted population at the start of next iteration. In our approach, we will leave the high level procedure of GAs as described above unchanged, while adapting representation method and reproduction operators to make them fit the hierarchical classification problem. We chose a GA instead of other generic search methods for at least two reasons. First, it intrinsically supports large populations rather than greedy, single path optimization. Second, we can adapt its reproduction operators to allow significant changes to the solutions without changing the high level search procedure. In an analysis of experimental results in Section 5.3, we will show that the above properties are essential to the performance improvement.

5.2.3 Hierarchy representation

We start by describing how to represent a hierarchy using a string. In a GA, reproduction among the population of a given generation produces the next generation. For easier reproduction operations and duplicate detection, we need to design a serialized representation for hierarchies. In our work, each hierarchy is represented as a sequence of numeric IDs and parentheses, in which each ID corresponds to a leaf category, and each pair of parentheses represents a more generic category consisting of the categories in between them. Multiple levels in the hierarchy are reflected using nested parentheses.

More formally, we represent a hierarchy using the following rules:

1. Each leaf node is represented by its numeric id;

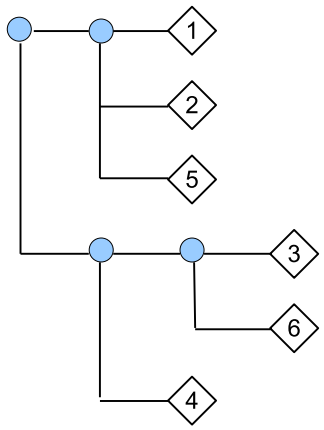


Figure 5.3: A small hierarchy example.

2. Each non-leaf node is represented by a list of all its children nodes enclosed in a pair of parentheses;

3. The hierarchy is represented recursively using Rule 1 and 2.

4. The outermost pair of parentheses is omitted.

Figure 5.3 illustrates a small example, which will be represented as (1 2 5) ((3 6) 4).

5.2.4 Representation string canonicalization

The hierarchy representation method above serializes a hierarchical tree into a sequence of tokens. However, different representations may correspond to the same hierarchy. Using the example in Figure 5.3, (1 2 5) ((3 6) 4) and (2 1 5) ((6 3) 4) define the same hierarchy. Since we limit the size of the population, detecting duplicate hierarchies not only saves fitness evaluations for already evaluated hierarchies, but also encourages variety in the population, which is an important factor for performance improvement as

we will show later. Therefore, we need a mechanism to normalize the representations so that duplicates are easily detected. We call this process canonicalization. Two steps of canonicalization are used in our work: trimming and sibling order canonicalization.

Trimming. We define a trivial node as a node with only one child. Trivial nodes are not useful in hierarchical classification. We use the following rule to trim the hierarchy and eliminate trivial nodes: $((S_t)) \Longrightarrow (S_t)$, where S_t is the representation of a subtree t.

Sibling order canonicalization. As described earlier, each non-leaf node is represented by a list of all its children nodes enclosed in a pair of parentheses. In this list, the order of the nodes is not important, i.e., if the only difference of two representation strings is the ordering of sibling nodes, they should be considered the same. For any node n_i in the hierarchy and the subtree t_i rooted at n_i, we define the function $smId(n_i)$ as the smallest ID in t_i. More formally, for any node n_i, $smId(n_i)$ is recursively defined as:

$$smId(n_i) = \begin{cases} n_i & \text{if } n_i \text{ is a leaf} \\ \min_{k:n_k \subset n_i} smId(n_k) & \text{otherwise} \end{cases} \tag{5.1}$$

where n_k is a child of n_i. In the representation, n_i and its siblings are sorted according to their $smId(\cdot)$ value. For example, (2 1 5) ((3 6) 4) and (2 1 5) (4 (3 6)) will both be canonicalized into (1 2 5) ((3 6) 4).

5.2.5 Seed generation

In GA, an initial population needs to be generated before the iterative evolution process is simulated. In our approach, we use a random algorithm to generate each hierarchy in

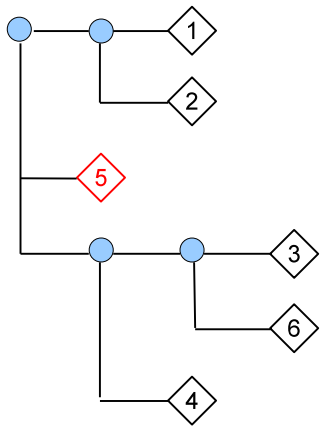


Figure 5.4: An example of promotion mutation: promoting Node 5 in Fig. 5.3.

the initial population. The pseudocode is shown in Algorithm 5.1. The algorithm first sequentially initializes a sequence using integers from 0 to $n-1$, where n is the number of leaf categories in the hierarchy. Then, using the Fisher Yates shuffling algorithm [62], the integers representing each leaf category are randomly shuffled such that each integer has an equal probability (i.e., $1/n$ in this case) appearing at any position in the sequence. After that, k pairs of parentheses are inserted into the sequence at random positions, where k is a random number between 0 and $n-1$.

5.2.6 Reproduction

Mutation

Mutation is the genetic operator in which an individual is slightly changed to maintain the diversity of the population. In a typical GA, this is performed by switching a random bit in the chromosome string. However, this operation is not as straightforward in the hierarchical setting. We design three mutation methods that are suitable for hierarchy

Algorithm 5.1 Algorithm to generate a random hierarchy with n leaf nodes.

```
1: for i := 0 to n − 1 do
2:    sequence[i] ← i
3: end for
4:
5: {randomly shuffle the sequence using Fisher Yates shuffling process}
6: for i := n − 1 downto 0 do
7:    randomly choose an integer r from the range [0..i], inclusive
8:    swap (sequence[i], sequence[r])
9: end for
10:
11: randomly choose an integer k from the range [0..n − 1], inclusive
12: {k will be the number of pairs of parentheses to be inserted}
13:
14: {insert parentheses}
15: for i := 0 to k − 1 do
16:    randomly choose two unequal integers pos1 and pos2 from the range
       [0..sequence.length], inclusive
17:    if pos1 > pos2 then
18:       swap (pos1, pos2)
19:    end if
20:    insert a left parenthesis at pos1 in sequence
21:    insert a right parenthesis at pos2 in sequence
22: end for
```

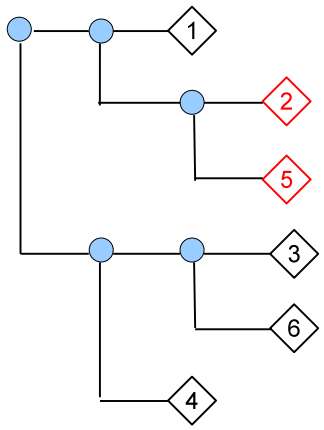


Figure 5.5: An example of grouping mutation: grouping Node 2 and 5 in Fig. 5.3.

evolution: promotion, grouping, and switching.

The *promotion* operator randomly selects a node n (which can be either a leaf node or a non-leaf node), and promote n as a child of its grandparent, i.e., n becomes a sibling of its parent node. For example, promoting Node 5 in Figure 5.3 generates the hierarchy in Figure 5.4. As a special case, promoting the root node results in no change in the hierarchy. If node n has only one sibling m, then promoting n is equivalent to promoting m, and also equivalent to promoting both m and n while removing their parent.

The *grouping* operator randomly selects two sibling nodes and groups them together. If a non-leaf node n has k children, $C = \{c_i | i = 1..k\}$, where $k \geq 2$. We randomly select two nodes c_x and c_y from c_1 through c_k, and remove them from C. Then we add a new node c_z into C so that c_z becomes a child node of n. Finally, we make c_x and c_y children of c_z. For example, grouping Node 3 and Node 5 in Figure 5.3 generates the hierarchy in Figure 5.5.

The *switching* operator randomly selects two nodes m and n (and the subtrees rooted

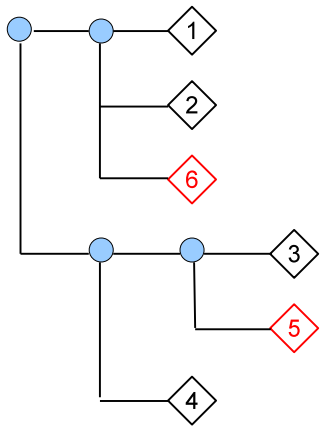


Figure 5.6: An example of switching mutation: switching Node 5 and 6 in Fig. 5.3.

at those locations) in the whole hierarchy, and switches their positions. For example, switching Node 5 and Node 6 in Figure 5.3 generates the hierarchy in Figure 5.6. In the above examples, all mutation operations happened at leaf nodes. This is only for the purpose of easy illustration. The three mutation operators introduced here can promote, group, or switch non-leaf nodes.

Crossover

Crossover is the genetic operator in which two individuals (parents) are combined to generate new individuals (children) so that each child will inherit some characteristics from each parent. In a typical GA, crossover is performed by swapping segments of the chromosome string between the parents (illustrated in Figure 5.7). In the hierarchical setting, however, directly swapping parts of the hierarchy representation will generate invalid hierarchies. As shown in the example in Figure 5.8, the resulting hierarchy representations have missing/duplicate leaf nodes and unmatched pairs of parentheses. Therefore, we need

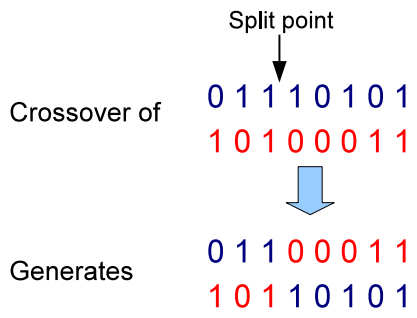


Figure 5.7: An example of crossover in a generic GA setting.

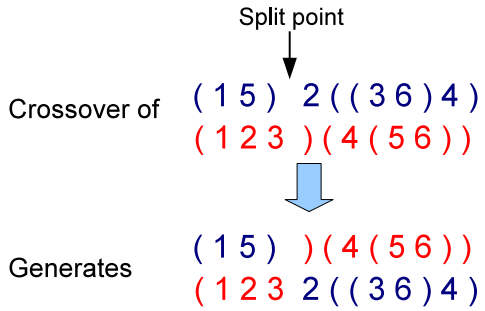


Figure 5.8: Applying the generic crossover operator on hierarchies may generate invalid offsprings.

crossover methods customized for hierarchy evolution.

We used two types of methods: swap crossover and structural crossover. The two parents are noted as h_{p1} and h_{p2}, the children h_{c1} and h_{c2}. In *swap crossover*, a child h_{c1} is generated using the following steps. First a split point p is randomly chosen in h_{p1}. We note the part starting from the beginning of the representation string to the split point p as h'_{p1}. We remove the segment after p from h_{p1} and only consider h'_{p1}. Then right parentheses are added at the end to balance with the existing left parentheses. Suppose S is the set of leaf nodes that appear in h'_{p1}; we go through h_{p2} and remove all the nodes n

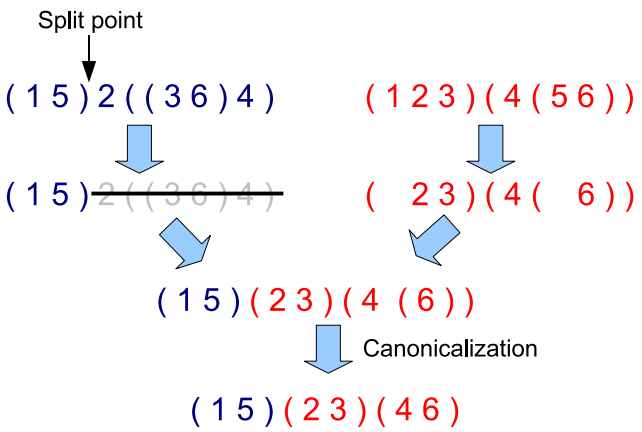


Figure 5.9: An example of swap crossover.

if $n \in S$. This removal transforms h_{p2} into h'_{p2}. Finally, h'_{p1} and h'_{p2} are concatenated to form h_{c1}. The other child h_{c2} is generated by switching h_{p1} and h_{p2} before applying the above procedure. The above operation guarantees that the generated children h_{c1} and h_{c2} are valid hierarchies while each inherits certain characteristics from their parents h_{p1} and h_{p2}. Figure 5.9 illustrates the process of swap crossover.

A hierarchy can be seen as an integration of two independent characteristics: the tree structure and the placement of leaf nodes. At a high level, *structural crossover* aims to "mix and match" these two factors. A child hierarchy inherits the structural information from one parent, and placement of leaf nodes from the other. In our implementation, h_{c1} is generated using the following method. First, every leaf node in h_{p1}'s representation is replaced with a blank space. Then these blank spaces are filled with the leaf nodes in h_{p2} using the order that they appear in h_{p2}. h_{c2} is generated by switching h_{p1} and h_{p2}. Figure 5.10 illustrates the process of structural crossover. Although the above reproduction

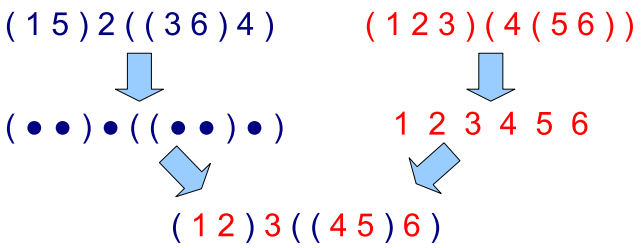


Figure 5.10: An example of structural crossover.

operators guarantee the validity of the generated child hierarchies, they may generate non-canonical representations. Therefore, canonicalization is needed after reproduction.

5.2.7 Fitness function

In each iteration of the GA, the individuals (i.e., hierarchies) in the new generation need to be evaluated. We define the fitness function $fit(h)$ of a hierarchy h simply as the classification accuracy on h. Given a set of training data D_{train}, validation data $D_{validation}$, and the base hierarchical classifier CL,

$$fit(h) = accuracy(CL(D_{train}), D_{validation}) \tag{5.2}$$

where accuracy is defined as the ratio of the correctly classified documents out of all the documents being classified.

5.2.8 Stopping criterion

A GA needs a stopping criterion to terminate the iterative evolution process. In our algorithm, we keep a watch list of top N_{watch} best hierarchies. If the performance of the

Category	Num. of Documents
course	930
department	182
faculty	1,124
other	3,764
project	504
staff	137
student	1,641

Table 5.1: Categories and class distribution in WebKB dataset.

top hierarchies do not change between two consecutive iterations, the algorithm stops and outputs the top hierarchies. In the following experiments, we set N_{watch} to 5.

5.3 Experiments

In this section, we test our hierarchy evolution algorithm using real-world data, and compare its performance with previous methods.

5.3.1 Experimental setup

In order to test our algorithm, we used three public datasets. The first two datasets are from the first Large Scale Hierarchical Text Classification (LSHTC) challenge[1] held in 2009. We selected a toy dataset from Task 1 with 36 leaf categories and 333 documents, which will be referred to as LSHTC-a. We also used the dry-run dataset from Task 1, which has 1,139 leaf categories and 8,181 documents. We will refer to this dataset as LSHTC-b. Both datasets are partitioned into three subsets: a training set used to train classifiers during the training process, a validation set used to estimate the fitness

[1] http://lshtc.iit.demokritos.gr/node/1

score of each generated hierarchy, and a test set to evaluate the final output hierarchy. The third dataset is WebKB[2], containing 7 leaf categories and 8,282 documents. The documents in WebKB dataset are web pages crawled from the following four universities: Cornell (867 web pages), Texas (827 web pages), Washington (1,205 web pages), and Wisconsin (1,263 web pages), plus 4,120 web pages from other universities. These web pages are manually categorized into one of the categories listed in Table 5.1. On the split of training and test data, the provider of the dataset suggests "training on three of the universities plus the misc collection, and testing on the pages from a fourth, held-out university". According to this, we performed four-fold cross-validation on WebKB with a minor adaptation of the split method. Each fold trains on data from two of the universities plus the "misc" collection (web pages from other universities), validates on a third university, and tests on a fourth university. LibSVM [32] is used as the base classifier to implement the standard hierarchical SVM. We used all the default settings in LibSVM, including the radial basis kernel function as it yields better performance than linear kernel according to our experiments. In our algorithm, we set the population size to 100 on LSHTC-a and WebKB, 500 on LSHTC-b.

We compared our approach (Evolution) with two existing state-of-the-art approaches: a hierarchy adaptation approach called Hierarchy Adjusting Algorithm (HAA) [188], and a hierarchy generation approach called Linear Projection (LP) [115]. We implemented HAA according to the algorithm outlined in Figures 12 and 13 of [188]. All three search

[2]http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

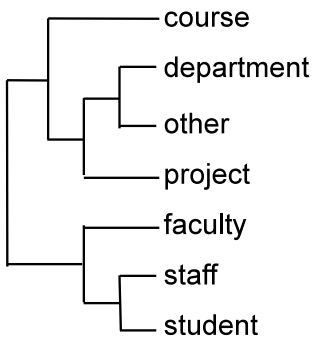


Figure 5.11: The hierarchy automatically generated by Linear Projection (redrawn based on the experimental result in the LP paper [115]).

methods were implemented. In our implementation of HAA, we set the stopping criterion to 0.001. That is, when the improvement between two consecutive iterations is less than 0.001, the algorithm terminates. We also compared our algorithm with Linear Projection on the WebKB dataset. Instead of re-implementing the LP algorithm, we directly used the automatically generated hierarchy on WebKB reported in the Linear Projection paper, and performed the four-fold cross-validation based on that hierarchy (shown in Figure 5.11).

5.3.2 Experimental results

On the small LSHTC-a dataset, a flat classification has an accuracy of 68.6%. The hierarchical classification using the original, human-built hierarchy performs slightly better at 70% accuracy. HAA converged after two iterations with the accuracy improved to 71.9%. Since the initial population in our Hierarchy Evolution algorithm is generated randomly, we ran our algorithm three times using different random seeds. The averaged accuracy is 79.5%. On the LSHTC-b dataset, it took 50.3 iterations for our algorithm to converge

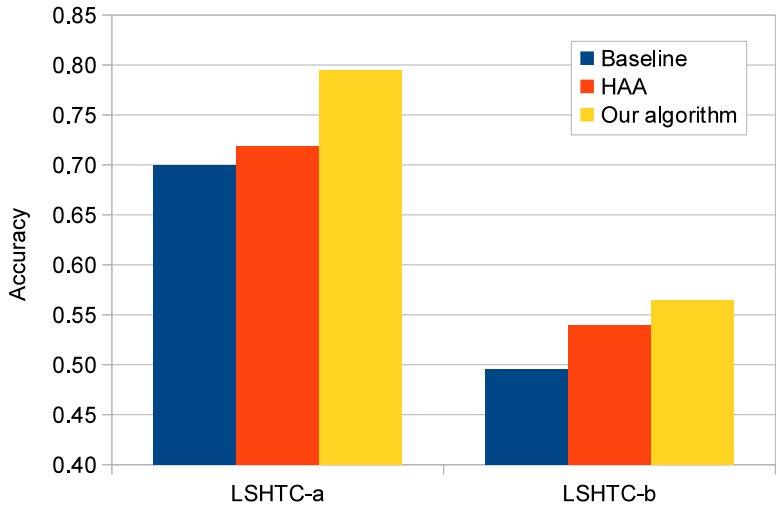


Figure 5.12: Accuracy on LSHTC datasets compared across different methods.

(averaged across 6 runs), and 18 iterations for HAA. HAA improved upon the baseline's accuracy of 49.6% to 54.0% (an improvement of 8.9%), while our algorithm's final output hierarchy has a 56.5% accuracy averaged across 6 runs (an improvement over the baseline of 13.9%). The results are shown in Figure 5.12. To be certain of a fair comparison, we let HAA continue running for three more iterations after convergence, but did not observe any additional improvement. Two-tailed t-tests show that our algorithm outperforms other approaches statistically significant on both LSHTC datasets ($p\ value \leq 0.02$).

When running our algorithm on LSHTC-b, at each iteration, we extracted the best hierarchy in terms of its classification accuracy on the validation set. For comparison, we plotted the per-iteration best hierarchy's accuracy on the validation and test set for one of the six trials in Figure 5.13. The validation accuracy increases monotonically until

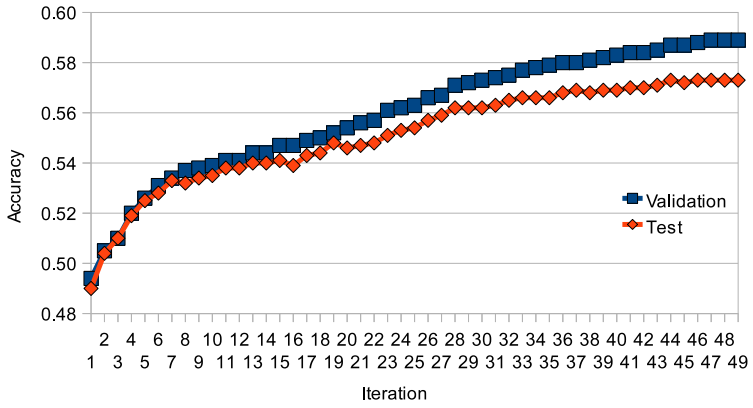


Figure 5.13: Best accuracy at each iteration.

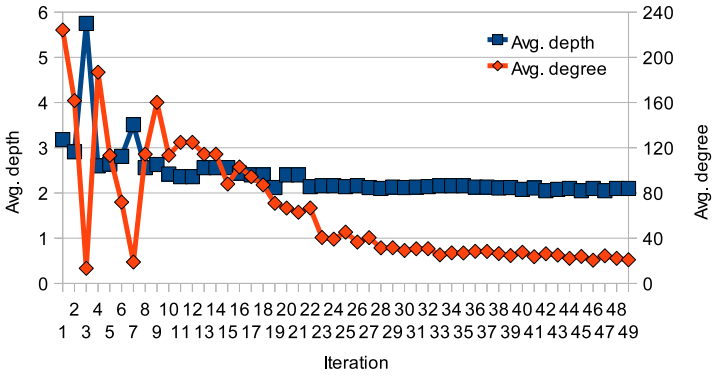


Figure 5.14: Average depth and degree of the best-performing hierarchy at each iteration.

convergence. Although the test accuracy fluctuates a little, it maintains an increasing trend in general. Figure 5.14 shows the average depth and degree of the best-performing hierarchy at each iteration.

On the WebKB dataset, we compared our algorithm with flat classification, HAA, and Linear Projection. Based on the seven leaf categories, a flat classification has an accuracy of 70.3% averaged across the four folds. Linear Projection and HAA improve the accuracy to 74.6% and 75.4%, respectively. Our algorithm further improves to 76.4%, a

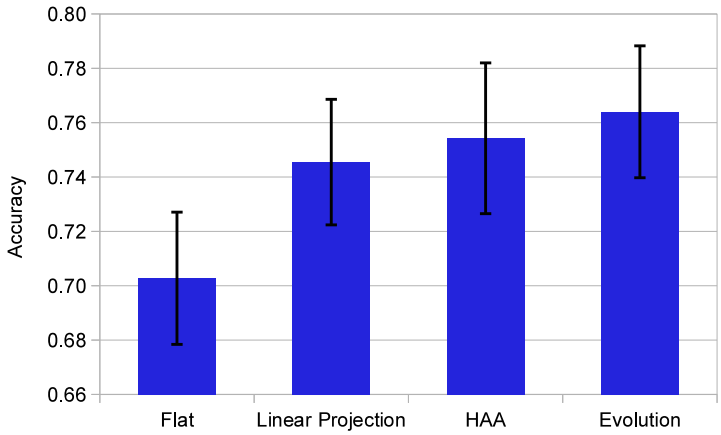


Figure 5.15: Accuracy on WebKB dataset compared across different methods.

Methods	**Flat**	**LP**	**HAA**	**Evolution**
Fold 1	0.687	0.734	0.742	0.749
Fold 2	0.694	0.721	0.721	0.738
Fold 3	0.691	0.774	0.780	0.788
Fold 4	0.739	0.753	0.774	0.781
Average	0.703	0.746	0.754	0.764
STDEV	0.024	0.023	0.028	0.024

Table 5.2: Accuracy of each fold on WebKB compared across different methods.

21% reduction in error rate compared with flat classification. Figure 5.15 shows the average performance and standard deviation for each method. The variance is mainly caused by the difference of data across folds. From Table 5.2, we can see that our approach consistently performs better than other methods on all folds. Two-tailed t-tests showed that our algorithm statistically significantly outperforms all other algorithms being compared, with p-values under 0.03 for all tests. Figure 5.16 shows the best hierarchy generated by our algorithm on the first fold of WebKB cross-validation.

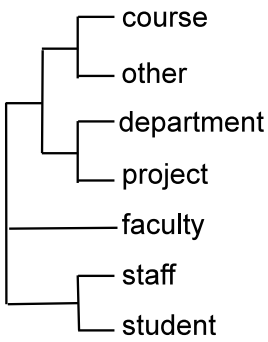


Figure 5.16: The hierarchy automatically generated by our Hierarchy Evolution Algorithm.

5.3.3 Experiment analysis

As we pointed out previously, our approach differs from existing hierarchy adaptation approaches from at least two aspects:

1. genetic operators that allow more significant changes in a hierarchy; and,

2. a larger population size to maintain population variety.

Now we analyze quantitatively whether these differences make our approach outperform existing methods. The following analysis is performed on the LSHTC-b dataset.

In order to evaluate the effectiveness of the genetic operators, we calculate the improvement that each type of operator brings to a hierarchy. Figure 5.17 shows the average improvement in terms of accuracy. On average, all the five operators have a negative impact on the accuracy. For example, the "swap crossover" even decreases accuracy by 0.018 when averaged across all evolution operations. That is, on average, every time a "swap crossover" is applied on a hierarchy, the newly generated hierarchy has an accuracy lowered by 0.018. Fortunately, the genetic algorithm only keeps the best hierarchies in the

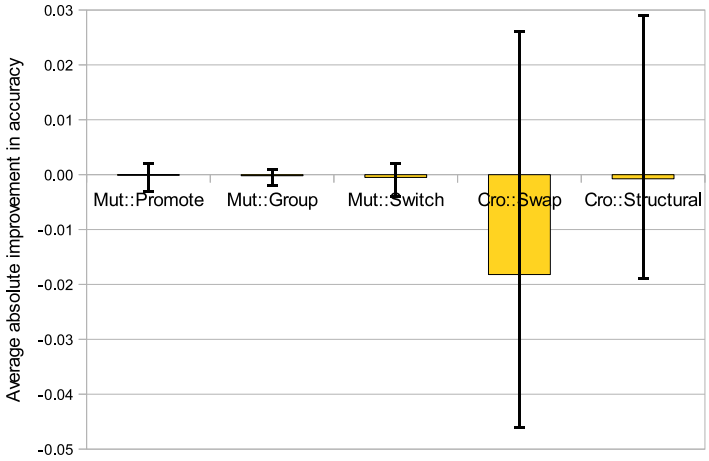


Figure 5.17: Improvement compared across different genetic operators.

population, and discards the rest. Therefore, the degradation is counter-balanced by such a selection process, making an overall increasing trend (as we showed previously in experimental results). We also calculated the standard deviation of the improvement, as well as minimal and maximal improvement. The error bars in Figure 5.17 show the minimal and maximal improvement. The standard deviation of the operators are 0.0004, 0.0005, 0.0009, 0.0100, 0.0028, respectively. This indicates that the mutation operators only have slight impact on the accuracy while changes made by crossover are more significant. In the best case, "swap crossover" improved accuracy by 0.026, "structural crossover" improved 0.029, while all the mutation operators can only improve no more than 0.002 at their best. These statistics match our intuition that more significant changes can potentially bring better improvements than local modifications.

Another method to examine the utility of different genetic operators is to use only a subset of the operators in our algorithm and check the accuracy of the final output

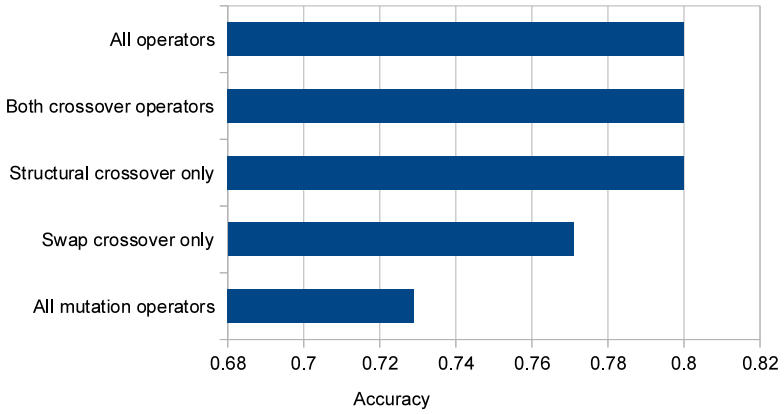


Figure 5.18: Accuracy comparison using different subsets of genetic operators.

hierarchy. The result of this analysis on LSHTC-a is shown in Figure 5.18. Using all operators yields a 80% accuracy as the algorithm converges after 7 iterations. Using both crossover operators without any mutation yields the same accuracy with a slightly slower convergence speed (8 iterations). Using structural crossover only, we can still discover a hierarchy with the same accuracy at 80%, but at a cost of two more additional iterations (10 iterations until convergence). Using swap crossover only, the algorithm converges after 9 iterations with an accuracy of 77%. If we take out both crossover operators and only use mutation operators, the performance of the final output hierarchy is significantly decreased down to 73%.

Unlike previous approaches that only carry the best hierarchy into the next iteration, we keep hundreds of hierarchies in the population. This raises a reasonable concern about our approach: is the large population necessary? To answer this question, we first identified the top 5 hierarchies at each iteration, then back-traced their parents from the

previous iteration, and finally found out the ranks for their parents. The results are shown in Figure 5.19. A point (x,y) means that if we were to keep only top x hierarchies in the previous generation, then only a percentage of y hierarchies that were in the top 5 of the next generation still exist. In other words, the rest of top hierarchies (i.e., 1-y) will no longer exist because their parents were removed from the previous generation. We can see that we can still generate a significant portion of top hierarchies by keeping 300 hierarchies in each iteration. From another point of view, if we were to keep only 50 hierarchies, we would have lost approximately 81% of the top 5 hierarchies at each iteration. Although we could probably shrink the population size by 20% without significant degradation in accuracy, this supports the idea that the comparatively large population is necessary. In order to further verify this conjecture, we performed the experiments again based on various population sizes from 100 to 500, and compared the test accuracy across the final output hierarchies. Figure 5.20 shows that increasing the population size from 100 to 400 gives a significant improvement in terms of accuracy on the final output hierarchy, while no additional benefit is perceived beyond the size of 400.

An additional experiment with respect to the usefulness of a large population size is performed on LSHTC-a dataset. In this experiment, instead of running HAA from a single initial hierarchy (i.e., the original, human created hierarchy), we ran HAA 100 times, with each trial starting from a randomly generated hierarchy. We controlled the random hierarchy generation process so that the 100 initial hierarchies are exactly the same as the initial population previously generated in the first trial of our Hierarchy

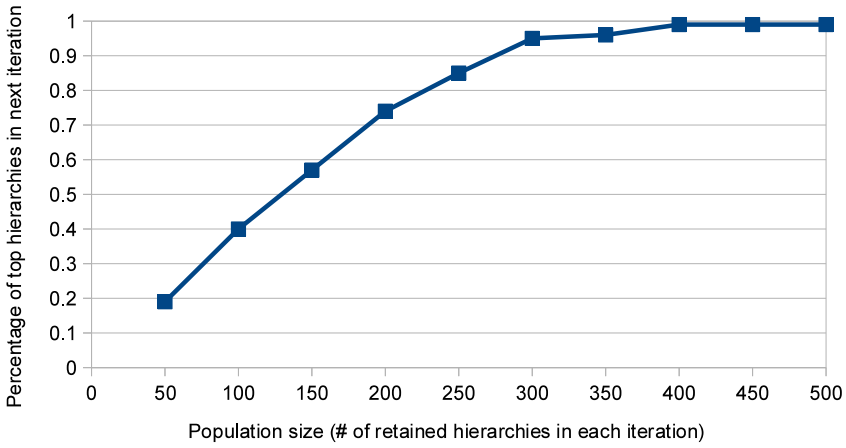


Figure 5.19: Top hierarchies that can be generated when a smaller population size is used.

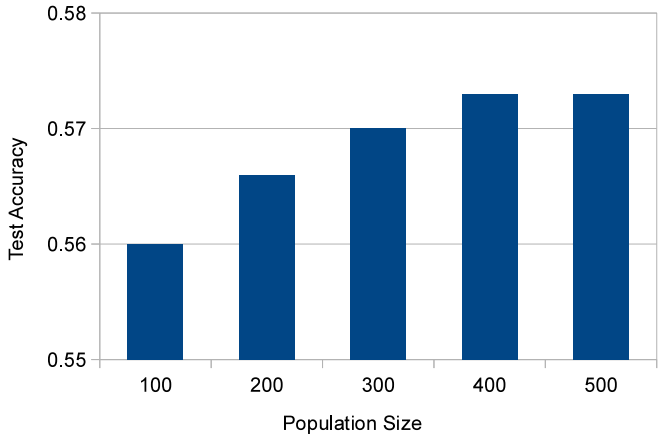


Figure 5.20: Classification accuracy on the output best hierarchy when varying population size.

Evolution algorithm. The purpose of this experiment is twofold: to test how much a large population contributes to finding a better solution, and to provide a fair comparison between HAA and our approach. Among the 100 trials, the best hierarchy found by HAA has a 74.3% accuracy on the test set (with an average accuracy of 69% across all trials). In comparison, the first trial of our algorithm found a best hierarchy with 80% accuracy. The result indicates that, although the contribution from the large population is clearly visible, it is the combination of the two factors that make our algorithm better than previous methods: a large population and the new genetic operators.

5.4 Discussion and Conclusion

Compared with previous approaches, our method explores a much larger space searching for better hierarchies. Although this enables us to find better solutions, it also brings significant cost. At each iteration, our approach evaluates thousands of hierarchies. Fortunately, the evaluation can be easily parallelized. Using a Condor [189] distributed computing platform running on around 100 nodes shared with multiple users, each iteration on the LSHTC-b dataset can be finished within approximately 2.5 hours in real elapsed time. For the additional improvement in classification performance, we consider this extra one-time cost worthwhile. The evaluation cost can be reduced using less expensive, approximate fitness evaluations. For example, smaller training and validation sets, and fast classifiers can be used in the evaluation process. Furthermore, since some generated hierarchies share common subtrees, trained models on such subtrees can be reused.

In this chapter, we proposed a hierarchy adaptation approach by using the standard mechanisms of a genetic algorithm along with special genetic operators customized for hierarchies. Unlike previous approaches which only keep the best hierarchy in the search process and modify the hierarchy locally at each step, our approach maintains population variety by allowing simultaneous evolution of a much larger population, and enables significant changes during evolution. Experiments on multiple classification tasks showed that the proposed algorithm can significantly improve automatic classification, outperforming existing state-of-the-art approaches. Our analysis showed that the variety in population and customized reproduction operators are important to improvement in classification performance.

One drawback of our approach is that the genetic operators select mutation points and split points purely at random. Smarter operators may select such points based on heuristic rules so that they are more likely to generate better hierarchies.

The choice of genetic operators is quite arbitrary and primitive. Although those operators are shown to be effective through our experiments, they are probably not the best or the only effective operators for the problem. Are there other operators that can work effectively on the problem? Are there better operators? Besides those discussed in this chapter, what other properties are shared among good operators? A comprehensive operator study is needed to answer these questions.

In our experiments, the parameters of the GA were arbitrarily assigned, and remained constant over the search process. Genetic algorithms with adaptive parameters (adaptive

GA) [178] may bring further improvement. In addition, if the solutions are confined to binary trees, it may change the speed of fitness evaluation and the rate of convergence. Therefore, additional modifications to our algorithm might be necessary.

Chapter 6

Enhancing Taxonomies by Providing Many Paths

6.1 Introduction

A taxonomy organizes concepts or topics in a hierarchical structure. The dmoz Open Directory Project[1] and the Yahoo! Directory[2] are two well known examples which, with the help of many human editors, organize what are considered to be good quality web pages into topical taxonomies. Figure 6.1 illustrates what dmoz ODP looks like. Besides being created and maintained manually, taxonomies can also be created by automated systems. Such systems often group objects together based on their syntactic similarities or distance in the feature space. Automatic taxonomy generation can reduce the enormous

[1] http://www.dmoz.org/

[2] http://dir.yahoo.com/

amount of human effort and thus speed up the process. However, automatically generated taxonomies do not always match a human's preexisting mental image about the "world"; and therefore are sometimes difficult for a human to understand.

A taxonomy (or hierarchy) is usually constructed based on supertopic-subtopic, or parent-child relationships. At any topic in the hierarchy, following a branch to get to a topic at a lower level means adding another constraint to its parent topic. When a user seeks information in a taxonomy, she usually starts from the root ("everything"), and narrows down the topic by choosing one child under the current topic until she finds what she needs. Unlike keyword search, seeking information in a taxonomy does not require the user to formulate her information need into search keywords. The names of topics (and sometimes descriptions, too) serve as guidance for the user.

One major drawback of taxonomies is that they require users to have the same view of the topics as the taxonomy creator. That is, when a user follows a top-down path to find the specific topic of her interest, she has to make choices along the constrained sequence that is present in the hierarchy. As a result, users who do not share that mental taxonomy

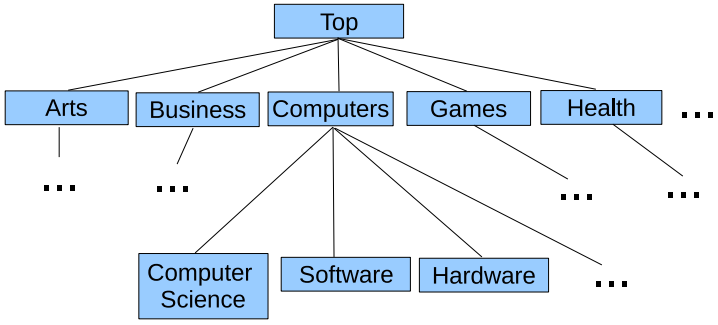


Figure 6.1: An illustration of ODP structure

are likely to have additional difficulties in finding the desired topic. For example, in one taxonomy, information about Emacs, an open source text editor, can be organized under /Software/OpenSource/Editors/Emacs. Such a taxonomy will not be helpful for a user who looks for information about Emacs but does not know that Emacs is an open source package. This problem can be somewhat reduced by remedies like cross-topic links (as used in ODP). However, after adding such links, the target nodes of the links will have more than one parent, making the taxonomy no longer a tree. Under this approach, nodes are not replicated—links are just added to the graph. Logically, such links are only appropriate when the alternative path also applies to all descendants of the topic. Furthermore, such links bring dependencies among topics and increase editing cost: editing one topic may result in changes in its linked topics and then the linked topics of those topics.

In this chapter, we propose a new approach to taxonomy expansion which is able to provide more flexible views. Based on an existing taxonomy, our algorithm finds possible alternative paths and generates a new, expanded taxonomy with flexibility in user browsing choices. In our experiments on the dmoz Open Directory Project and a social bookmarking dataset, the rebuilt taxonomies show favorable characteristics (more alternative paths and shorter paths to information). We define a flexible taxonomy as any reasonable taxonomy that can provide flexibility in user browsing choices. Our algorithm can be used to expand current web hierarchies such like those provided by ODP and Yahoo!.

Our main contributions include:

- an analysis and formulation of existing problems of hypernym/hyponym taxonomies;

and

- a new approach to generate flexible taxonomies that is able to work on existing taxonomies.

The rest of this chapter is organized as follows. We motivate our work and define the problem we aim to tackle in Section 2. In Section 3, we describe our approach in detail. Experimental results are shown in Section 4. We conclude with a discussion in Section 5.

6.2 Motivation and Problem Definition

6.2.1 Motivation

Browsing through a large data collection that is organized in a taxonomy is an interesting while different problem from keyword search. Search is effective when the user knows the name of the target information. When the desired information is a large set of instances (e.g., American sci-fi movies from the 1960s), or something that the user do not recall its name (e.g., the 1980 Steven Spielberg film about a space alien), a generic search often gives poor results. Browsing (or navigating) in taxonomies is usually more effective in such cases.

Typically it is supertopic-subtopic relationships that connect a topic with its child topics in a taxonomy. As natural it may seem, this methodology often poses extra difficulty for both the creator and the user. One reason is that there are often many ways to split a topic. For example, movies can be classified by their genre, country of origin, director,

etc. When creating a taxonomy of movies, one may choose to first split by genre then by country of origin, or first by director then by genre. However, no property of any of the above aspects (or facets) by itself is intrinsically a subtopic of another. In fact, they are orthogonal characteristics to some degree. A user may wish to narrow his selection by choosing one property of any reasonably arbitrary facet. Therefore, although there are many reasonable ways to split the movie category, none of them is able to satisfy all users' needs. We will refer to this problem as the *multiple facets* problem.

There are at least two ways to address the multiple facets problem. One is to split the topic in question in multiple ways. However, this method, without any fix, will inevitably result in duplicate objects under different subtopics. A simple fix is to treat one group of subtopics as real topics, and others as symbolic links. Figure 6.2 illustrates a taxonomy generated using this method, which is an actual subgraph of ODP. This method is widely used in ODP to address the multiple facets problem. However, to the best of our knowledge, this method is only used by human editors. No automated system is able to generate taxonomies in similar ways.

Another method to address the multiple facets problem is faceted browsing. Unlike browsing through a tree structure, choosing one option at a time, users are presented with multiple orthogonal facets and offered a chance to narrow the selection by choosing property constraints from multiple facets simultaneously. This method, when implemented properly, is able to address the multiple facets problem well. However, identifying the facets, especially from large datasets, is often a challenge for both human and automated

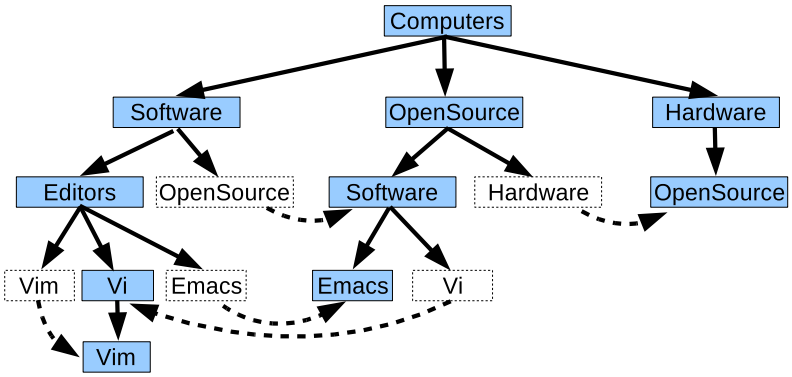


Figure 6.2: A taxonomy with symbolic links.

systems and still may have too many facets (or even classes of facets) to present at one time. While in this chapter we will focus on automating the multiple split method, the results of our approach may be of help to human and automated systems to identify facets and enhance faceted search/browsing.

Another problem of existing taxonomies also originates from the strict supertopic-subtopic relationships. When users need to find a particular object, they have to go all the way along the path from the root to their topic of interest. We will refer to this problem as *exhaustive path* problem. For example, even if the Emacs topic can be reached by two alternative paths (i.e., presumably with the multiple facets problem fixed), Software/Editors/OpenSource/Emacs and Software/OpenSource/Editors/Emacs, what if a user who does not know whether the package is open source just wants to find a list of representative text editors? In this case, a link like Software/Editors/Popular/Emacs will come in handy. By presenting popular descendant topics closer to the root, we may reduce the choices a user need to make and thus reduce the time to find desired information. We

are interested in back-end algorithms that can facilitate a better presentation.

If the user knows exactly what she is looking for, she can start from a keyword search in the taxonomy, in which case the above problems can be easily solved. However, most users who intend to find answers in a taxonomy usually do not know an appropriate keyword with which to start. The above problems could also be alleviated by building customized taxonomies on a user basis. However, the user profile that is required in a personalized approach is difficult to acquire. Furthermore, a user's interest and browsing preference may change frequently. Given the size of taxonomies we target, how to update the taxonomy to match the user's change in interests is another challenging problem in both efficiency and effectiveness. Therefore, instead of building personalized taxonomies for each user, we propose a method to build an expanded taxonomy with more branches to help users find information easier and faster in the given dataset.

6.2.2 Problem definition

As mentioned in previous sections, we aim to build a flexible hierarchy which can expose different interpretations of the data. A user traversing the hierarchy to find a target item can be viewed as a series of restrictions which narrow the search scope step by step. At each step, the user is provided a set of candidate topics to further narrow the scope. If we call the internal nodes in a taxonomy the *topics* (e.g., the topics in ODP), and call what is to be classified the *objects* (e.g., the outgoing links of ODP), we can formalize the problem as follows.

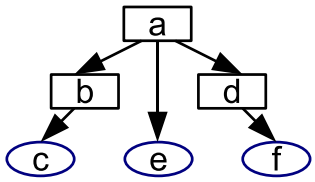


Figure 6.3: A simple predefined taxonomy

Given a graph $G = (V, E, O, R)$,

where $V = \{v$— v is a topic in the taxonomy$\}$,

$E = \{(v_1, v_2)|v_1, v_2 \in V\}$,

$O = \{o$— o is an object in the taxonomy$\}$,

$R = \{(v, o)|v \in V$, and $o \in O\}$;

Output a graph $G' = (V, E', O, R)$,

where $E' = \{(v_1, v_2)|v_1, v_2 \in V\}$.

Note that we do not require the input graph to be a tree in order to allow for more relaxed relationships, such as reference links across different branches.

6.3 Approach

So far, we have formulated the taxonomy generation problem as a graph transformation problem. We will solve this graph transformation problem in two steps: break down the taxonomy hierarchy into a set of tag-object tuples and then rebuild a flexible taxonomy from these tuples.

First, we break down the input taxonomy into a set of tag-object tuples. We consider the original tree structure and treat each internal node (topic) as a tag. We assume the

same topic name always has the same meaning. That is, the same topic name appearing in different locations in the original taxonomy is considered to be the same tag. Then each object in the taxonomy (i.e., URLs in the ODP case) is associated with all the tags from its ancestors in the tree. For example, in the original hierarchy in Figure 6.3, there are three objects c, e and f, and three topics, a, b, and d. After we tag each object with its ancestors, we get $tag(e) = \{a\}$, $tag(c) = \{a, b\}$, $tag(f) = \{a, d\}$. We also overload this function to define the tag set of multiple objects as the union of their tag sets, i.e., $tag(\{e, c\}) = tag(e) \cup tag(c) = \{a, b\}$. We define $obj(x)$ as the set of objects that are associated with tag x. In this example, $obj(a) = \{c, f, e\}$, $obj(b) = \{c\}$, $obj(d) = \{f\}$. Similar to the tag set of multiple objects, we define the object set of multiple tags as the union of their object sets. In this case, $obj(\{b, d\}) = obj(b) \cup obj(d) = \{c, f\}$. After this, we discard the original taxonomy structure, with only the tags, objects, and their relationships left. Since there are no relationships between two tags or two objects, the problem left for consideration now is a bipartite graph, with tags and objects being two sets of nodes, and each edge connecting exactly one tag and one object. Therefore, we now have

$$G = (T, O, E)$$

$$T = \{t | t \text{ is a tag}\}$$

$$O = \{o | o \text{ is an object}\}$$

$$E = \{(t, o) | t \in T, o \in O, \text{ object } o \text{ has tag } t\}$$

We now transform the taxonomy rebuilding problem into a problem to generate a

taxonomy based on a bipartite graph of tags and objects.

Hierarchy problem

Given a bipartite graph $G = (T, O, E)$,

Output a graph $G'' = (V'', E'', O, R'')$

where V'' is a set of topics, E'' is a set of edges connecting topics in V'' to objects in O, and R'' is a set of edges connecting topics to their subtopics.

Although such a problem transformation process seems tedious and arguably unnecessary, we will show in the following that the transformation allows us to adapt solutions to a well-studied problem for our purpose of taxonomy generation.

6.3.1 Set covering

A straightforward approach to taxonomy generation is a top-down method, in which the topic at each leaf node is split into a number of subtopics until no further split is necessary. Every time a topic needs to be split, if we consider the topics (tags) as the sets, and the associated objects as the objects to be covered, the split problem can be seen as a generalized set covering problem.

In a set covering problem, given a universe O consisting of n objects, and T a set of subsets of O, we say a subset $C \subseteq T$ covers O iff $O = \bigcup_{t \in C} t$. The set covering problem is known to be NP-complete. However, there are greedy approximation algorithms with polynomial time complexity (e.g., Algorithm 1 in [195]). Let H_k denote $\sum_{i=1}^{k} 1/i \approx \ln k$, where k is the largest set size. This algorithm is guaranteed to return a set cover of weight

at most H_k times the minimum weight of any cover.

Algorithm 6.1 Greedy Algorithm for Set Cover (Vazirani, 2001)

```
1: // Input T is a set of tags, and O is a set of objects.
2: Initialize C ← ∅.
3: // Loop while C does not cover all objects
4: while Obj(C) ≠ O do
5:    Choose t ∈ T to maximize f(t)
6:    Let C ← C ∪ {t}
7: end while
8: Return C
```

$f(t)$ is the objective function which the algorithm aims to maximize. In the original version of the set covering algorithm [195], it is defined as $|Obj(C \cup \{t\}) - Obj(C)|$. We choose to base our approach on this algorithm for two main reasons. First, it is a natural requirement for the generated hierarchy to cover all the objects. Second, in order to generate our desired hierarchy, it is convenient just to extend its objective function to match our purpose, leaving the rest of the algorithm unchanged. In the rest of this section, we will show how we change this function to meet our needs in flexible taxonomy generation.

In our approach, every time a topic is split, we generate three types of subtopics. A group of basic subtopics to cover all the associated objects (in some sense, the most obvious split), one or more groups of orthogonal subtopics to provide alternative paths, and a group of popular subtopics to allow fast access to the most popular descendant topics. We will discuss them in the following subsections.

6.3.2 The basic group

The original bipartite graph consists of the tags and objects as the two groups of vertices, and the relationships among them as the edges. Starting from a tree with a single node *root*, we iteratively split its leaf nodes to generate the taxonomy.

Suppose we are at the point to split a node (tag) t_{cur}, let T_a be the set of tags that has been used in t_{cur}'s ancestor nodes. The tag set $T' = T - T_a$, the corresponding object set O', and the relationship E' between T' and O' make up a bipartite graph $G'(T', O', E')$, a subgraph of the original bipartite graph $G(T, O, E)$. Let T_c be the subtopics that have already been chosen, O_c the set of objects that have been covered. The method of generating the basic group of subtopics is as follows.

Cover Score. For each $t \in T' - T_c$, the cover score is defined as follows.

$$CoverScore(t) = \frac{\sum_{o \in O' - O_c, and (t,o) \in E'} w(o)}{\sum_{o \in O'} w(o)} \tag{6.1}$$

The score is normalized so that for any t, $CoverScore(t) \in [0, 1]$. $w(o)$ is a weight of the object o. In this work, it is simply set as 1. In the future, it can be generalized to any importance measure of the object (e.g., the PageRank of the web page).

Tag Similarity Score. Define function F_v that maps a tag to an $|O|$ dimensional vector, in which the ith element of $F_v(t)$

$$F_v(t)_i = \begin{cases} 1 & if(t, o_i) \in E \\ 0 & otherwise \end{cases} \tag{6.2}$$

Then the similarity between two tags t_1 and t_2 is defined as

$$TagSim(t_1, t_2) = \frac{F_v(t_1) \cdot F_v(t_2)}{|O|} \tag{6.3}$$

At any time during the process of choosing subtopics, let T_c be the set of subtopics that have been chosen, then the similarity score of any tag t for consideration is

$$TagSimScore(t) = \begin{cases} \frac{|T_c|}{\sum_{t' \in T_c} TagSim(t,t')} & if|T_c| > 0 \\ 0 & otherwise \end{cases} \tag{6.4}$$

The cover score focuses on how many new objects are covered by adding the tag in question; it does not care about the overlap between the newly added tag and the existing tags. The tag similarity score compensates for that by favoring the tags with less overlap.

Based on the cover score and the tag similarity score, we define the objective function $f_b(t)$ as follows and plug it into Algorithm 1 to calculate the basic group of subtopics.

$$f_b(t) = \alpha \cdot CoverScore(t) + \beta \cdot \frac{\mu}{TagSimScore(t)} \tag{6.5}$$

μ is a normalization factor, which is set to $min_t[1/TagSimScore(t)]$ in our work. The parameters α and β are used to adjust the impact of different scores. We will tune them in the experiments.

6.3.3 The extension group

In the previous subsection, we discussed how to generate the first group of subtopics. In the extension group, we aim to discover orthogonal dimensions for alternative views. We exploit two types of information to compute this group.

Impurity Score. We employ the entropy impurity to measure this property of tags. Entropy impurity has been used in decision trees [19]. The impurity of tags in our system is defined as follows. Let O'_t be the set of objects which are covered by t, $O'_t = \{o|(t,o) \in E'$ and $o \in O'\}$. Given the basic tag set T_b generated by the method in Section 6.3.2, and a tag t, we define $P_t(t_i)$ as the fraction of objects for $t_i \in T_b$ which is

$$P_t(t_i) = \frac{\sum_{(t_i,o)\in E', o\in O'_t} w(o)}{\sum_{o\in O'_t} w(o)} \tag{6.6}$$

Then, the impurity of a candidate subtopic t given T_b is defined as

$$I(t) = -\sum_{t_i \in T_b} P_t(t_i) \log_2 P_t(t_i) \tag{6.7}$$

By definition, if all the objects covered by t are uniformly covered by the topics in T_b, it will get the maximum value.

Sibling Score. After a set of subtopics are chosen, the sibling score of a candidate topic t measures how likely t is a sibling topic with the chosen tags. The sibling likelihood can be obtained from various sources, such as Wikipedia, WordNet, or the generic web. In this work, we obtain this information from the original taxonomy. For each pair of tags, we

examine the positions of the two tags within the original hierarchy and count the number of times that these two tags share the same parent. We define the function $parent(t)$ as the the set of parent topics of t. The sibling function between any two tags is defined as follows.

$$Sibling(t_1, t_2) = \log_2 (1 + |parent(t_1) \cap parent(t_2)|) \tag{6.8}$$

Then, $Sibling(t_1, t_2)$ is normalized over all tag pairs so that they sum to 1. We note the normalized results as $Sibling_{norm}(t_1, t_2)$.

Based on the pairwise sibling likelihood, the sibling score of a candidate subtopic t given a set of already chosen subtopics T_c is computed as follows. For any t, $SiblingScore(t) \in [0, 1]$.

$$SiblingScore(t) = \begin{cases} \frac{\sum_{t' \in T_c} Sibling_{norm}(t,t')}{|T_c|} & if |T_c| > 0 \\ 0 & otherwise \end{cases} \tag{6.9}$$

Similar to the generation of the basic group, when generating the extension groups, we still use the greedy approximation algorithm of set covering problem, only with a different objective function.

$$f_e(t) = \gamma \cdot CoverScore(t) + \delta \cdot SiblingScore(t) + \eta \cdot I(t) \tag{6.10}$$

The weights γ, δ, η are used to adjust the impact of different scores. We will tune them in the experiments.

Depending on the particular situation, the extension set may need many topics to cover all objects. In order not to overwhelm users with too many choices, we slightly change the

stopping criterion to let it be satisfied when the number of subtopics exceeds k, where k is set to be $1.5 \times |T_b|$.

Above we presented how to generate one group of extension topics. After this, a new group of extension topics can be generated by removing the selected topics from the candidate, and running the above process again.

6.3.4 The shortcut group

We discussed the generation of the extension groups in the previous subsection. The extension groups aim to address the multiple facets problem. In this subsection, we discuss how to address the exhaustive path problem using a group of shortcut topics.

Popularity Score. The idea of the shortcut group is to increase the visibility of popular/important topics by putting them closer to the root, and thus reduce the time for users to find them. There are a variety of metrics to assess whether a descendant topic should be promoted. Here, we use the number of times the topic name is used as a tag for bookmarks in delicious. The popularity score is defined as follows:

$$popularity(t) = \frac{|bookmark(t)|}{\max_{t \in T} |bookmark(t)|} \tag{6.11}$$

where $|bookmark(t)|$ is the number of distinct bookmarks which are associated with tag t in Delicious. The value of popularity also is normalized to fit in the range [0,1].

We again use Algorithm 1 but with a different objective function to generate the

shortcut group:

$$f_s(t) = popularity(t) \tag{6.12}$$

Although the data in Delicious changes continuously, it is safe to assume that the relative frequencies of tags are somewhat stable over time. Given that the Delicious data is only used as a *rough* estimate of tag popularity/importance, it is not necessary to always keep it up-to-date. In addition, such a measure can also be estimated from other sources like Wikipedia, query log, and click log of a web hierarchy. Once there is a significant change in tag popularity, the hierarchy can be regenerated by running the algorithm on the updated data. Although directly linking a topic to its indirect descendants may break the logical consistency, we expect this method can help a user to find popular topics faster. Furthermore, putting such shortcuts in a separate group can reduce any user-perceived confusion.

6.4 Experiments

6.4.1 Datasets and experimental setup

We used an ODP dump from April 2009 to test our algorithm. It contained 4,225,962 external links and 763,377 categories. We consider categories with the same name to be the same tag, which gives us 292,550 tags. In the original ODP hierarchy, the average depth is 7.25 and maximum depth is 14.

We used two subsets of ODP to experiment with our approach. One is

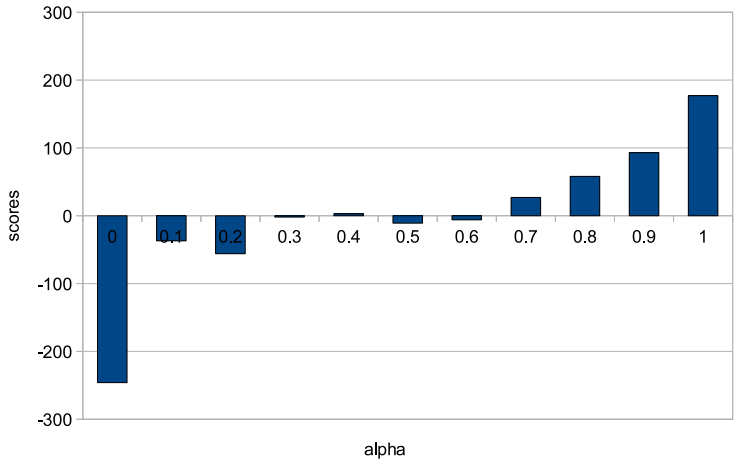


Figure 6.4: User evaluation result when tuning parameter to generate the basic set.

"Top/Computers/Computer_Science/" and all its subcategories and objects, which contains 184 unique tags and 2061 URLs. The other contains "Top/Science/" and all its subcategories and objects, with 9415 tags and 100,109 URLs. We also collected popularities for 31,342 tags from Delicious.

6.4.2 Parameter tuning

So far, we have introduced a taxonomy generation method with parameters. α and β are used to combine the cover score and tag similarity score when generating the basic set. We also used γ, δ, and η to balance the weight of cover score, sibling score, and impurity score. Here we tune the parameters in our method before generating the final taxonomy.

We changed the value of α and β by 0.1 in each step while maintaining their sum to be one. Thus we generated 11 taxonomies. We asked users to conduct pairwise comparison of the subtopics under one particular category at a time, and record their rating. Users can

choose to rate quality of subtopics in one taxonomy better than, worse than, or almost the same as another. They can also choose not to make a judgment when uncertain. When a user considers the subtopics in one taxonomy better than another, we add one point to the former taxonomy's score, and subtract one from the latter. Nine users participated in the evaluation. The average scores across users are plotted in Figure 6.4, showing a clear trend that the more weight we put on covering score, the better result we get. The best result is achieved when only using covering score.

We continue using this approach to tune the parameters to generate the extension set. We fixed $\alpha = 1$ and $\beta = 0$, then changed each of the other parameters by 0.1 at each step, while maintaining the sum to be one. Although there are significant difference in user ratings from different parameter settings, the result did not show any clear pattern. The best score is achieved when $\gamma = 0.8$, $\delta = 0.1$, and $\eta = 0.1$.

6.4.3 Taxonomy comparison and analysis

After the parameters are tuned, we applied our algorithm on the two subsets of ODP using the best parameter setting. Figures 6.5 and 6.6 illustrate the hierarchies generated by our algorithm for "Top/Computers/Computer_Science" and "Top/Science", respectively. From the results, we can see our automatically generated hierarchies are reasonable. For comparison, we implemented the approach proposed by Heymann and Garcia-Molina [88], which will be referred to as "Communal Taxonomies". The generated hierarchy is illustrated in Figure 6.7. The algorithm output depends on a tag similarity threshold. We

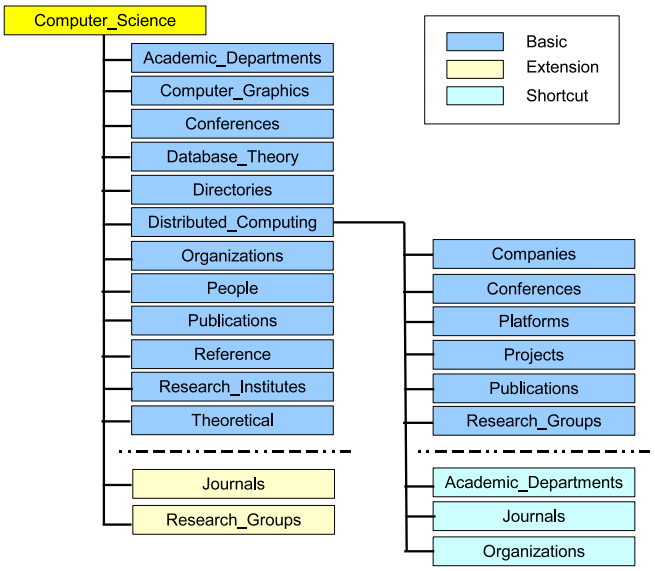


Figure 6.5: Automatically generated subtree under Top/Computers/Computer_Science

tried our best to tune it in order to get the best result. However, the generated hierarchy does not seem to be reasonable. It has 183 categories and 3 levels, with 25.9 subcategories per category on average. We think the following can provide one possible explanation. The "Communal Taxonomies" approach is designed for generating taxonomies out of flat tagging systems, as opposed to a tag-object dataset extracted from an existing hierarchy. It starts building the hierarchy from the center of the tag similarity graph. If the original hierarchy is dramatically unbalanced, the center may shift from the original root to its most developed subtree, resulting in a more balanced yet less reasonable taxonomy.

We expect that the subtree consisting of only the basic sets generated by our algorithm ("basic subtree") should be similar to the original ODP hierarchy. Our observation of the resulting hierarchies matches this hypothesis. In order to test it analytically, we implemented the taxonomy comparison metric called "taxonomy overlap" proposed by

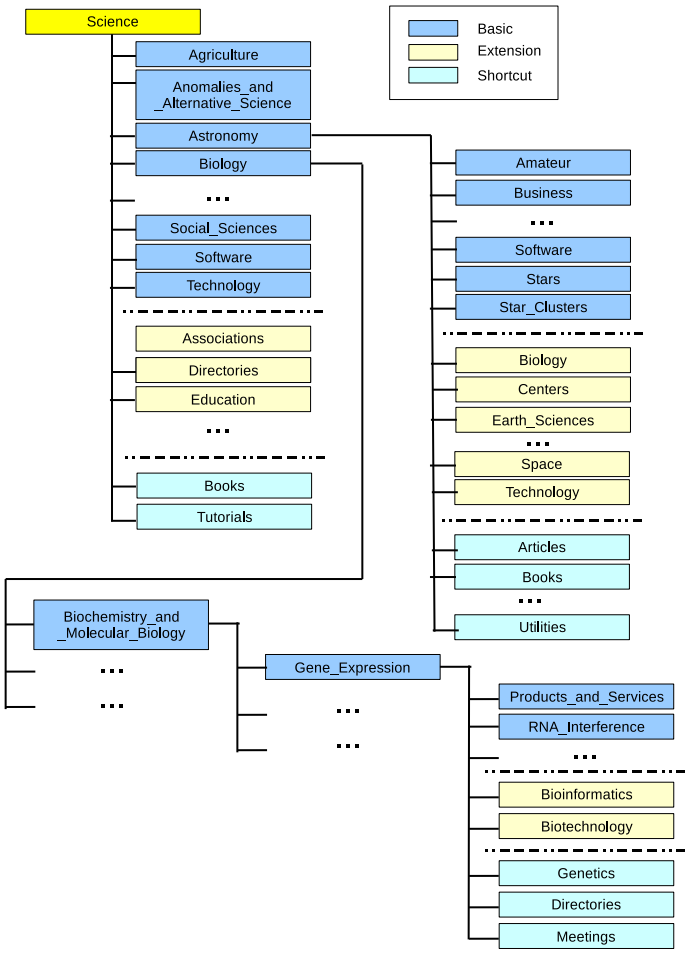


Figure 6.6: Automatically generated subtree under Top/Science

Maedche et al. [122]. The context of a concept in a taxonomy is defined by all its superconcepts and subconcepts. Then for each common concept across two taxonomies, its overlap is computed by the Jaccard similarity of its context in each taxonomy. The overall taxonomy overlap of two taxonomies is the average of individual overlap over all concepts. The value of taxonomy overlap ranges between 0 and 1, where a higher score means more similar. We computed the overlap between ODP and our basic subtree, as well as the

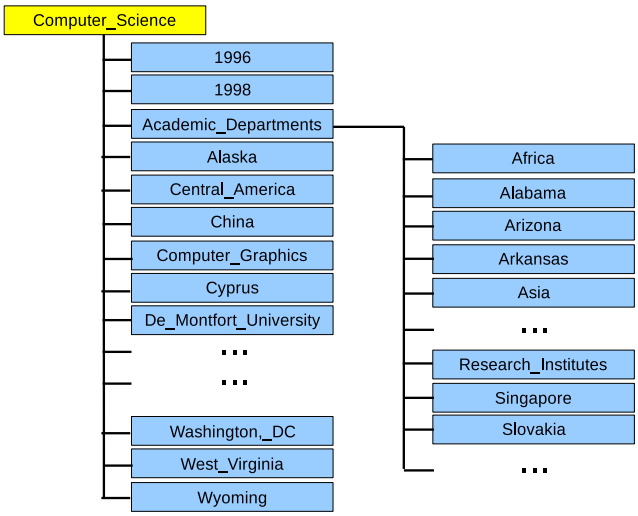


Figure 6.7: A subtree under Top/Computers/Computer_Science generated by "Communal Taxonomies"

overlap between ODP and the one generated by "Communal Taxonomies". The overlap between ODP and our basic subtree is 0.98, meaning they are extremely similar. The overlap is 0.77 for ODP and "Communal Taxonomies". One may question the necessity of the method generating the basic tag group. Given that the generated basic subtree highly resembles the original hierarchy, a simple alternative is to directly operate on the original hierarchy instead of regenerating it. However, the method proposed in Section 6.3.2 is useful for at least two reasons. First, we need to generate basic groups of subtopics under the extension group of topics, which are not present in the original tree. Second, such a method makes it possible to extend our approach to work on not only corpora with existing tree structures, but flat tagging systems like Delicious.

6.4.4 User studies

We conducted two types of user studies to evaluate user satisfaction of our hierarchy.

Algorithm 6.2 Reservoir Sampling with a Bias Towards High Level Topics

```
{INPUT: an array candidate[n] consists of n candidate topics;}
{ an integer k, the number of topics to be selected; and,}
{ a function getLevel, given a topic, returns its level.}
{OUTPUT: an array selected[k] consists of the k selected topics.}
for i = 1 to k do
    selected[i] = candidate[i]
end for
for i = k + 1 to n do
    r = random (i) {Generate a random integer between 1 and i, inclusively.}
    if r ≤ k and getLevel(candidate[i]) ≤ getLevel(candidate[r]) then
        selected[r] = candidate[i]
    end if
end for
```

In the first user study, we compare three hierarchies: our hierarchy, the ODP hierarchy and the ODP hierarchy with random extensions. The random extensions are generated by randomly selecting descendant categories and promote them as the direct children of the current category. This random selection algorithm is a variation based on reservoir sampling. While a generic random sampling method gives each candidate equal probability to be selected, our method favors descendant categories that are closer to the current category (i.e., more generic topics have higher chances to be selected). The algorithm is shown in Algorithm 6.2. This random extension is used as an alternative baseline than the original hierarchy to eliminate the possible effect that users may simply choose the hierarchy with more branches. In total, we randomly selected 27 categories from "Top/Computers/Computer_Science" and 50 categories from "Top/Science" to be

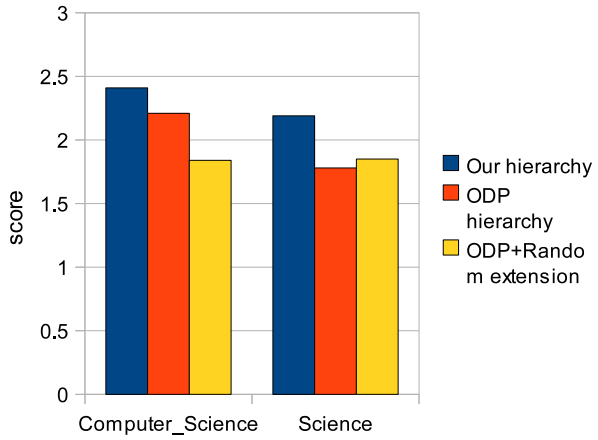


Figure 6.8: Results of user study on quality of hierarchies.

evaluated by users. For each category, we randomly reordered the sequence of these three hierarchies before presenting them to the users. For each of the selected categories, we only show a local view of each hierarchy, i.e., the current category and its children. Evaluators are asked if the quality of the child concepts are good(3), fair(2), or bad(1).

Figure 6.8 shows the average score for each of the hierarchies. Our hierarchy is considered to be much better than the random extension hierarchy, and is even better than the manually created hierarchy from ODP. In the "Top/Science" dataset, our extension branches give users positive impressions and the satisfaction score judged by users is improved by 23% when compared with the original ODP. Similar results can be found on the "Top/Computers/Computer_Science" dataset.

We also designed specific tasks, asking users to find particular topics in the generated taxonomies. There are two main types of tasks. The first is to find the subtopics under some given topics. The second is to find a specific topic. Three users participated in this

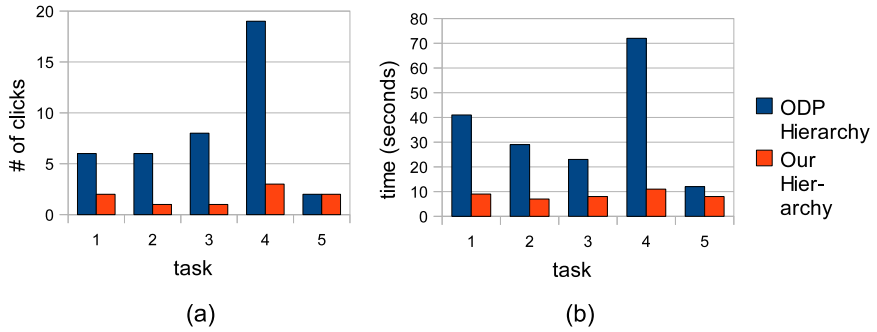


Figure 6.9: Average number of click counts and time for users to find designed items

experiment.

For the first kind of task, under a given level, if there is the target topic presented as a choice, the user can just click it and enter into the subtopic list. However, if there is no such alternative choice, the only way to find the subtopics of this topic is to try every possible branch, and find whether these branches contain this topic. In this case, our algorithm will be very helpful. This is because our algorithms can mine this kind of topic, which is contained by several other topics (these can be treated as potential subtopics of the current topic).

In the second kind of task, the user should find the exact topics. Our algorithms can provide extra paths to give the user more options. So, intuitively, the user should find the topic faster on average. when a user tries to find a certain topic, he must formulate a model of how to find it. For example, the user is trying to find C, and $A->B->C$ is his original idea. However, the system only provides $B->A->C$, forcing the user to discover this structure and change his mental model of the topic structure. Our algorithms can additionally provide $A->B->C$ to help fit this kind of user. Thus, on this task,

our algorithms can also help users find what they want faster.

In our experiment, we design five tasks, which are

1. Find software related links.
2. Find fields which contain research groups in this web site.
3. Find journal related links.
4. Find theoretical publications.
5. Find conferences in 2008.

Task 1, task 2, and task 3 belong to the first type of task (subtopic task); task 4 and task 5 belong to the second type of task (specific topic task). Figure 6.9(a) shows the number of clicks of users for each task and Figure 6.9(b) shows the elapsed time of users for each task. We can see that the two figures are very similar; that is, if the number of clicks is higher, the elapsed time is also higher. For the subtopic task (task 1, task 2 and task 3), our hierarchy can outperform the original ODP hierarchy consistently. These results verify our hypothesis. The comparison of click counts for task 4 and 5 are extremely consistent. In task 4, in ODP hierarchy, the "Publication" is under some subtopics of "Theoretical", such as "Complexity_Theory". So if users want to find the "Publication" of "Theoretical", they have to try subtopics to find "Publication". In our hierarchy, we have promoted "Publication" to the same level of "Theoretical". In task 5, our algorithm provides one extra path to the target, but the original hierarchy already provides a path

which can be easily found by users. So the used time and the number of clicks for two hierarchies are similar. Note that these tasks are designed to demonstrate the advantages of our generated taxonomy. We expect the difference to be less dramatic on more generic tasks.

6.5 Discussion and Conclusion

In conclusion, we proposed a new model to automatically expand an existing taxonomy by providing more paths. Our experiments show that our approach is able to generate a more flexible and comprehensive hierarchy from an existing hierarchy, leading to significant reductions in user effort and time for hierarchy-centric task completion.

In this work, we assumed that category names are unique. However, this is not always the case; a word may have different meanings in different contexts. Polysemy detection and disambiguation may be of help in this situation. Another issue is that people may use different terms (synonyms) to express the same meaning; our current approach does not take such situations into account.

In the future, we plan to adjust our method so that it can generate flexible taxonomies out of a collections of arbitrarily tagged documents like Delicious and BibSonomy. In addition, with the help of appropriate tag suggestion methods, we plan to apply this idea to any generic collections where tags may not be already available.

Chapter 7

Conclusion and Future Work

In this dissertation, we studied two important topics related to classification: web page classification and hierarchy adaptation. In this final chapter, we will first summarize the most important points of the dissertation, then point out important problems left to be solved by future research.

7.1 Summary

Web page classification, also known as web page categorization, is the process of assigning a web page to one or more predefined category labels. Many information management and retrieval tasks are dependent on classification, which makes web classification an important topic to study. On-page features like textual content, tags, and visual features can be used for web page classification. However, many pages do not contain enough information for a classifier to make a reasonable decision. Information from neighboring

pages makes a valuable supplement. When using information from neighbors, a generic assumption is made that the neighborhood of same-category pages share certain common characteristics. This assumption is made in almost all web classification work which uses neighbor information. Another assumption assumes stronger correlations between links and page categories: a page is more likely to be surrounded by pages with the same category. This assumption works well in practice for subject classification on broad topics.

After reviewing existing work in Chapter 2, we proposed two methods to enhance web page classification using neighbor information in Chapters 3 and 4. In Chapter 3, we proposed the Neighboring Algorithm, which uses the class or topic vector from four types of neighboring pages to help classification. The algorithm combines topic vectors or class assignments from neighbors and the target pages itself using a weighted combination. The weights are determined by tuning on real-world data with consideration of a variety of page properties including human label availability, host, number of paths to the target, and, the most important, neighbor type. As a result, the algorithm can significantly improve classification accuracy. We also found that sibling pages give a good indication of a page's topic and that intra-host links provide some benefit.

One drawback of the Neighboring Algorithm is that it considers all information on a page as a whole, without recognizing different values from different parts of a page. In Chapter 4, we proposed an enhanced algorithm called F-Neighbor which breaks a web page into several fields, recognizes the important fields and emphasizes them in the combination of the neighborhood. Experiments showed that F-Neighbor further improved

performance over the Neighboring Algorithm. We also found that although sibling pages are still valuable, titles of parent pages become a very useful signal.

In Chapter 5, we continue investigating methods to improve classification, with the subject focused on hierarchical classification. Most hierarchies used in hierarchical classification are created by and for humans, instead of being optimized for automatic classification. We proposed the Hierarchy Evolution Algorithm to adapt hierarchies for better classification accuracy using evolutionary computation methods. By keeping a large number of hierarchies under scrutiny, and encouraging variety by making significant changes to hierarchies, our method is able to find far better hierarchies for classification.

Besides constructing hierarchical classifiers, hierarchies are usually used to provide a well-structured organization of information for users to explore and navigate. In Chapter 6, we continue to focus on hierarchy adaptation, but with the aim changed from automatic classification to better facilitating user navigation. We analyzed a problem encountered by most existing hierarchies: incompatible mental models between users and creators. We extended a basic solution of the set covering algorithm by enriching its optimization function to include a variety of hierarchy-oriented properties. The proposed method is able to provide multiple, flexible paths, and thus generate expanded hierarchies for improved user experience.

After recapitulating the dissertation work, we briefly summarize the lessons we learned from the dissertation work.

- Web pages often contain inadequate information for classification. Information from

neighboring pages, when used appropriately, can improve classification significantly.

- In topical classification on broad topics, an assumption is often made that a page is likely to be surrounded by pages of the same topic. Many existing approaches, including the approaches proposed in this dissertation, are based on this assumption.

- In general, text and human-generated labels from sibling pages are very useful in classification, as well as titles of parent pages.

- Hierarchical classification can be more accurate than flat classification if appropriate hierarchies are used. Using poor quality hierarchies can yield classification accuracy even worse than flat classification.

- The huge number of possible hierarchies makes it prohibitively expensive for an exhaustive search. A good search strategy should keep a significant number of hierarchies under scrutiny (as opposed to only a few), and encourage variety by making significant changes to hierarchies (as opposed to small modifications).

- Existing human-generated hierarchies, when used as an exploration mechanism for users, often suffer from the problem of incompatible mental models of the domain between users and creators. Expanding such hierarchies with selected additional paths can alleviate the problem, and better facilitate user navigation.

7.2 Future Work

Although a variety of research has been performed in the area of web classification and hierarchy adaptation, some important problems still remain unsolved.

- How much do text and link similarity measures reflect the semantic similarity between documents? Although the work by Davison [52] and by Menczer and colleagues [126, 123] cast some light on this question, a more definitive answer requires further study.

- Information from neighboring nodes is valuable. But such information is also noisy, and so all neighbors (even of the same type) are unlikely to be equally valuable. How might neighbors (or portions of neighbors) be weighted or selected to best match the likely value of the evidence provided?

- Hyperlink information often encodes semantic relationships along with voting for representative or important pages. Would the complete integration of content information into link form be beneficial? This could be performed with various types of artificial links as we have surveyed, or in a combined model of underlying factors, as in the combination of PHITS and PLSA [45] for web information retrieval.

- The lack of a standardized dataset, especially one with the spatial locality representative of the Web, is a significant disadvantage in web classification research. How can a truly representative dataset with these properties that is multiple orders of magnitudes smaller than the actual Web be selected?

- In Chapter 2, we made a distinction between single-label and multi-label classification. Most existing work focused on single-label classification. However, multi-label classification should not be considered a simple extension. For example, the correlation among categories can be exploited to improve performance [222]. Other questions remain. How does the degree of such correlations affect the classification? Are the metrics used to evaluate single-labeled classification also reasonable for multi-label?

- Search engine spam [26] is a significant concern in web information retrieval. What effect does web spam have on topical or functional classification?

- A fast, accurate estimate of a classifier's performance will greatly benefit hierarchy adaptation processes, as well as many other tasks. Is there a light-weight method to effectively estimate a classifier's performance without training and testing the classifier?

- Faceted search/browsing is an efficient method to organize and explore information in certain domains where facets are well-recognized by human. Is faceted browsing viable in any arbitrary domain, or on the generic web where well-defined facets are not available? An automated approach to facet identification and extraction will be valuable in such scenarios.

It is expected that solutions or even a better understanding of these problems may lead to the emergence of more effective web classification systems, as well as improvements in

other areas of information retrieval and web mining.

Bibliography

[1] K. Aas and L. Eikvil. Text categorisation: A survey. Technical report, Norwegian Computing Center, P.B. 114 Blindern, N-0314, Oslo, Norway, June 1999. Technical Report 941.

[2] E. Amitay. Using common hypertext links to identify the best phrasal description of target web documents. In *Proceedings of the SIGIR'98 Post-Conference Workshop on Hypertext Information Retrieval for the Web*, Melbourne, Australia, 1998.

[3] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: Detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, pages 38–47, New York, NY, 2003. ACM Press.

[4] R. Angelova and S. Siersdorfer. A neighborhood-based approach for clustering of linked document collections. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 778–779, New York, NY, 2006. ACM Press.

[5] R. Angelova and G. Weikum. Graph-based text classification: Learn from your neighbors. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 485–492, New York, NY, 2006. ACM Press.

[6] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. WebWatcher: A learning apprentice for the World Wide Web. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*, Stanford University, Mar. 1995. AAAI Press.

[7] A. P. Asirvatham and K. K. Ravi. Web page classification based on document structure. Awarded second prize in National Level Student Paper Contest conducted by IEEE India Council, 2001.

[8] G. Attardi, A. Gulli, and F. Sebastiani. Automatic web page categorization by link and context analysis. In *Proc. of the European Symposium on Telematics, Hypermedia and Artificial Intelligence*, pages 105–119, 1999.

[9] K. Balog and M. de Rijke. Finding experts and their details in e-mail corpora. In *Proceedings of the 15th International Conference on the World Wide Web*, pages 1035–1036. ACM, 2006.

[10] E. Baykan, M. Henzinger, L. Marian, and I. Weber. Purely URL-based topic classification. In *Proceedings of the 18th International Conference on World wide web*, pages 1109–1110, New York, NY, USA, 2009. ACM.

[11] E. Baykan, M. Henzinger, L. Marian, and I. Weber. A comprehensive study of features and algorithms for url-based topic classification. *ACM Transactions on the Web*, 5:15:1–15:29, July 2011.

[12] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 407–415, 2000.

[13] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Using manually-built web directories for automatic evaluation of known-item retrieval. In *Proceedings of the 26th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–374. ACM Press, July 2003.

[14] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and D. A. Grossman. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *Proceedings of the 12th ACM International Conference on Information and Knowledge Management*, pages 17–23. ACM Press, 2003.

[15] P. N. Bennett, S. T. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2005.

[16] P. N. Bennett and N. Nguyen. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–18. ACM, 2009.

[17] B. Berendt and C. Hanser. Tags are not metadata, but "just more content" - to some people. In *Proceedings of International Conference on Weblogs and Social Media*, 2007.

[18] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, New York, NY, 1998. ACM Press.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1984.

[20] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 231–238, New York, NY, July 2007. ACM Press.

[21] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 559–566, New York, NY, USA, 2007. ACM.

[22] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 231–238, New York, NY, USA, 2007. ACM.

[23] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 78–87, New York, NY, USA, 2004. ACM.

[24] P. Calado, M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M. A. Goncalves. Combining link-based and content-based methods for web document classification. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, pages 394–401, New York, NY, 2003. ACM Press.

[25] A. Cardoso-Cachopo and A. L. Oliveira. An empirical comparison of text categorization methods. In *Proceedings of the 10th International Symposium on String Processing and Information Retrieval*, volume 2857 of *LNCS*, pages 183–196, Berlin, Oct. 2003. Springer.

[26] C. Castillo and B. D. Davison. Adversarial web search. *Foundations and Trends in Information Retrieval*, 4(5):377–486, 2010.

[27] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, 2007. ACM Press. In press.

[28] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, CA, 2003.

[29] S. Chakrabarti, B. E. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[30] S. Chakrabarti, M. M. Joshi, K. Punera, and D. M. Pennock. The structure of broad topics on the web. In *Proceedings of the 11th International World Wide Web Conference*, pages 251–262. ACM Press, May 2002.

[31] S. Chakrabarti, M. van den Berg, and B. E. Dom. Focused crawling: A new approach to topic-specific Web resource discovery. In *Proceedings of the 8th International World Wide Web Conference*, pages 545–562, May 1999.

[32] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[33] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automated classification. In *Proceedings of the 6th International World Wide Web Conference*, Santa Clara, CA, Apr. 1997. Poster POS725.

[34] C. Chekuri, M. H. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. In *Proceedings of the 6th International World Wide Web Conference*, 1996.

[35] H. Chen and S. Dumais. Bringing order to the Web: Automatically categorizing search results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 145–152, New York, NY, 2000. ACM Press.

[36] Z. Chen, O. Wu, M. Zhu, and W. Hu. A novel web page filtering system by combining texts and images. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 732–735, Washington, DC, 2006. IEEE Computer Society.

[37] P.-J. Cheng, C.-H. Tsai, C.-M. Hung, and L.-F. Chien. Query taxonomy generation for web search. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 862–863, New York, NY, USA, 2006. ACM.

[38] P. Chesley, B. Vincent, L. Xu, and R. K. Srihari. Using verbs and adjectives to automatically classify blog sentiment. In N. Nicolov, F. Salvetti, M. Liberman, and J. H. Martin, editors, *Computational Approaches to Analyzing Weblogs: Papers from the 2006 Spring Symposium*, pages 27–29, Menlo Park, CA, March 2006. AAAI Press. Technical Report SS-06-03.

[39] S.-L. Chuang and L.-F. Chien. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, page 75, Washington, DC, USA, 2002. IEEE Computer Society.

[40] S.-L. Chuang and L.-F. Chien. Automatic query taxonomy generation for information retrieval applications. *Online Information Review*, 27(4):243–255, 2003.

[41] S.-L. Chuang and L.-F. Chien. Taxonomy generation for text segments: A practical web-based approach. *ACM Transactions on Information Systems*, 23(4):363–396, 2005.

[42] P. Cimiano, A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24:305–339, August 2005.

[43] P. Clough, H. Joho, and M. Sanderson. Automatically organising images using concept hierarchies. In *SIGIR Workshop on Multimedia Information Retrieval*, 2005.

[44] W. W. Cohen. Improving a page classifier with anchor extraction and link analysis. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1481–1488. MIT Press, Cambridge, MA, 2002.

[45] D. Cohn and T. Hofmann. The missing link—a probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems 13*, 2001.

[46] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 509–516, Menlo Park, CA, July 1998. AAAI Press.

[47] D. R. Cutting, D. R. Karger, J. O. Pederson, and J. W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the*

15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 318–329, 1992.

[48] N. Dai, B. D. Davison, and X. Qi. Looking into the past to better classify web spam. In *Proceedings 5th International Workshop on Adversarial Information Retrieval on the Web*, pages 1–8. ACM, Apr. 2009.

[49] N. Dai, X. Qi, and B. D. Davison. Bridging link and query intent to enhance web search. In *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia*, June 2011.

[50] N. Dai, X. Qi, and B. D. Davison. Enhancing web search with entity intent. In *Proceedings of the 20th International World Wide Web Conference*, Apr. 2011.

[51] D. Davidov, E. Gabrilovich, and S. Markovitch. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 250–257, New York, NY, 2004. ACM Press.

[52] B. D. Davison. Topical locality in the Web. In *Proceedings of the 23rd Annual ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 272–279, July 2000.

[53] B. D. Davison. The potential of the metasearch engine. In *Proceedings of the Annual Meeting of the American Society for Information Science and Technology*, volume 41, pages 393–402, Providence, RI, Nov. 2004.

[54] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[55] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Articial Intelligence Research*, 2:263–286, 1995.

[56] I. Drost, S. Bickel, and T. Scheffer. Discovering communities in linked data by multi-view clustering. In *From Data and Information Analysis to Knowledge Engineering: Proceedings of 29th Annual Conference of the German Classification Society*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 342–349, Berlin, Mar. 2005. Springer.

[57] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, NY, 1973.

[58] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 256–263, New York, NY, 2000. ACM Press.

[59] E. Elgersma and M. de Rijke. Learning to recognize blogs: A preliminary exploration. In *EACL Workshop: New Text - Wikis and blogs and other dynamic text sources*, March 2006.

[60] M. Ester, H.-P. Kriegel, and M. Schubert. Web site mining: A new way to spot competitors, customers and suppliers in the World Wide Web. In *Proceedings of*

the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 249–258, New York, NY, 2002. ACM Press.

[61] M. J. Fisher and R. M. Everson. When are links useful? Experiments in text classification. In *Advances in Information Retrieval. 25th European Conference on IR Research*, pages 41–56. Springer, Apr. 2003.

[62] R. A. Fisher and F. Yates. *Statistical tables for biological, agricultural and medical research*. Oliver & Boyd, London, 1938.

[63] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: Social searching? In *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313, Philadelphia, PA, July 1997.

[64] J. Fürnkranz. Exploiting structural information for text classification on the WWW. In *Proceedings of the 3rd Symposium on Intelligent Data Analysis*, volume 1642 of *LNCS*, pages 487–497. Springer-Verlag, 1999.

[65] J. Fürnkranz. Hyperlink ensembles: A case study in hypertext classification. *Journal of Information Fusion*, 1:299–312, 2001.

[66] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of the 21st International Conference on Machine Learning*, page 41, New York, NY, 2004. ACM Press.

[67] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference for Artificial Intelligence*, pages 1048–1053, July 2005.

[68] E. Gabrilovich and S. Markovitch. Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1301–1306, Menlo Park, CA, July 2006. AAAI Press.

[69] E. Gabrilovich and S. Markovitch. Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization. *Journal of Machine Learning Research*, 8:2297–2345, 2007.

[70] B. Ganter, G. Stumme, and R. Wille. *Formal Concept Analysis: Foundations and Applications*. Springer, New York, NY, 2005.

[71] R. Ghani. Combining labeled and unlabeled data for text classification with a large number of categories. In *First IEEE International Conference on Data Mining*, page 597, Los Alamitos, CA, 2001. IEEE Computer Society.

[72] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *Proceedings of the 19th International Conference on Machine Learning*, pages 187–194, San Francisco, CA, 2002. Morgan Kaufmann.

[73] R. Ghani, S. Slattery, and Y. Yang. Hypertext categorization using hyperlink patterns and meta data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 178–185, San Francisco, CA, 2001. Morgan Kaufmann.

[74] N. S. Glance. Community search assistant. In *Artificial Intelligence for Web Search*, pages 29–34. AAAI Press, July 2000. Presented at the AAAI-2000 workshop on Artificial Intelligence for Web Search, Technical Report WS-00-01.

[75] E. Glover, D. M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 507–514. ACM, 2002.

[76] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake. Using web structure for classifying and describing web pages. In *Proceedings of the 11th International Conference on World Wide Web*, pages 562–569, New York, NY, 2002. ACM Press.

[77] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake. Using Web structure for classifying and describing Web pages. In *Proceedings of the 11th International Conference on the World Wide Web*, 2002.

[78] K. Golub and A. Ardo. Importance of HTML structural elements and metadata in automated subject classification. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries*, volume 3652 of *LNCS*, pages 368–378, Berlin, Sept. 2005. Springer.

[79] N. Gövert, M. Lalmas, and N. Fuhr. A probabilistic description-oriented approach for categorizing web documents. In *Proceedings of the 8th International Conference on Information and Knowledge Management*, pages 475–482, New York, NY, 1999. ACM Press.

[80] H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. In *18th International World Wide Web Conference*, pages 201–201, April 2009.

[81] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proceedings of the First International Workshop on Adversarial Information Retrieval*, May 2005.

[82] M. Hammami, Y. Chahir, and L. Chen. Webguard: Web based adult content detection and filtering system. In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, page 574, Washington, DC, 2003. IEEE Computer Society.

[83] S. M. Harabagiu, M. A. Pasca, and S. J. Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 292–298, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[84] T. H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the 11th International World Wide Web Conference*, pages 517–526. ACM Press, May 2002.

[85] T. H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

[86] X. He, H. Zha, C. H. Q. Ding, and H. D. Simon. Web document clustering using hyperlink structures. *Computational Statistics and Data Analysis*, 41(1):19–45, 2002.

[87] U. Hermjakob. Parsing and question classification for question answering. In *Proceedings of the ACL Workshop on Open-Domain Question Answering*, pages 1–6, July 2001.

[88] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University, April 2006.

[89] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: a repository of Web pages. *Computer Networks*, 33(1–6):277–293, 2000.

[90] T. Hofmann. The cluster-abstraction model: unsupervised learning of topic hierarchies from text data. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, pages 682–687, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[91] T. Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pages 289–296. Morgan Kaufmann, July 1999.

[92] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, New York, NY, USA, 1999. ACM Press.

[93] C.-C. Huang, S.-L. Chuang, and L.-F. Chien. Liveclassifier: Creating hierarchical text classifiers through web corpora. In *Proceedings of the 13th International Conference on World Wide Web*, pages 184–192, New York, NY, 2004. ACM Press.

[94] C.-C. Huang, S.-L. Chuang, and L.-F. Chien. Using a web-based categorization approach to generate thematic metadata from texts. *ACM Transactions on Asian Language Information Processing*, 3(3):190–212, 2004.

[95] R. Jäschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Proceedings of Knowledge Discovery in Databases: 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4702 of *Lecture Notes in Computer Science*, pages 506–514. Springer, 2007.

[96] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598, New York, NY, 2004. ACM Press.

[97] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge,

MA, 1998.

[98] T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of 18th International Conference on Machine Learning*, pages 250–257, Williams College, US, 2001. Morgan Kaufmann Publishers, San Francisco, US.

[99] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A tour guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 770–775. Morgan Kaufmann, Aug. 1997.

[100] M. Käki. Findex: Search result categories help users when document ranking fails. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 131–140, New York, NY, 2005. ACM Press.

[101] M.-Y. Kan. Web page classification without the web page. In *Proceedings of the 13th International World Wide Web Conference Alternate Track Papers & Posters*, pages 262–263, New York, NY, 2004. ACM Press.

[102] M.-Y. Kan and H. O. N. Thi. Fast webpage classification using URL features. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 325–326, New York, NY, 2005. ACM Press.

[103] S. Kiritchenko. *Hierarchical text categorization and its application to bioinformatics*. PhD thesis, University of Ottawa, 2005.

[104] C. Kohlschutter, P.-A. Chirita, and W. Nejdl. Utility analysis for topically biased PageRank. In *Proceedings of the 16th International Conference on World Wide Web*, pages 1211–1212, New York, NY, 2007. ACM Press.

[105] M. Kovacevic, M. Diligenti, M. Gori, and V. Milutinovic. Visual adjacency multigraphs - a novel approach for a web page classification. In *Proceedings of the Workshop on Statistical Approaches to Web Mining*, pages 38–49, Sept. 2004.

[106] R. Krishnapuram and K. Kummamuru. Automatic taxonomy generation: Issues and possibilities. In *Fuzzy Sets and Systems*, page 184. Springer, 2003.

[107] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th International Conference on World Wide Web*, pages 658–665, 2004.

[108] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[109] O. Kurland and L. Lee. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313, New York, NY, July 2005. ACM Press.

[110] O. Kurland and L. Lee. Respect my authority!: HITS without hyperlinks, utilizing cluster-based language models. In *Proceedings of the 29th Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval, pages 83–90, New York, NY, July 2006. ACM Press.

[111] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *Proceedings of the 10th International Conference on World Wide Web*, pages 150–161, New York, NY, 2001. ACM Press.

[112] O.-W. Kwon and J.-H. Lee. Web page classification based on k-nearest neighbor approach. In *Proceedings of the 5th International Workshop on Information Retrieval with Asian Languages*, pages 9–15, New York, NY, 2000. ACM Press.

[113] O.-W. Kwon and J.-H. Lee. Text categorization based on k-nearest neighbor approach for web site classification. *Information Processing and Management*, 29(1):25–44, January 2003.

[114] G. Leshed and J. J. Kaye. Understanding how bloggers feel: Recognizing affect in blog posts. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, pages 1019–1024, New York, NY, 2006. ACM Press.

[115] T. Li, S. Zhu, and M. Ogihara. Hierarchical document classification using automatically generated hierarchy. *Journal of Intelligent Information Systems*, 29:211–230, October 2007.

[116] C. Lindemann and L. Littig. Coarse-grained classification of web sites by their structural properties. In *Proceedings of the 8th ACM International Workshop on*

Web Information and Data Management, pages 35–42, New York, NY, 2006. ACM Press.

[117] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations Newsletter*, 7(1):36–43, 2005.

[118] W. Liu, G.-R. Xue, Y. Yu, and H.-J. Zeng. Importance-based web page classification using cost-sensitive SVM. *Advances in Web-Age Information Management*, 3739:127–137, 2005.

[119] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning*, Menlo Park, CA, Aug. 2003. AAAI Press.

[120] J. Luxenburger and G. Weikum. Query-log based authority analysis for web information search. In *Proceedings of 5th International Conference on Web Information Systems Engineering*, volume 3306 of *LNCS*, pages 90–101, Berlin, November 2004. Springer.

[121] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, May 2007.

[122] A. Maedche and S. Staab. Comparing ontologies - similarity measures and a comparison study. In *Proceedings of International Conference on Knowledge Engineering and Knowledge Management*, number 2473 in LNCS. Springer, 2002.

[123] A. G. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani. Algorithmic detection of semantic similarity. In *Proceedings of the 14th International World Wide Web Conference*, pages 107–116, Chiba, Japan, May 2005.

[124] H. Malik. Improving hierarchical svms by hierarchy flattening and lazy classication. In *Proceedings of Large-Scale Hierarchical Classification Workshop*, 2010.

[125] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367. ACM, 1998.

[126] F. Menczer. Mapping the semantics of web text and links. *IEEE Internet Computing*, pages 27–36, May/June 2005.

[127] R. Mihalcea and H. Liu. A corpus-based approach to finding happiness. In *Computational Approaches to Analyzing Weblogs: Papers from the 2006 Spring Symposium*, pages 139–144, Menlo Park, CA, March 2006. AAAI Press. Technical Report SS-06-03.

[128] G. A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[129] G. Mishne. Experiments with mood classification in blog posts. In *Workshop on Stylistic Analysis of Text for Information Access*, August 2005.

[130] G. Mishne and M. de Rijke. Capturing global mood levels using blog posts. In *Computational Approaches to Analyzing Weblogs: Papers from the 2006 Spring Symposium*, pages 145–152, Menlo Park, CA, March 2006. AAAI Press. Technical Report SS-06-03.

[131] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[132] D. Mladenic. Turning Yahoo into an automatic web-page classifier. In *Proceedings of the European Conference on Artificial Intelligence*, pages 473–474, 1998.

[133] D. Mladenic. Text-learning and related intelligent agents: A survey. *IEEE Intelligent Systems and Their Applications*, 14(4):44–54, Jul/Aug 1999.

[134] A. Möller, J. Dörre, P. Gerstl, and R. Seiffert. The TaxGen framework: Automating the generation of a taxonomy for a large document collection. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, volume 2, page 2034, 1999.

[135] T. Nanno, T. Fujiki, Y. Suzuki, and M. Okumura. Automatically collecting, monitoring, and mining Japanese weblogs. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pages 320–321, New York, NY, 2004. ACM Press.

[136] L. Nie, B. D. Davison, and X. Qi. Topical link analysis for web search. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 91–98, Aug. 2006.

[137] NIST. Text REtrieval Conference (TREC) home page. http://trec.nist.gov/, 2008.

[138] K. Nitta. Improving taxonomies for large-scale hierarchical classifiers of web documents. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1649–1652. ACM, 2010.

[139] S. Nowson. *The Language of Weblogs: A study of genre and individual differences.* PhD thesis, University of Edinburgh, College of Science and Engineering, June 2006.

[140] The dmoz Open Directory Project (ODP), 2009. `http://www.dmoz.org/`.

[141] H.-J. Oh, S. H. Myaeng, and M.-H. Lee. A practical hypertext catergorization method using links and incrementally available class information. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 264–271, New York, NY, 2000. ACM Press.

[142] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University, 1998. Available from http://dbpubs.stanford.edu/pub/1999-66. Accessed 29 March 2008.

[143] S.-B. Park and B.-T. Zhang. Large scale unstructured document classification using unlabeled data and syntactic information. In *Advances in Knowledge Discovery and Data Mining, 7th Pacific-Asia Conference*, volume 2637 of *LNCS*, pages 88–99, Berlin, Apr. 2003. Springer.

[144] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting Web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 54–61, Portland, OR, 1996. AAAI Press.

[145] X. Peng and B. Choi. Automatic web page classification in a dynamic and hierarchical way. In *Proceedings of the IEEE International Conference on Data Mining*, pages 386–393, Washington, DC, 2002. IEEE Computer Society.

[146] D. Petkova and W. B. Croft. Hierarchical language models for expert finding in enterprise corpora. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 599–608. IEEE Computer Society, 2006.

[147] J. M. Pierre. On the automated classification of web sites. *Linköping Electronic Articles in Computer and Information Science*, 6, February 2001. http://www.ep.liu.se/ea/cis/2001/001/.

[148] S. P. Ponzetto and R. Navigli. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the 21th International Joint Conference on Artificial Intelligence,* Pasadena, Cal., 14–17 July 2009, pages 2083–2088, 2009.

[149] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 1440–1445. AAAI Press, 2007.

[150] K. Punera, S. Rajan, and J. Ghosh. Automatically learning document taxonomies for hierarchical classification. In *Special interest tracks and posters of the 14th International Conference on World Wide Web*, pages 1010–1011. ACM, 2005.

[151] X. Qi and B. D. Davison. Knowing a web page by the company it keeps. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 228–237, Nov. 2006.

[152] X. Qi and B. D. Davison. Classifiers without borders: Incorporating fielded text from neighboring web pages. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 643–650, July 2008.

[153] X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys*, 41(2), Feb. 2009.

[154] X. Qi and B. D. Davison. Hierarchy evolution for improved classification. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2193–2196, October 2011.

[155] X. Qi, L. Nie, and B. D. Davison. Measuring similarity to detect qualified links. In *Proceedings of the Third International Workshop on Adversarial Information Retrieval on the Web*, pages 49–56, May 2007.

[156] X. Qi, D. Yin, Z. Xue, and B. D. Davison. Choosing your own adventure: automatic taxonomy generation to permit many paths. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1853–1856, October 2010.

[157] H. Qu, A. L. Pietra, and S. Poon. Automated blog classification: Challenges and pitfalls. In *Computational Approaches to Analyzing Weblogs: Papers from the 2006 Spring Symposium*, pages 184–186, Menlo Park, CA, March 2006. AAAI Press. Technical Report SS-06-03.

[158] S. R. Ranganathan. *Colon Classification*. Madras Library Association, 1933.

[159] S. R. Ranganathan. *Classification, Coding, and Machinery for Search*. UNESCO, Paris, 1950.

[160] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[161] D. Riboni. Feature selection for web page classification. In *Proceedings of the Workshop on Web Content Mapping: A Challenge to ICT*, 2002.

[162] M. Richardson and P. Domingos. The Intelligent Surfer: Probabilistic combination of link and content information in PageRank. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[163] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 42–49, New York, NY, USA, 2004. ACM Press.

[164] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, 1994.

[165] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6:420–433, 1976.

[166] G. Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Retrieval*. Prentice Hall, Englewood Cliffs, NJ, 1971.

[167] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, New York, 1983.

[168] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213, 1999.

[169] P. Schmitz. Inducing ontology from flickr tags. In *Proceedings of the Workshop on Collaborative Web Tagging*, 2006.

[170] F. Sebastiani. A tutorial on automated text categorisation. In *Proceedings of 1st Argentinean Symposium on Artificial Intelligence*, pages 7–35, 1999.

[171] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[172] K. Seki and J. Mostafa. An application of text categorization methods to gene ontology annotation. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–145, New York, NY, USA, 2005. ACM.

[173] P. Sen and L. Getoor. Link-based classification. Technical Report CS-TR-4858, University of Maryland, 2007.

[174] V. Shanks and H. E. Williams. Fast categorisation of large document collections. In *Proceedings of Eighth International Symposium on String Processing and Information Retrieval*, pages 194–204, Nov. 2001.

[175] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu, and W.-Y. Ma. Web-page classification through summarization. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 242–249, New York, NY, 2004. ACM Press.

[176] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. A comparison of implicit and explicit links for web page classification. In *Proceedings of the 15th International Conference on World Wide Web*, pages 643–650, New York, NY, 2006. ACM Press.

[177] S. Slattery and T. M. Mitchell. Discovering test set regularities in relational domains. In *Proceedings of the 17th International Conference on Machine Learning*, pages 895–902. Morgan Kaufmann, June 2000.

[178] M. Srinivas and L. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):656 –667, April 1994.

[179] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[180] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.

[181] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of tf-idf schemes for web pages using their hyperlinked neighboring pages. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, pages 198–207, New York, NY, USA, 2003. ACM.

[182] A. Sun and E.-P. Lim. Hierarchical text classification and evaluation. In *Proceedings of IEEE International Conference on Data Mining*, pages 521–528, Washington, DC, November 2001. IEEE Computer Society.

[183] A. Sun, E.-P. Lim, and W.-K. Ng. Web classification using support vector machine. In *Proceedings of the 4th International Workshop on Web Information and Data Management*, pages 96–99, New York, NY, 2002. ACM Press.

[184] A. Sun, M. A. Suryanto, and Y. Liu. Blog classification using tags: An empirical study. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, volume 4822 of *LNCS*, pages 307–316. Springer, 2007.

[185] A.-H. Tan. Text mining: The state of the art and the challenges. In *Proceedings of PAKDD Workshop on Knowledge Discoverery from Advanced Databases*, pages 65–70, 1999.

[186] S. Tan and Y. Wang. Combining error-correcting output codes and model-refinement for text categorization. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 699–700, New York, NY, USA, 2007. ACM.

[187] L. Tang, H. Liu, J. Zhang, N. Agarwal, and J. J. Salerno. Topic taxonomy adaptation for group profiling. *ACM Transactions on Knowledge Discovery from Data*, 1:1:1–1:28, February 2008.

[188] L. Tang, J. Zhang, and H. Liu. Acclimatizing taxonomic semantics for hierarchical content classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 384–393. ACM, 2006.

[189] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency and Computation: Practice and Experience*, 17:323–356, February 2005.

[190] Y. Tian, T. Huang, W. Gao, J. Cheng, and P. Kang. Two-phase web site classification based on hidden markov tree models. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, page 227, Washington, DC, USA, 2003. IEEE Computer Society.

[191] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, Nov. 2001.

[192] K. Toutanova, F. Chen, K. Popat, and T. Hofmann. Text classification in a hierarchical mixture model for small training sets. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 105–113, New York, NY, USA, 2001. ACM.

[193] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[194] H. Utard and J. Fürnkranz. Link-local features for hypertext classification. In *Semantics, Web and Mining: Joint International Workshops, EWMF and KDO*, volume 4289 of *LNCS*, pages 51–64, Berlin, Oct. 2005. Springer.

[195] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[196] C. Veres. The language of folksonomies: What tags reveal about user classification. In *Natural Language Processing and Information Systems*, volume 3999 of *LNCS*, pages 58–69, Berlin / Heidelberg, July 2006. Springer.

[197] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, Jan. 2002.

[198] W. Wibowo and H. E. Williams. Simple and accurate feature selection for hierarchical categorisation. In *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 111–118, New York, NY, 2002. ACM Press.

[199] W. Wibowo and H. E. Williams. Strategies for minimising errors in hierarchical web categorisation. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 525–531, New York, NY, 2002. ACM Press.

[200] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[201] X. Xiao, Q. Luo, X. Xie, and W.-Y. Ma. A comparative study on classifying the functions of web page blocks. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 776–777, 2006.

[202] D. Xing, G.-R. Xue, Q. Yang, and Y. Yu. Deep classifier: automatically categorizing search results into large-scale hierarchies. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 139–148. ACM, 2008.

[203] Z. Xu, I. King, and M. R. Lyu. Web page classification with heterogeneous data fusion. In *Proceedings of the 16th International Conference on World Wide Web*, pages 1171–1172, New York, NY, USA, 2007. ACM.

[204] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 619–626. ACM, 2008.

[205] G.-R. Xue, Q. Yang, H.-J. Zeng, Y. Yu, and Z. Chen. Exploiting the hierarchical structure for link analysis. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193, New York, NY, Aug. 2005. ACM Press.

[206] G.-R. Xue, Y. Yu, D. Shen, Q. Yang, H.-J. Zeng, and Z. Chen. Reinforcing web-object categorization through interrelationships. *Data Mining and Knowledge Discovery*, 12(2-3):229–248, 2006.

[207] Yahoo!, Inc. Yahoo! developer network. `http://developer.yahoo.com/`, 2006.

[208] Yahoo!, Inc. Yahoo! `http://www.yahoo.com/`, 2007.

[209] J. Yan, N. Liu, B. Zhang, S. Yan, Z. Chen, Q. Cheng, W. Fan, and W.-Y. Ma. OCFS: Optimal orthogonal centroid feature selection for text categorization. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 122–129, New York, NY, USA, 2005. ACM.

[210] C. C. Yang and N. Liu. Web site topic-hierarchy generation based on link structure. *Journal of the American Society for Information Science and Technology*, 60(3):495–508, 2009.

[211] H. Yang and T.-S. Chua. Effectiveness of web page classification on finding list answers. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 522–523, New York, NY, 2004. ACM Press.

[212] H. Yang and T.-S. Chua. Web-based list question answering. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1277, Morristown, NJ, 2004. Association for Computational Linguistics.

[213] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, 1997. Morgan Kaufmann.

[214] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2-3):219–241, 2002.

[215] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96–103, New York, NY, 2003. ACM Press.

[216] D. Yin, Z. Xue, X. Qi, and B. D. Davison. Diversifying search results with popular subtopics. In *The 18th Text REtrieval Conference*, Gaithersburg, MD, 2009. NIST.

[217] H. Yu, J. Han, and K. C.-C. Chang. PEBL: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

[218] O. R. Zaiane and A. Strilets. Finding similar queries to satisfy searches based on query traces. In *Proceedings of the International Workshop on Efficient Web-Based Information Systems*, volume 2426 of *LNCS*, pages 207–216, Berlin, Sept. 2002. Springer.

[219] S. Zelikovitz and H. Hirsh. Using LSI for text classification in the presence of background text. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 113–118, New York, NY, 2001. ACM Press.

[220] D. Zhang and W. S. Lee. Question classification using support vector machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 26–32, New York, NY, 2003. ACM Press.

[221] T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 821–826, New York, NY, USA, 2006. ACM.

[222] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 274–281, New York, NY, USA, 2005. ACM.

[223] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 487–494, New York, NY, USA, 2007. ACM.

[224] E. H. Zornitsa Kozareva and E. Riloff. Learning and evaluating the content and the structure of a term taxonomy. In *AAAI 2009 spring symposium "Learning by Reading and Learning to Read"*. AAAI Press, 2009.

[225] S. M. zu Eissen and B. Stein. Genre classification of web pages. In *Proceedings of the 27th German Conference on Artificial Intelligence*, volume 3238 of *LNCS*, pages 256–269, Berlin, 2004. Springer.

Vita

Xiaoguang Qi

1997	Graduated from No. 2 Middle School in Shijiazhuang, Hebei Province, China.
2001	B.E. in Computer Science and Engineering, B.A. in English, Xi'an Jiaotong University.
2004	M.E. in Computer Science and Engineering, Xi'an Jiaotong University.
2004 - 2011	Graduate study in Department of Computer Science and Engineering, Lehigh University.
2006	L. Nie, B. D. Davison, X. Qi. Topical Link Analysis for Web Search. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval, pages 91-98.
2006	X. Qi, B. D. Davison. Knowing a Web Page by the Company It Keeps. In Proceedings of the 15th ACM Conference on Information and Knowledge Management, pages 228-237.
2007	X. Qi, L. Nie, B. D. Davison. Measuring Similarity to Detect Qualified Links. In Proceedings of the International Workshop on Adversarial Information Retrieval on the Web.
2008	Research intern, Microsoft Live Labs, Bellevue, WA.
2008	X. Qi, B. D. Davison. Classifiers without Borders: Incorporating Fielded Text from Neighboring Web Pages. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research & Development on Information Retrieval, pages 643-650.
2009	X. Qi and B. D. Davison. Web Page Classification: Features and Algorithms. ACM Computing Surveys, 41(2).
2009	N. Dai, B. Davison and X. Qi. Looking into the Past to Better Classify Web Spam. In Proceedings of the Fifth International Workshop on Adversarial Information Retrieval on the Web, pages 1-8.
2010	Intern, Yandex Labs, Palo Alto, CA.
2010	X. Qi, D. Yin, Z. Xue and B. D. Davison. Choosing Your Own Adventure: Automatic Taxonomy Generation to Permit Many Paths. In Proceedings of 19th ACM Conference on Information and Knowledge Management, pages 1853-1856.

2011 N. Dai, X. Qi and B. D. Davison. Enhancing Web Search with Entity Intent. In Companion Proceedings of the 20th International World Wide Web Conference, pages 29-30.

2011 N. Dai, X. Qi and B. D. Davison. Bridging Link and Query Intent to Enhance Web Search. In Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, pages 17-26.

2011 X. Qi and B. D. Davison. Hierarchy Evolution for Improved Classification. In Proceedings of the 20th ACM Conference on Information and Knowledge Management, pages 2193-2196.